DESIGN OF SOFTWARE PACKAGE FOR INCORPORATION OF RANDOM LOAD TESTING AND DATA PROCESSING ON MATERIALS TESTING SYSTEM MACHINE.

Frederick Martin Blakely

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGN OF SOFTWARE PACKAGE FOR INCORPORATION OF RANDOM LOAD TESTING AND DATA PROCESSING ON MATERIALS TESTING SYSTEM MACHINE

by

Frederick Martin Blakely, Jr.

June 1978

Thesis Advisor: G. H. Lindsey

Approved for public release; distribution unlimited.

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) Design of Software Package for Incorporation of Random Load Testing and Data Processing on Materials Testing System Machine		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis: June 1978 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Frederick Martin Blakely		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1978 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified 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)

Spectrum Loading
Fatigue Monitor
Microcomputer
Microprocessor
Sequence Loading
Materials Testing System
Data Acquisition System

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

This thesis describes the software design and implementation of a microprocessor-based, random load drive and data acquisition system on a Material Testing System (MTS) machine.

A microprocessor, combination analog input/output module, magnetic cassette tape recorder, and strain gage network form a strain data acquisition system for recording sequential strain peaks and troughs on specimens subjected to flight load histories.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 | 1

(Page 1)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

The data will be used to estimate the fraction of the fatigue life expended in a test specimen.

A high speed digital computer is linked by telephone line to a microcomputer development system to create the randomization of fatigue loads specified in Mil Spec 8866 Spectrum A for use by the MTS machine.

Design of Software Package for Incorporation of Random Load Testing and Data Processing on Materials Testing System Machine

by

Frederick Martin Blakely
Lieutenant Commander, United States Navy
B.S., Naval Postgraduate School, 1970

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL

June 1978

ABSTRACT

This thesis describes the software design and implementation of a microprocessor-based, random load drive and data acquisition system on a Material Testing System (MTS) machine.

A microprocessor, combination analog input/output module, magnetic cassette tape recorder, and strain gage network form a strain data acquisition system for recording sequential strain peaks and troughs on specimens subjected to flight load histories. The data will be used to estimate the fraction of the fatigue life expended in a test specimen.

A high speed digital computer is linked by telephone line to a microcomputer development system to create the randomization of fatigue loads specified in Mil Spec 8866 Spectrum A for use by the MTS machine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author would like to express his sincere appreciation to all those who provided background material and technical information in the microprocessor and computer science fields.

Particular thanks are due to Dr. Gerald H. Lindsey, thesis advisor, for his direction and counsel; to LT. Cleveland D. Englehart, USN, for his technical knowledge and software programming techniques; to Mr. Ted Dunton, Chief Technician, for his tireless and cheerful troubleshooting; to Mr. John Drakeford, Industrial Control Project Manager, and Mr. Tom Rossi, System Design Engineer, Intel Corporation, for their aid in software application and product support.

Most of all, thanks to my family for their endless support and understanding.

I. INTRODUCTION

With the greater complexity and cost of new weapon systems, together with the general economic pressures to control expenditures, there is an urgent need to obtain maximum use from the operational lives of aircraft.

Currently, fatigue monitoring of naval aircraft is based on the total number of g readings recorded at four selected levels by an exceedence level counting accelerometer. Using microprocessors, it will soon be possible to record in sequence each maximum and minimum load level experienced by an aircraft. The data collected can be used to monitor the fatigue life of a structure via the determination of damage accumulated at a point found to be critical in a structural test of a prototype.

The objective of this research work was to design a software package to incorporate random load testing and data processing on a Materials Testing System (MTS) machine (Figure 1). The random load history would come from either monitored flight data or computer generated sequences.

The entire software package was written in two phases. The first phase consists of software for a single channel strain data acquisition system. This acquisition system is designed to process and record in sequence data originating from strain gages located at fatigue critical points. The data obtained will be sequential peaks and troughs that will be used to estimate the fraction of the fatigue life of the structure that has been expended.

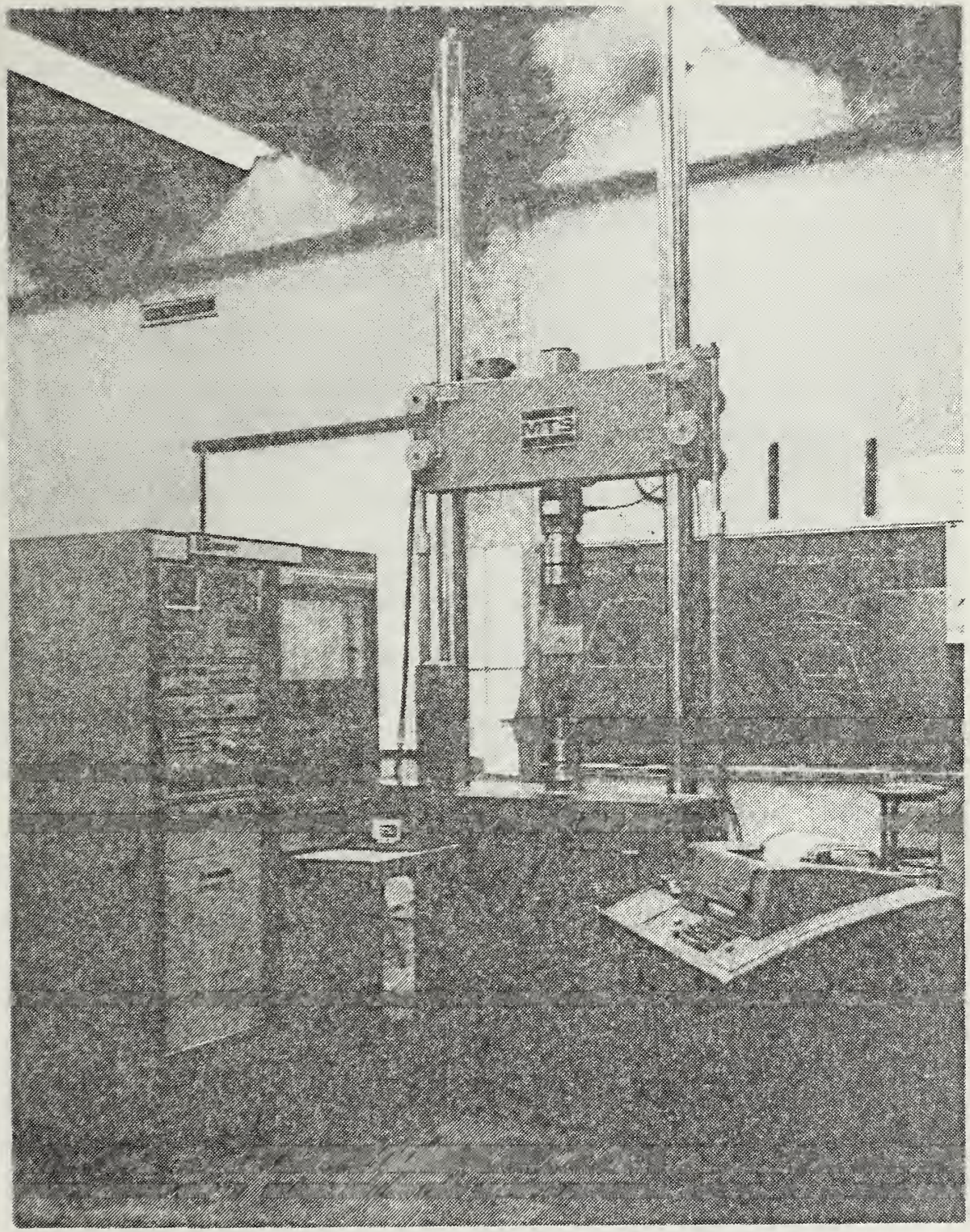

Figure 1. Materials Testing System (MTS)

The system developed is composed of two major subsystems: the Write subsystem and the Read subsystem. The Write subsystem is tasked with the collection of data. It makes use of a microprocessor, analog to digital converter, a magnetic tape recorder, and a strain gage network to monitor strain-generated signals, identify significant events and record the collected data. The Read subsystem is tasked with the retrieval and display of the data from the Write subsystem recorder.

Phase two of the software package consists of software necessary to incorporate randomization of MIL SPEC 8866 SPECTRUM A loads into a material testing system.

This system creates random loads on the IBM 360 computer, transmits load data via telephone line to the MDS 800 microcomputer development system, and finally punches the load data on punch paper tape, which supplies the random load sequence for the MTS.

The entire software package represents a smooth interface between the high level Fortran language used by the IBM system 360 and the low level assembly language of the system 80/10 microcomputer.

A glossary of terms commonly used in the instrumentation engineering, data processing and computing disciplines is presented in Appendix A.

II. STRAIN DATA RECORDING

A. HARDWARE COMPONENTS

1. Intel System 80/10 Microcomputer

The Intel 80/10 Microcomputer System is self-contained, utilizing the SBC-80/10 single board computer. The standard system 80/10 contains 1K (1K - 1024 bytes) of 8-bit read/write, Random Access Memory (RAM). Sockets for up to 4K of 8-bit words of non-volatile Read-Only-Memory (ROM) are provided in the system. The 8-bit Intel 8080A CPU is the central processor for the system 80/10. [Figure 2].

The 8080A contains six 8-bit general purpose registers and an accumulator. The six general purpose registers may be addressed individually or in pairs, providing both single and double precision operators.

The 8080A has a 16-bit program counter, which allows direct addressing of up to 64K bytes of memory. An external stack at memory location 7FFFH may be used as a last in/first out stack to store the contents of the program counter, flags, accumulator, and all of the six general purpose registers. A 16-bit stack pointer addresses the external stack. This provides subroutine nesting that is bounded only by memory size.

The system 80/10 contains 48 programmable parallel input/output (I/O) lines implemented by two Intel 8255 Programmable Peripheral Interface (PPI) devices. Software is used to configure the I/O lines in combinations of undirectional

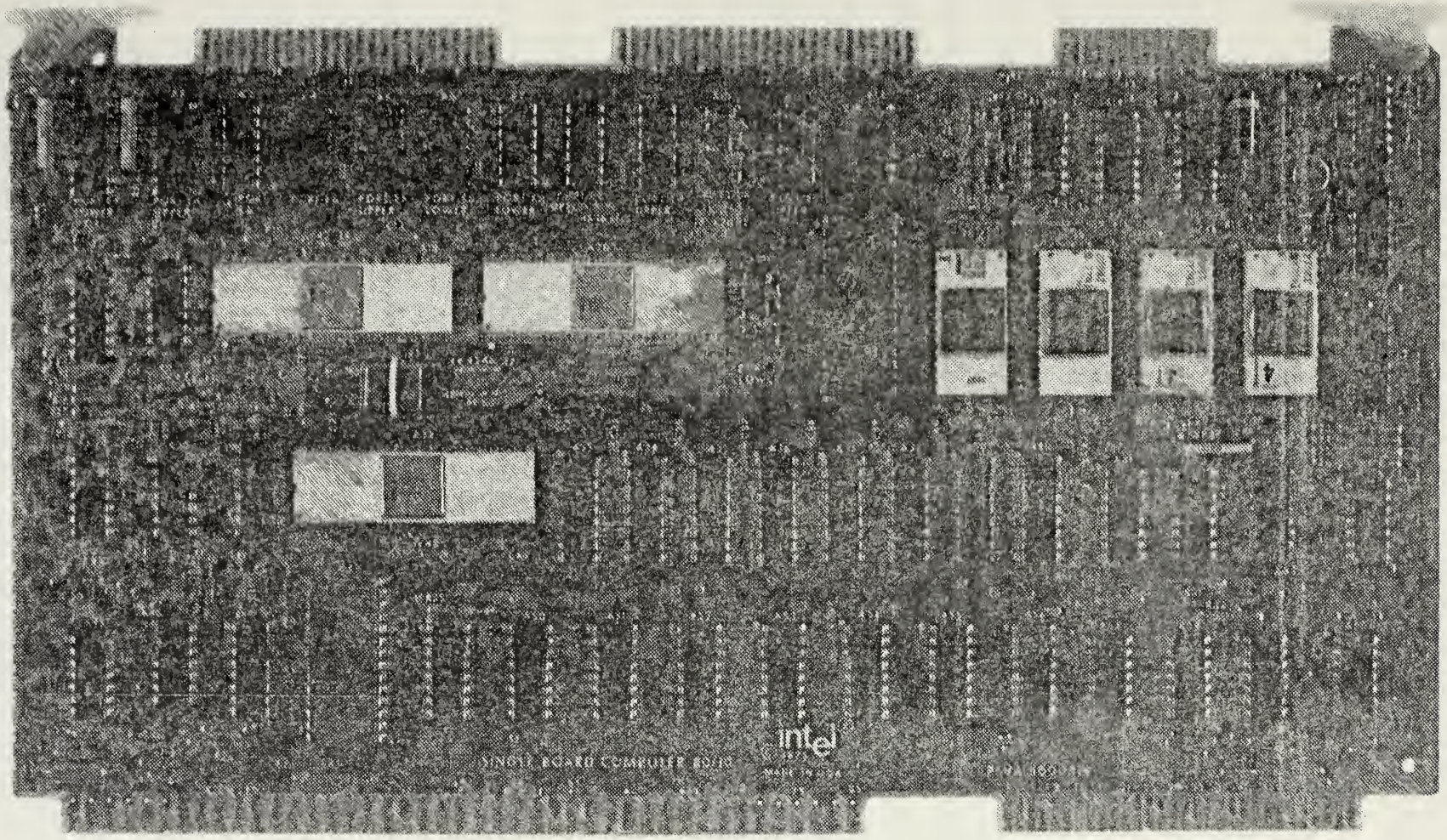

Figure 2. Single Board Computer 80/10.

parts and bidirectional parts. Section 3 will discuss in detail the setup of the Intel 8255 PPI devices used for this thesis.

2. System 80/10 Monitor Program

A standard feature of the system 80/10 is a software monitor, which is programmed in two ROM's. The monitor provides two basic capabilities: (1) it accesses console input/output routines as well as paper tape input/output control software; (2) the monitor, along with a teletype (TTY) or a cathode ray tube (CRT), provides the user with a console that furnishes immediate access to both memory and registers. It also has control commands to begin execution, display, or alter the contents of the memory or registers. The system monitor ROM's are positioned in the first two ROM sockets of the SBC-80/10 computer, occupying memory locations 0 to 2048. A more complete description of monitor commands is given in Ref. [1].

3. 8255 Programmable Peripheral Interface Device

The peripheral interface section of Group 2 contains 24 peripheral interface lines, buffers, and control logic. The characteristics and functions of the interface lines are determined by the operating mode selected under program control; three modes of operation may be selected:

MODE 0 - BASIC INPUT/OUTPUT

MODE 1 - STROBED INPUT/OUTPUT WITH INTERRUPT SUPPORT

MODE 2 - BIDIRECTIONAL BUS WITH INTERRUPT SUPPORT

Table I lists the basic features of the peripheral interface lines within each mode group. This thesis will utilize the mode 0 basic input/output mode only.

TABLE I

Features of Peripheral Interface Lines

Mode 0 — Basic Input/Output
Two 8-bit ports Two 4-bit ports with bit set/reset capability Outputs are latched Inputs are not latched
Mode 1 — Strobed Input/Output
One or two strobed ports Each Mode 1 port contains: 8-bit data port 3 control lines Interrupt support logic Any port may be input or output If one Mode 1 port is used, the remaining 13 lines may be configured in Mode 0. If two Mode 1 ports are used, the remaining 2 bits may be input or output with bit set/reset capability.
Mode 2 — Strobed Bidirectional Bus
One bidirectional bus which contains: 8-bit bidirectional bus supported by Port A 5 control lines Interrupt support logic Inputs and outputs are latched The remaining 11 lines may be configured in either Mode 0 or Mode 1.

The mode definition control word shown in Figure 3 is used to specify the configuration of the peripheral interface lines on the 8255 device. When the opcode field (bit 7) of the control word is equal to one, the control word is interpreted by the 8255 as a mode definition control word. The system softwear may specify the modes of port A and B independently as input or output when in mode 0. Port C may be treated as input only, output only, or divided into two portions of input and output, as required by the port A and port B mode definitions.

In group 2 the 8255 chip, with the control register address of EBH, was configured through the use of the mode control word interface as:

PORT A - MODE 0 OUTPUT

PORT B - MODE 0 INPUT

PORT C - MODE 0 INPUT (STATUS)

PORT C - MODE 0 OUTPUT (CONTROL)

The following mode control word was used:

```
1 0 0 0   0 0 1 1  (Binary)
    8       3      (Hex)
```

The assembly language program is:

```
CONTROL  EQU  OEBH;  8255  #2  ADDRESS
MVI      A,83H    ;  move control word into ACCM
OUT      CONTROL  ;  output to address OEBH
```

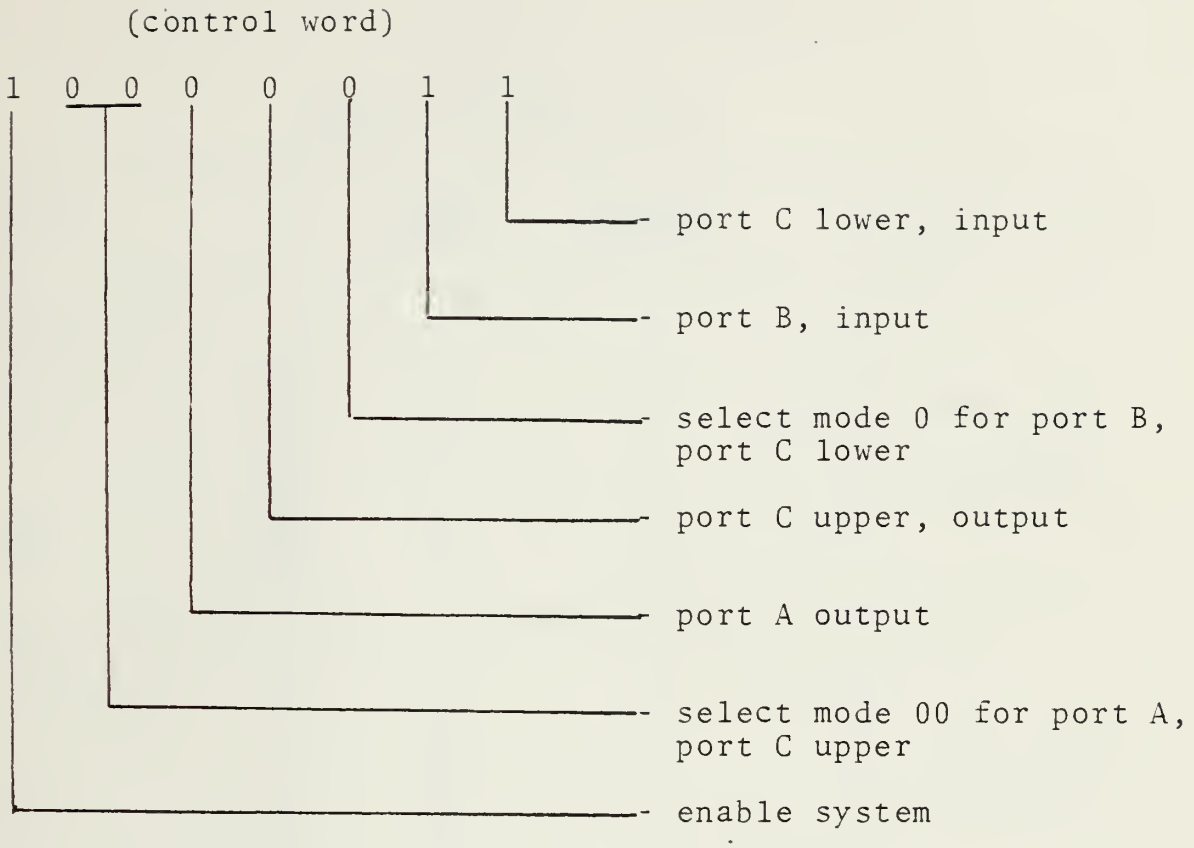

Figure 3. 8255 #2 Mode Control Word.

Now, let's take a closer look at port C since its functions are split between control and status monitoring. The address of port C is ØEAH, as shown in Table II. For example, 11110000, or ØFØH, will put the recorder in the Write mode and start writing characters. The assembly language program follows:

```
PORT C   EQU   ØEAH;  PORT C   ADDRESS
MVI      A,ØFØH    ;  PORT C   CONTROL WORD
OUT      PORT C    ;
```

TABLE II

Port Assignment 8255

SBC-80/10 ADDR	J2	Cable DB25	(Tape) Memodyne	Name/Function
Port A	43	2	M	2^0
	45	3	N	2^1
	47	4	U	2^2 DATA TO
	49	5	V	2^3
	51	6	W	2^4 RECORDER
	39	7	X	2^5
EB	37	8	Y	2^6
	35	9	19	2^7
Port B	Ø5	14	2	2^0
	7	15	C	2^1
	9	16	F	2^2 DATA FROM
	3	17	H	2^3
	11	18	J	2^4 RECORDER
	13	19	K	2^5
E9	15	20	10	2^6
	17	21	11	2^7
STATUS				
Port C	25	25	20	STATUS
	23	10	8	$\overline{\text{TAPE SYNC}}$
EA	21	11	S	$\overline{\text{CIP}}$
	19	12	17	$\overline{\text{BEOT}}$
CONTROL				
Port C	27	13	A	LWD FWD
	29	22	L	$\overline{\text{RWD}}$
EA	31	23	12	START/$\overline{\text{STOP}}$
	33	24	22	WR/$\overline{\text{RD}}$
		1	14	GND

4. SBC-732 Combination Analog Input/Output Board

The SBC-732 is an analog input/output subsystem which, under microprocessor control, performs the basic functions of data acquisition of analog inputs and controlled analog output signals [Fig. 4]. There are three programmable modes of operation for the acquisition of analog inputs: repetitive single input, sequential channel scan input, and random channel input.

The 732 multiplexer can accommodate 8 differential or 16 single-ended analog input channels. All input channels are pretested to ±28V by clamping diodes and fusible current-limit resistors.

The selected differential or single-ended input is applied to the analog-to-digital connection (ADC) via a programmable gain amplifier, which under program control was selected to provide a gain of 1. The ADC is a 12-bit, 35.7 microsecond, successive approximation device with an internal sample-and-hold (S/H) amplifier. The ADC was jumper selected for ±10V full scale inputs. The A/D conversion process was initiated by program command.

Each of the two 12-bit non-isolated digital-to-analog converters (DAC's) were configured for ±10 volts full-scale voltage output. Table III contains SBC-732 board specifications.

5. Memodyne Model 173 Cassette Recorder

The recorder is a memodyne model 173 magnetic tape recorder. The model 173 is a parallel input/output, read/write unit designed to be compatible with ASCII requirements.

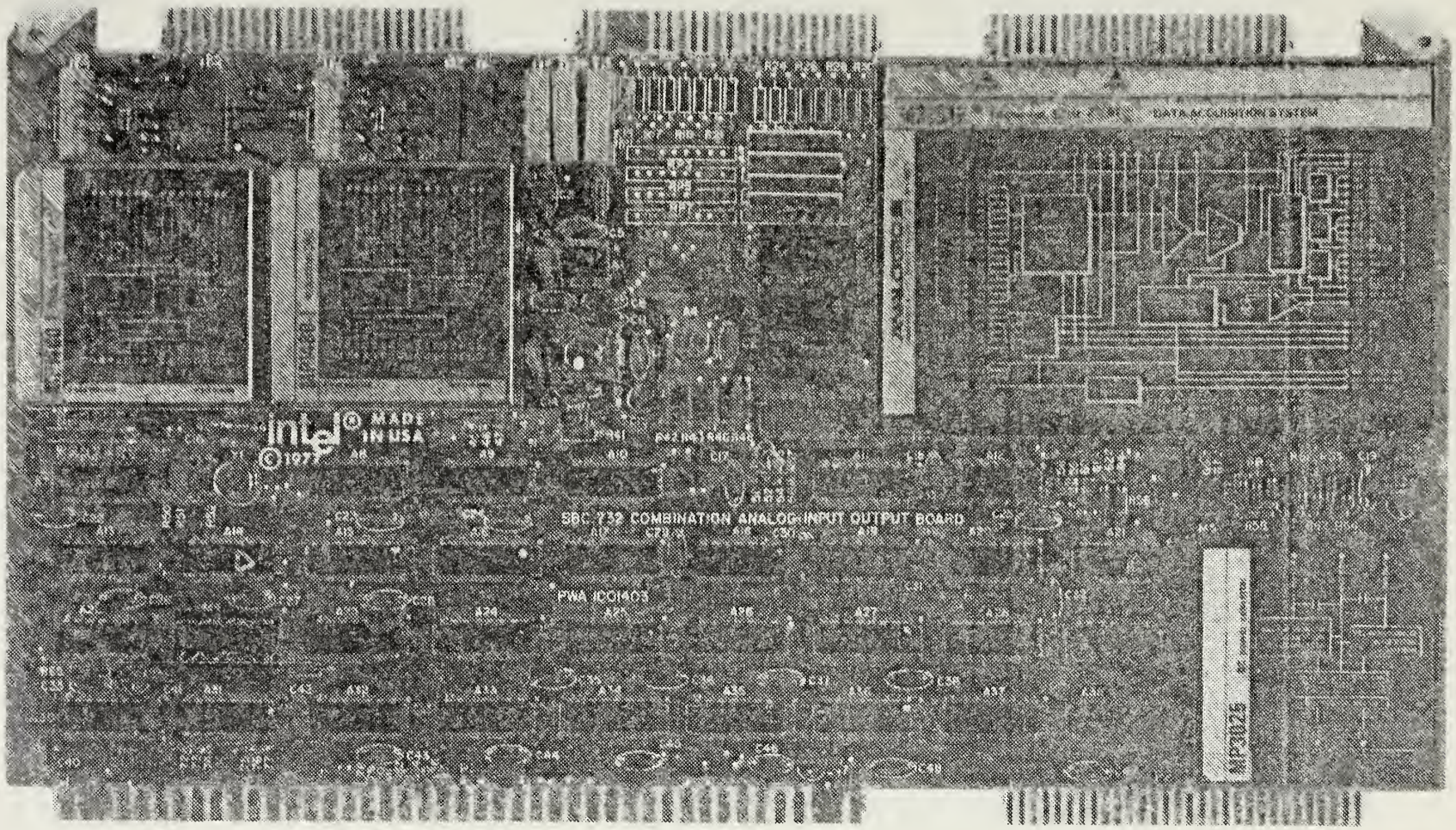

Figure 4. SBC 732 Combination Analog - Input/Output Board.

TABLE III

SBC-732 Specifications

Power Requirements:	V_{CC} = +5V ± 5%
	I_{CC} = 2.5A maximum
Physical Characteristics:	
Width:	34.48 cm (12.00 inches)
Depth:	17.15 cm (6.75 inches)
Thickness:	1.27 cm (.50 inches)
Weight:	567 gm (20 ounces)
Addressing:	Reserves a block of 16 contiguous memory locations relating to a jumper-selectable memory base address.
Analog Input:	
Number of Channels:	8 differential or 16 single-ended; expandable to 16 differential or 32 single-ended
Resolution:	12 bits (.025%), bipolar or unipolar
S/H Aperture Time	< 20 nanoseconds
S/H Uncertainty	5 nanoseconds
Overall Accuracy (25°C)	0.05% FSR ± 1/2 LSB (Gain 2)
A/D Conversion Speed	28 KHZ
Throughput:	
Sample Rate (single channel)	17 KHZ
Channel-to-Channel Rate	16 KHZ
Analog Output:	
Number of Channels	Two, non-isolated
Resolution	12 bits, bipolar or unipolar (jumper selectable)
Voltage Output Characteristics:	
Output Ranges	+5V, +10V, ±5V, ±10V (jumper selectable)
Output Current	5 MA @ ±10V
Output Impedance	0.2 ohm

When writing, this mode accepts 7-bit parallel input data and formats the data word into serial format suitable for recording on the tape. When reading, a start command will cause one 8-bit character to be read and will present this data in parallel format at the output. Figure 5 shows the cassette tape recorder front panel and controls.

B. SOFTWARE PACKAGE

1. Tape File Format

The purpose of the file format is to control input and output data in a manner that will permit the determination of the quantity and the accuracy encountered in the Read/Write operations.

All data stored on the tape will be arranged in the file format illustrated in Figure 6. The three leading zeroes in each record of the file provide the means for a simple test to determine that the file type character is approaching. The next character after the leading zeroes is the file type character. This type of designation enables the specific identification of each file. Record type 1 refers to source data that was obtained from actual flights; on the other hand, record type 2 refers to source data derived from computer generated flight loads. The next character in a record is the record length, which indicates the quantity of bytes which comprise the remainder of the record (excluding the terminal checksum character). The record length character permits the writing of specific length Write/Read routines

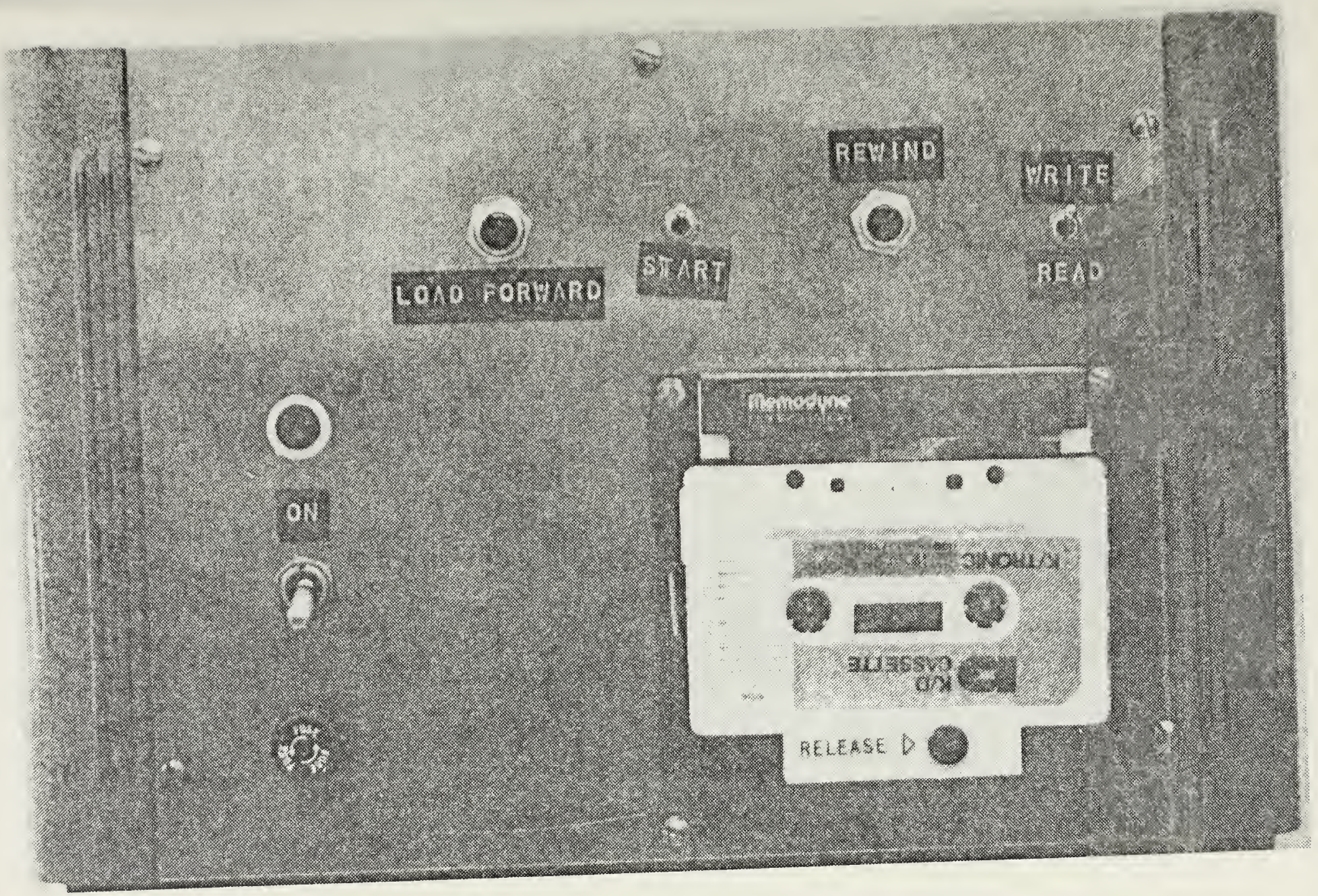

Figure 5. Memodyne Model 173 Cassette Recorder.

Leading Zeroes	Record Type	Record Length	Data	Checksum
000	01	81H	Julian Date Aircraft Type BUNO Configuration Gross Weight Mission	CS
000	02	94H	MTS Scale Factor Limit Load Cross Sectional Area Random Number Seed Strain Data Follows	CS
000	03	(00-FF)Hex	(Strain Gage Data)	CS
000	04	ODH	End of File	CS

Figure 6. Tape File Format.

and ensures that the proper quantity of data is transferred to/from the tape. The final character in a record is the checksum, which is a Modulo 257 sum of the data in the record between the file length character and the checksum itself. As the data are written on the tape, the checksum is calculated and then written at the end of each record to which they pertain. As the data are read from the tape, a new checksum is calculated, computed with the previously calculated checksum and, if an error exists, a CHECKSUM ERROR message is printed. This error would indicate that a difference exists between the data that were written on the tape and those which were read from the tape. A rewind and rerecord operation is then at the discretion of the operator.

2. The Write Program

The modularity of the design of the program permits a very simple conceptual construction of the Write main program. Subroutines are called to perform the "work" of the program, while the main program itself acts as the "manager." The operation of the main Write program is as follows:

The type of source load data, actual or simulated, determines whether record type 1 or 2 is accepted into RAM. The recorder head is moved onto the tape oxide in preparation to write. For each file, only three records are written on the tape. The tape is stopped after the fourth record is written. The system 80/10 returns to monitor for future control of the operation.

a. Write a Record Subroutine

The "write a record" subroutine (WRREC) writes the individual records on the tape. Operation of the subroutine is as follows: 1) The address in RAM that starts the record data chain is loaded into the H,L register pair. 2) The record type character is loaded into the D register. 3) The record length is loaded into the C register. 4) The subroutine is called. 5) A gap is put on tape for the physical separation of records. 6) The checksum is initiated. 7) The three leading zeroes, file type and file length are written on the tape. 8) One data byte is written on the tape. 9) The checksum is updated. 10) The record length counter is decremented. 11) If the record counter has not reached zero, indicating that all data have been transferred, the program continues looping through step 8. If the record counter has reached zero, all data have been transferred; and 12) the checksum is written on the tape. 13) Finally, another gap is put on the tape for further physical separation of files.

b. Write a Character Subroutine

This subroutine (WRCHAR) takes a byte of data from the accumulator and writes it on the tape. The operation of the "write a character" subroutine is as follows: 1) the tape recorder head is positioned near the tape oxide; 2) a write/start signal is sent to the tape recorder control port (the signal must be present for approximately one millisecond for the recorder to recognize it); 3) the status bit is sampled: if it is high, the recorder is still writing and the sampling

continues; when the status bit goes low, the recorder has completed the Write operation; and 4) a five millisecond delay is activated to ensure that the tape recorder is prepared to accept another write/start signal initiating the writing of the next data bit.

3. The Read Program

The Read main program also "manages" the Read subroutines. The operation of the Read main program is as follows: 1) the tape recorder head is positioned near the tape oxide; 2) the RAM address at which the tape data will be stored is loaded into H,L register pair; 3) all records on the tape are read with each record type character being checked for the "end of file" record character (in this program, record type 4 indicates the end of the file). If the file type 4 is sensed, 4) the "end of file" message is printed and the 80/10 is returned to the calling program.

a. Read a Record Subroutine

This subroutine reads a record from the tape, stores the data in RAM, and outputs the data to the output device (CRT or TTY). The operation of the READ A RECORD subroutine is as follows: 1) the RAM storage address is loaded into the H,L register pair; 2) incoming data is checked and rejected until the input of the record leading zeroes; 3) the leading zeroes are noted, but rejected as data; 4) the record type character is accepted as the first bytes of significant data. If the record type is 4, the END OF FILE message is printed and 80/10 is returned to monitor. If the record type

is not 4, then 5) the record length character is accepted, indicating the length of data to be read from this record. 6) As each data byte is read, stored and output to CRT or TTY, a new checksum is calculated. After all data have been read, 7) the new checksum is compared with the previous checksum stored on tape, and if in error the CHECKSUM ERROR message is printed. If no error, 8) control is returned to the calling program.

b. Read a Character Subroutine

This subroutine takes a byte of data from the tape and moves it into the accumulator. The operation of the READ A CHARACTER subroutine is as follows: 1) the read/start signal is sent to the tape recorder control port for at least one millisecond; 2) the tape SYNC bit is sampled until it is high, indicating that the tape recorder has commenced reading; 3) the tape SYNC bit is sampled until it is low, indicating that the tape recorder has completed the Read operation; 4) a 5-millisecond delay is provided to ensure that the tape recorder is ready to accept the next read/start signal for the next character; 5) the byte of data is sent from the tape recorder output port to the accumulator; 6) control is returned to the calling program.

4. ADC and DAC Programming

a. General Description

This section illustrates and describes the command, status, and data formats for programming the ADC and DAC channels. A more complete description can be found in Ref. [2]. Also see Table IIIA for pin assignments.

TABLE IIIA

SBC 732 ANALOG OUTPUT/INPUT
PIN ASSIGNMENT

Edge Connector/ Pin Number	Function	EIA	Comments
J1/36	DAC 1: V_{out}	19	
J1/39	DAC 1: Analog Rtn		Analog
J1/42	DAC Ø: V_{out}	18	Output
J1/45	DAC Ø: Analog Rtn		
J2/4	Channel Ø	1	
J2/6	8	9	
J2/8	1	2	
J2/10	9	10	
J2/12	2	3	
J2/14	10	11	
J2/16	3	4	Analog
J2/18	11	12	Input
J2/20	4	5	
J2/22	12	14	
J2/24	5	6	
J2/26	13	13	
J2/28	6	7	
J2/30	14	16	
J2/32	7	8	
J2/34	15	17	
J2/3-33	Analog Rtn	21-25	
J2/39-45	Digital Common	1	
J2/40	Clock Out	2	
J2/42	Ext. Trigger In	3	
J2/44	EOC Trigger Out	4	
J2/46	EOS Status Out	5	
J2/48	Analog Return	6	

The system 80/10 communicates with SBC-732 through a sequence of Read and Write commands. Table IV lists the individual commands associated with the SBC-732 ADC and DAC's. These commands are addressed as specific memory locations relative to the memory base address F700H. If, for example, the memory base address is F700, the address M + A implies the specific memory address F70A.

b. Read/Write Formats

The multiplier (MUX) address and gain format is shown in Figure 7. Bits 0-4 select the desired channel. Starting channel for a random ohms, bit 5 is ignored, and bits 6-7 select the programmable gain amplifier (PGA) input voltage gain. Table V lists programmable gain versus ADC full-scale range available to SBC-732 board.

The MUX address and gain are established by performing a Write to M + 1; the MUX address and gain may be verified by performing a Read of M + 1.

c. Command Register Format

The command register, which is associated entirely (either directly or indirectly) with the A/D conversion process, is loaded by a Write command to M + 0. Bit 0 must be set before the A/D conversion can occur. The command register format is shown in Figure 8.

d. Status Register Format

The contents of the status register, which contains the status of the ADC and the function associated with the A/D conversion process, are assessed by a Read command

TABLE IV

SBC-732
MEMORY ADDRESS ASSIGNMENTS

MEMORY ADDRESS	COMMAND	FUNCTION
M+0	Write	Load Command Register
M+0	Read	Read Status Register
M+1	Write	Load MUX Address Register and Gain Register
M+1	Read	Read MUX Address Register and Gain Register
M+2	Write	Load Last Channel Register
M+3	Write	Clear Interrupts
M+4	Read	Read Lower Byte of ADC Value
M+5	Read	Read Upper Byte of ADC Value
M+8	Write	Output Lower Byte for DAC0 to Hold Register (HR)
M+9	Write	Output Upper Byte to DAC0 (DAC0←HR automatically)
M+A	Write	Output Lower Byte for DAC1 to Hold Register (HR)
M+B	Write	Output Upper Byte to DAC1 (DAC1←HR automatically)

TABLE V

Programmable Gain Vs. ADC Full-Scale Range

ADC FULL-SCALE RANGE				GAIN
+5V	+10V	±5V	±10V	
+5V	+10V	±5V	±10V	X1
+2.5V	+5V	±2.5	±5V	X2
+1.25V	+2.5V	±1.25	±2.5	X4
+0.625V	+1.25V	±0.625V	±1.25	X8

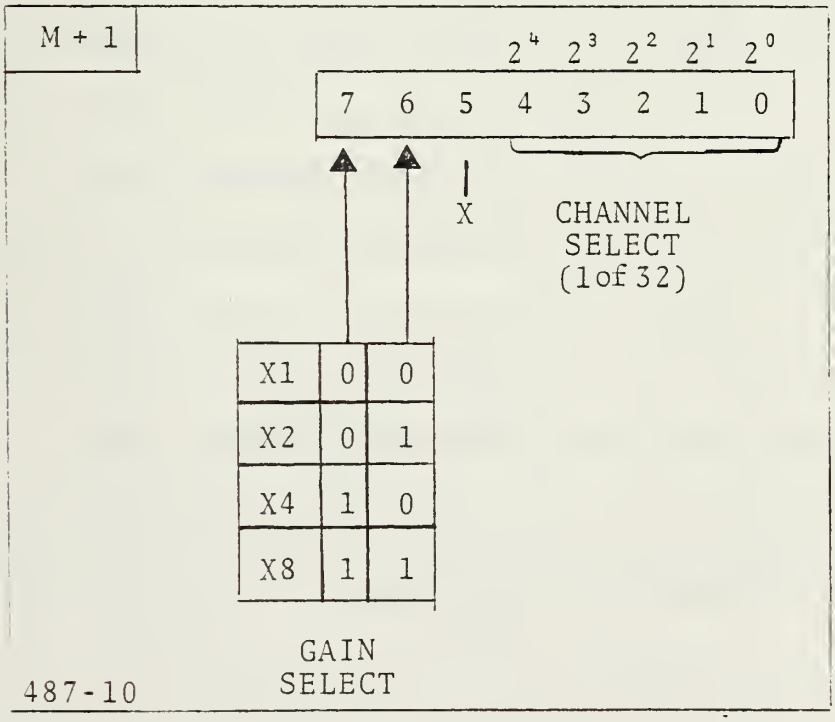

Figure 7. MUX Address and Gain Format.

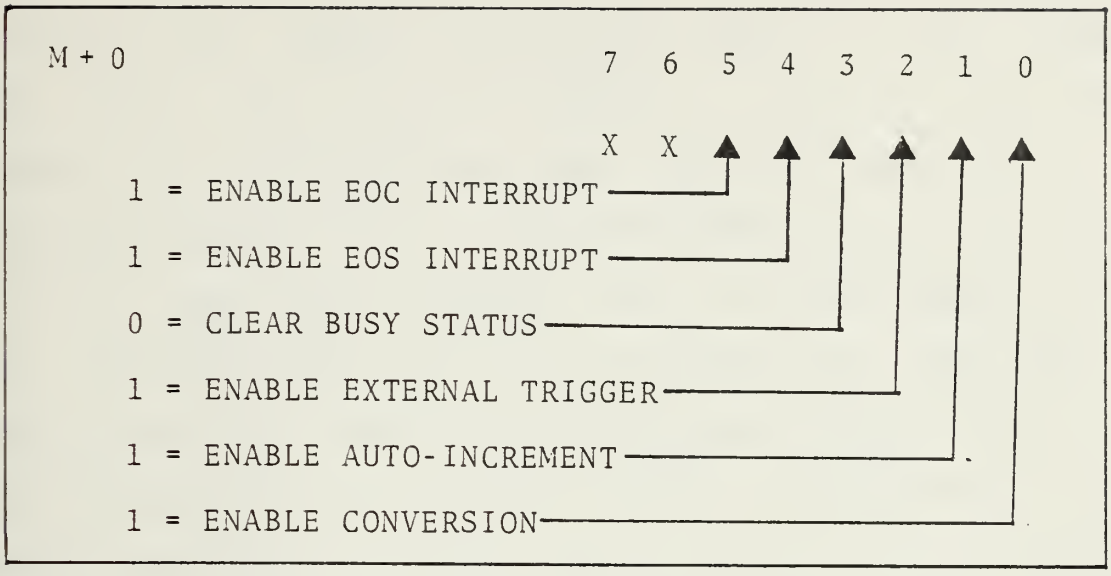

Figure 8. Command Register Format.

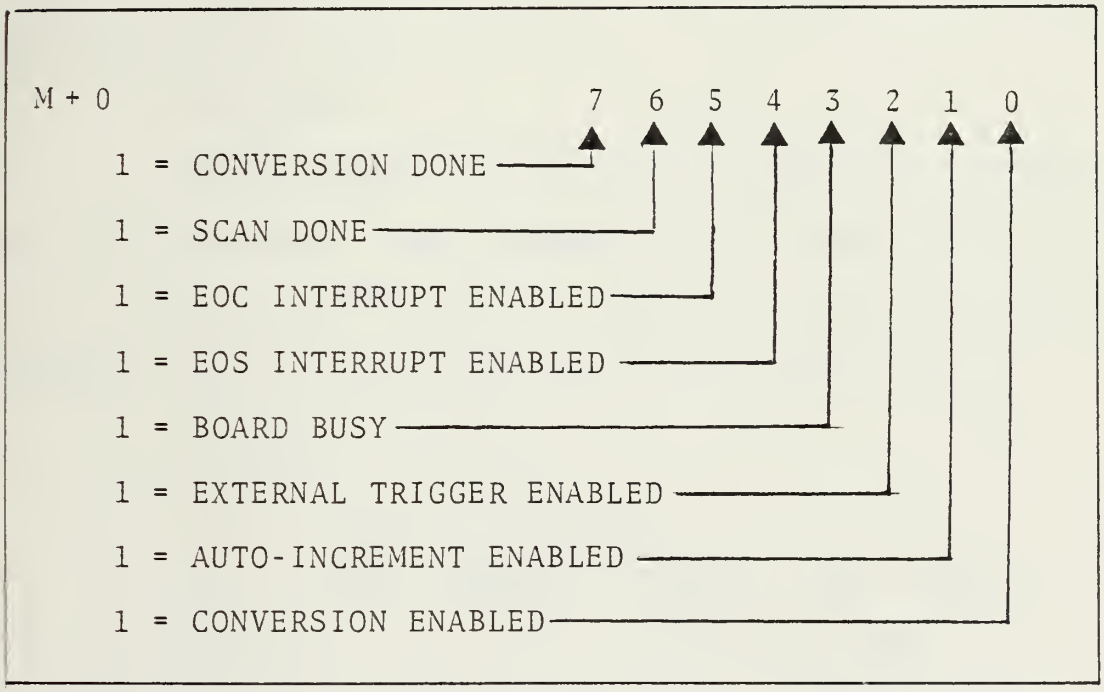

Figure 9. Status Register Format.

to M + 0. As shown in Figure 9, bits 0 through 5 essentially verify the last command word written into M + 0. Bit 3 (BOARD BUS), however, has a special function. The first time a Read command to M + 0 is performed after the busy status bit is cleared by a Write to M + 0, the BOARD BUSY bit will be read as a "0". Each time thereafter that the status register is read, the BOARD BUSY bit will be returned true. This function is useful for multiprocessor systems in which two or more processors are sharing the SBC-732. The BOARD BUSY bit can be cleared only by Write command to M + 0 with bit 3 clear. Bits 6 and 7 are used primarily in non-interrupt driven programs to determine when valid ADC data are ready to be read (CONVERSION DONE).

e. ADC Data

After the A/D conversion is complete, the data word is obtained by a Read command to M + 4 and a Read command to M + 5, as shown in Figure 10. M + 4 contains the ADC bits 0 through 3 and M + 5 contains ADC bits 4 through 11.

ASSEMBLY LANGUAGE:

```
ADDATA  EQU  BASE + 4
LHLD  ADDATA ;  LOAD HL  REG  WITH
                ADC  DATA  WORD
```

f. DAC Data

No program control other than the output data word is required by the two DAC channels. For DACØ, a Write command must be given first to M + 8 to load the low byte into

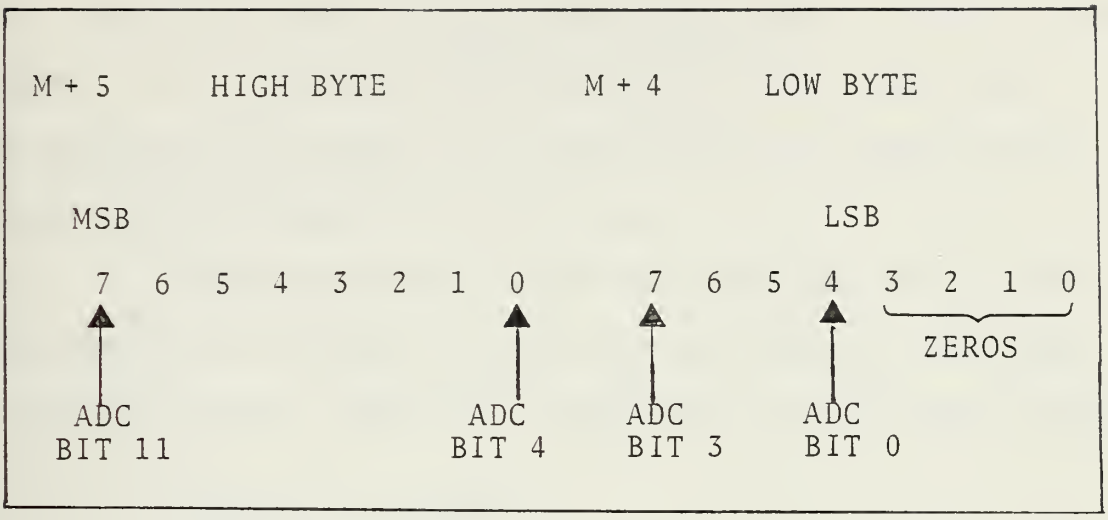

Figure 10. ADC Data Format.

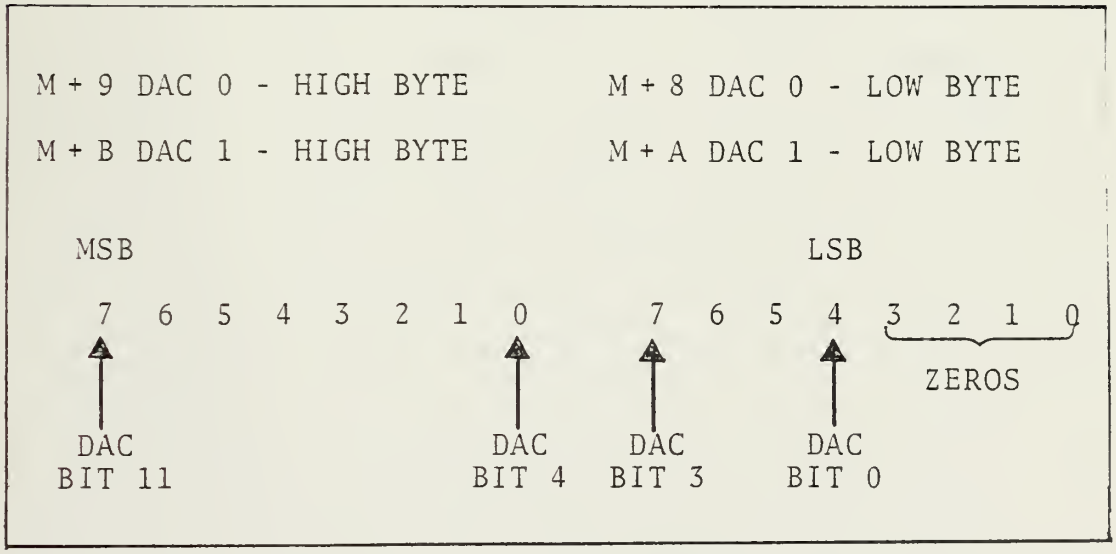

Figure 11. DAC Data Format.

the Hold Register, and the Hold Register maintains the four least significant bits valid at the DACØ input; then a Write command to M + 9 presents the high byte to the DACØ input. DACØ automatically makes the conversion of the input data when the Write command to M + 9 occurs.

The operation of DAC1 is identical to that of DACØ except M + A is used for the low byte and M + B is used for the high byte. Figure 11 illustrates the DAC data format.

ASSEMBLY LANGUAGE:

```
        DACPFS  EQU  OFFFØH;  DAC  POSITIVE FUL1 SCALE
        LXI     H, DACPFS   ;  LOAD HL  FFFØH
        CALL    WRDACØ      ;
                .
                .
                .
        WRDACØ: SHLD  DACØ  ;  OUTPUTS POSITIVE FULL
                               SCALE
                RET                    (FFFØ)
```

III. RANDOM LOAD TESTING

This section deals with the load sequence randomization technique developed by Lt. John Scott Atkinson, Jr.,[Ref. 3]. Fortran arithmetic declarations and manipulations required for 8080 CPU compatibility; the communications interface program called "HEXLINK" which links the MDS-800 to the IBM-360 via telephone line; and the preparation of paper tape of random load data using modification of DUMP program written by Digital Research for CP/M.

A. LOAD SEQUENCE RANDOMIZATION TECHNIQUE

The load sequence randomization technique uses the computer library subroutine RANDU to place 10 percent of the MIL-SPEC 8866, spectrum A positive loads [Table VI] in a random order. Each of the positive loads is paired with a minimum load of 11 percent limit load, representing 1-g flight. During the randomization, each load has an equal probability of selection. A counter restricts the number of times a value is selected to the number of occurrences of the particular load level in MIL spectrum A.

B. RANDOM LOAD ACCURACY AND FORMAT

The accuracy and format of the random loads were dictated by the 16-bit double precision accuracy of the 8080A CPU and the ADC and DAC 12-bit data word format.

TABLE VI

FREQUENCY OF MANEUVER LOADS

Number of Times per Thousand Hours
that Load Factor is Experienced

Percent of Maximum (Positive) Symmetrical Limit Load Factor	Flight Maneuver Load Spectrum A
35	17,000
45	9,500
55	6,500
65	4,500
75	2,500
85	1,360
95	440
105	150
115	40
125	16
Total	42,006

1. 8080A CPU Double Precision Accuracy

Since the 8080A CPU can handle 16-bit binary data, the loads generated in Fortran algorithm were limited to 16-bit accuracy by simply declaring them INTEGER *2.

2. ADC and DAC Data Word Format

The ADC and DAC data word is formed using double precision (2 bytes), referred to as the high byte and the low byte. The 12 most significant bits are used in the conversion process, thus leaving the least significant 4 bits of the low byte with zeroes. This nibble was used to indicate the

designated channel (0-15). The ADC and DAC word is shown in figure 12.

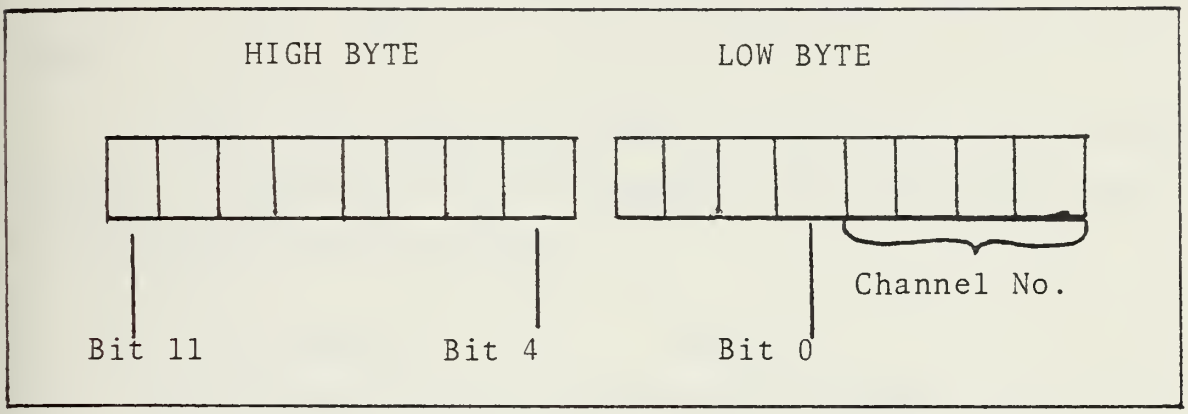

Figure 12. ADC and DAC Data Word.

3. Conversion of Loads to Voltages

The next step in the analysis was to correlate a load generated with a corresponding voltage. The MTS requires ±10 volts to operate its four selectable load ranges. The load ranges are: ±10,000, ±20,000, ±50,000, and ±100,000 lbs. The range is selected by estimating the maximum stress expected to occur. For example, if the 20,000 lb. range is selected and +10 volts is applied, then the MTS will produce a 20,000 lb. load. Using 12 bits, the largest number that can be represented is 16 x 16 x 16 = 4096 parts. This means that if the voltage range is ±10 volts, then there would be 4096 incremental steps between the voltage limits. Since our investigation will deal with positive loads only (0 to +10 volts), the incremental range is now 2048 steps. In other words, there are 204.8 steps per volt in the 0 - 10 volt range.

Now, to determine the number of steps required for a specific voltage supply, multiply the voltage by 204.8 steps/volt: for example, 9.375 volts is equivalent to 1920 steps.

Next, the number of steps is represented in a 16-bit pattern. Continuing with our example, the 1920 steps have the following 16-bit pattern.

BYTE 1		BYTE 2		
0 0 0 0	0 1 1 1	1 0 0 0	0 0 0 0	= 1920

Since the analog converter data word used the three most significant nibbles of the combination of Byte 1 and Byte 2, the above 16-bit pattern is shifted 4 bits to the left. This is accomplished by multiplying the above 16-bit pattern by 16. The 16-bit pattern now is as follows:

0 1 1 1	1 0 0 0	0 0 0 0	0 0 0 0

.

The representation of 0 volts by the ADC or DAC is 8000H or (1000 0000 0000 0000) in 16-bit binary pattern. This means that the least value of the most significant nibble is eight (1000). This means that 1 is added to the most significant bit, giving the following 16-bit pattern:

1 1 1 1	1 0 0 0	0 0 0 0	0 0 0 0

The same results can be achieved by adding 32,678 to the 16-bit pattern.

The Hexdecimal representation of the above 16-bit binary number is F800. In summary, if F800 is outputted via DAC channel, it will produce a voltage of 9.375 volts, which corresponds to a load of 18,750 lbs. on the 20,000 lb. MTS load range.

4. Example

Suppose the critical stress analysis of a specimen with 0.9 $in.^2$ cross-sectional area had a limit load of 14,000 PSI. The largest load in the spectrum will be

$$1.25 \times 14{,}000 \times .9 = 15{,}750.$$

Selecting the 20,000 lb. load range and multiplying by 10 volts,

$$\frac{15{,}750 \text{ lbs.}}{20{,}000 \text{ lbs.}} \times 10 \text{ volts} = 7.875 \text{ volts.}$$

In other words, 7.875 volts is equivalent to 15,750 lb. load, when the 20,000 lb. range is selected on the MTS.

To determine the number of steps from zero, multiply by 204.8 steps/volt, and add 2048 (difference between 8000H and 0000H. The result is 3660.8 steps with respect to 0000H. Converting to hexidecimal gives EaDX, where X designates channel number.

Now that the random load data have been created and stored in a vector, a technique for transmitting the data to the microcomputer development system needs to be considered.

C. COMMUNICATIONS LINK BETWEEN IBM-360 AND MDS-800 VIA TELEPHONE LINE

1. CP/CMS Control Program-67/Cambridge Monitor System

The CP-67/CMS time-sharing system consists of two independent components: the control program (CP67, or CP for short) and the Cambridge monitor system (CMS). The control program creates the time-sharing part of the system to allow many users to access the computer simultaneously. The Cambridge Monitor System provides the conversational part of the system to allow a user to monitor his work from a remote terminal.

CMS gives the user a full range of capabilities--creating and managing files, compiling and executing problem programs, and debugging, using only a remote terminal. For a complete description of CP/CMS capabilities, see references [4] and [5].

2. MDS-800 Microcomputer Development System

In its basic configuration, the INTEL MDS-800 Microcomputer Development System consists of a CPU, 16K RAM, peripheral interface controller, front panel controller, power supply and enclosure. The MDS was directly connected to the following peripheral devices: CRT and keyboard console, high speed line printer, standard teletype with paper tape reader and punch [Ref. 6].

3. Disk Operating System

The INTEL Disk Operating System (DOS) consists of three major components, a dual floppy disk drive unit, a disk controller, and the DOS support software.

Each 7.5 inch diameter floppy disk (diskette) had a capacity of 256K bytes of semi-random access storage. With the dual drive, over 0.5 million bytes of data, programs, or other information could be assessed with relative ease and moderate speed.

The software support package offered by Digital Research was chosen as the operating system. CP/M consists of several utility routines in addition to the Basic Disk Operating System (BDOS). These routines allow the user to form and edit disk files, programs, or data files, and to assemble and load assembly language programs, and provide a powerful debug routine. A more complete description of the CP/M BDOS is contained in Ref. [7].

4. Hexlink Program

Assembly language was used in the construction of the communications interface program "LINK." "LINK" was developed by Lt. Mack T. Elliott, USN [Ref. 8] to interface the MDS-800 (and Model 40 printer) with CP/CMS through a 1200-band telephone line. Both the line and the printer are driven by an 8251 USARTS incorporated in a SBC-534 I/O board. LINK operates in one of three modes:

a. Direct link up mode

b. Transmit file mode

c. Receive file mode.

Our discussions will be limited to the Receive File Mode. The Receive File Mode issues all CP/CMS commands to effect the transfer of an entire P-Disk File to the floppy disk.

It was necessary to modify "LINK" in its original form to include special filtering techniques of characters so that load data would be in hexidecimal format. This means that all characters other than (0-9, A-F) were excluded. It was still necessary to employ additional filtering because of the CP/CMS DumpF command output format.

5. CP/CMS DumpF Command

CP/CMS DumpF command types the contents of all or part of a specified file in hexidecimal format. The output format is as follows:

```
RECORD     1   LENGTH = 512
(Hexidecimal Data)
```

The additional filtering is required because the E, C, and D in the word "Record" would slip through the filter, since these letters are between A and F. These letters would pair up and pass as valid data, i.e., E and C would form together the hexidecimal byte value of "EC." Now, taking the entire first line, the following characters would be accepted as valid data: EC D1, E5 12, followed by normal data. It can be readily seen that when the record count becomes double digit, i.e., 10 or greater, the number of characters to pass through the existing filter system will be odd, thus allowing an invalid character to form with valid data. This has the effect of shifting valid data to the right one character, completely changing the meaning of the data. The solution to this problem was to check for character R, then filter out

all characters until carriage return line feed was sensed, which comes after the 512 in the above example. This ensures that all characters that pass through are valid data. With the above filtering techniques, the system is ready to pass data from CP/CMS to the Microcomputer Development System via telephone line. The procedures are outlined below:

```
USER    :  CONTROL  R
HEXLINK :  CMS  FILENAME  FILETYPE
USER    :  FILE  FT09F001
HEXLINK :  DISK:  FILENAME  FILETYPE
USER    :  A : DATUM1  HEX
HEXLINK :  DUMPF
           RECEIVING
            .
            .
            .
HEXLINK >  TRANSMISSION COMPLETE
           0200  RECORDS  TRANSMITTED
```

D. PUNCHING LOAD DATA ON PAPER TAPE

1. CP/M

CP/M is a monitor control program for microcomputer system development which uses IBM compatible flexible disks for back-up storage. Using a computer mainframe based upon Intel's 8080 microprocessor, CP/M provides a general environment for program construction, storage, and editing, along with assembly and program check-out functions [Ref. 9].

2. CP/M Dump Command Modification (Dump1)

The Dump program was developed by Digital Research Corp. and is compatible with the CP/M system. Dump types the contents of the disk file at the console in hexadecimal form. The file contents are listed 16 bytes at a time with the absolute byte address listed to the left of each line in hexadecimal.

Dump1 is a modification to DUMP program that types into memory, beginning at address 4400H, the contents of the disk file. This modification was necessary in order to punch a paper tape of data.

3. Punching Paper Tape Command

The MDS-800 monitor commands, specifically W4400, F700, will punch a paper tape beginning at address 4400H and terminating at F700H. The tape will be punched in standard INTEL HEX FORMAT. The data generated on the IBM-360/67 is now on paper tape in proper INTEL format, ready to be read into the system 80/10 via TTY.

IV. SYSTEM OPERATION

A. SUBROUTINES

Appendix C contains a complete source listing of all subroutines, including a branch table with PROM and memory address locations. Selected subroutines will be discussed in detail because of complexity and function. The less complex subroutines are self-explanatory. Other subroutines, such as WRCHAR, WRREC, RDCHAR, and RDREC, are covered in a separate section.

The following subroutines will be discussed in detail: VALDT, SGLCHN, HEADING, DSFILE, and RAMP.

1. VALDT Subroutine

The VALDT subroutine is designed to determine whether an event is strain significant or not. VALDT was intended to be used in conjunction with SGLCHN and STORE subroutines.

The change in magnitude of a signal is the first indication that an event may be strain significant.

The sign of the slope of the change is determined and compared with the sign of the slope of the previous change. Any change in sign will identify a peak or a valley in the analog signal and is further indication that a strain-significant event may have occurred. Furthermore, to be significant, the value of the strain reading must be above a predetermined positive threshold or below a predetermined negative threshold.

2. SGLCHN Subroutine

The SGLCHN subroutine uses the variables FSTCHN and GAIN to define the channel and gain to be used in the conversion of the analog input. The logic is similar to the subroutine, RANCHN, supplied by the manufacturer; however, SGLCHN uses VALDT subroutine to determine strain significant data. This routine converts a single channel at a time, although the algorithm could very easily be expanded to include a series of channels.

The mechanics of the software that accomplished this task are based on a four-element vector. They are as follows:

X : the current value of the signal on the designated channel

XLST: the previous value of the signal on the channel designated

SIGN: the sign of the last change of the signal on the channel designated (ØØH means positive slope; Ø1H means negative slope)

FLAG: an indicator showing the status of the last strain-significant event. ØØH means within the threshold, non-zero means outside the threshold.

Once a signal has been determined to be a strain-significant event, it is identified by channel number and stored in RAM. If an event fills the last storage location in RAM, the recording procedure is initiated. The block of 256 bytes containing the data words is transferred, byte-by-byte, to the recorder and written on the magnetic tape. Upon completion of the transfer, the RAM is free to be refilled.

3. HEADING Subroutine

The HEADING subroutine is designed to format the heading information that is to be written on cassette tape. The particular heading information depends on whether the load data were computer generated or obtained from actual aircraft flights. The HEADING subroutine is used in ADCCHN, ADCMXM and MASTER Executive routines. The operator is guided through the subroutine by system prompt output on the (TTY). The first prompt the operator sees is as follows:

"LOAD DATA - ACTUAL OR SIMULATED A/S"

The operator now selects actual (A) or simulated (S) load data. If the operator selects "S", then the system will respond with the following prompts:

"MTS SCALE FACTOR (ENTER 5-DIGITS + SB)"

The MTS SCALE FACTOR refers to the four load ranges ($\pm$ 10,000; $\pm$ 20,000; $\pm$ 50,000; $\pm$ 100,000) on the MTS. The operator enters the appropriate five digits. If 100,000 scale is used, the operator would enter 99999.

"LIMIT LOAD (ENTER 5-DIGITS)"

The LIMIT LOAD refers to the design load derived from stress analysis. The operator enters five digits plus a space bar.

"CROSS SECT AREA (ENTER 3-DIGITS + DP)"

The CROSS SECT AREA refers to the cross-sectional area of the test specimen at the point of investigation.

The operator can enter any combination of three digits and a decimal point, plus a space bar.

"RANDOM NUMBER SEED (ENTER 6 DIGITS)"

The RANDOM NUMBER SEED refers to the seed used in the load sequence randomization technique. The operator enters six digits plus a space bar.

Had the operator declared the load data as actual, then the system would display the following messages:

"JULIAN DATE (ENTER 4 DIGITS + SB)"

The operator enters three digits representing the number of days that have expired since the beginning of the year plus the last digit of the year; example, 1358 (135th day, 1978). The operator types space bar for next prompt.

"AIRCRAFT TYPE (ENTER 4 CHARS + SB)"

The AIRCRAFT TYPE refers to the type of aircraft the data were recorded from; for example, A-7E, F-4J, EA-6B, etc. The operator enters aircraft type plus space bar.

"BUNO (ENTER 6 DIGITS + SB)"

The BUNO refers to the Bureau number of the aircraft. The operator enters the six-digit Bureau number plus a space bar.

"CONFIGURATION (ENTER 1-LETTER A, B, C OR D)"

The CONFIGURATION refers to the external stores loading. The letters A through D are defined by the user. For example, the following definitions might be used:

"A" BASIC AIRCRAFT NO STORES

"B" CONF. "A" PLUS CENTERLINE STORE

"C" CONF. "B" PLUS INBOARD WING STORES

"D" CONF. "C" PLUS OUTBOARD WING STORES.

The operator enters the appropriate letter plus a space bar.

"GROSS WEIGHT (ENTER 5 DIGITS + SB)"

The GROSS WEIGHT refers to the aircraft's appropriate gross weight. The operator enters five digits plus space bar.

"MISSION (ENTER 1-DIGIT 1, 2, 3, 4 or 5)"

The MISSION is user defined. For example, the user may wish to define "1" as Air Combat or "2" as Close Air Support or "3" as Point Intercept, etc. The operator enters the appropriate digits plus a space bar.

The system will not write the above heading information on the cassette tape. Once all the heading information is recorded, the tape is stopped and the system returns to the calling routine.

4. Display File Subroutine (DSFILE)

The DSFILE subroutine is designed to output each record to the teletype in a specified format. It is designed to read each record of a file from a cassette tape and output the record on the TTY in a specified format. The output varies slightly, depending on whether the load data were actual, or computer generated. The operator is guided through the routine by prompts at the console.

We now consider the two different output formats. If the load data were computer generated, then the following output will be displayed:

```
SIMULATED LOAD DATA
MTS SCALE FACTOR    : 20000
LIMIT LOAD          : 30000
CROSS SECT AREA     :  .500
RANDOM NUMBER SEED :000583
STRAIN DATA FOLLOWS:
```

The system will now prompt the operator with the following:

```
HARD COPY STRAIN DATA (Y/N)
```

The operator now has the choice of typing out the converted data or not. If "Y" is depressed, then the converted data will be displayed in blocks of 256 bytes. Each block of data will have eight columns with 16 rows. Sample output follows:

```
HARD COPY STRAIN DATA (Y/N)Y
 E4F8 03B8 7018 02B8 7018 03B8 7018 0048
 7018 0138 7018 0AB8 7018 02B8 7018 0A38
 7018 03B8 7018 03B8 7018 01B8 7018 02B8
 7018 03B8 7018 02B8 7018 03B8 7018 03B8
 7188 03B8 7018 02B8 7018 03B8 7018 03B8
 7018 03B8 7018 02B8 7018 03B8 7018 01B8
 7018 03B8 7018 0C38 7018 02B8 7018 02B8
 7018 03B8 7018 03B8 7018 01B8 7018 03B8
 7018 02B8 7018 03B8 7018 1138 7018 03B8
 7018 02B8 7018 03B8 7018 08B8 7018 03B8
 7018 02B8 7018 02B8 7018 02B8 7018 03B8
 7018 02B8 7018 03B8 7018 003B 7018 03B8
 7018 03B8 7018 0E38 7018 01B8 7018 03B8
 7018 0C38 7018 02B8 7018 03B8 7018 02B8
 7018 02B8 7018 02B8 7018 02B8 7018 0AB8
 7018 3138 7018 0E38 7018 02B8 7018 02
```

If the operator did not want a hard copy of the data, depressing any other key will dump the next 256 byte block a record type 3 data. This process will continue until control senses a record type 4, at which time the system will type "END OF FILE" on the TTY and return the user to the MAIN Executive routine.

Had the load data been from actual aircraft flights, then the operator would display the following information:

ACTUAL LOAD DATA

JULIAN DATE : 1628

AIRCRAFT TYPE : EA-6B

BUNO : 132628

CONFIGURATION : C

GROSS WEIGHT : 52500

MISSION : 1

The remainder of the routine would be exactly as described above. The display subroutine is used in the READ Executive routine.

5. RAMP Subroutine

The RAMP subroutine is designed to output a voltage via one of the DAC's corresponding to a particular load. The driver loads are stored in the random load buffer, which begins at memory location 4400H and extends to memory location 7F00H. The routine compares two successive loads, and increments or decrements the DAC output in steps of 0.00488 volts until the load values are equal. In between each incremental

step the system delays a total of 6 millisec and checks the assigned channel for significant strain data. The 6 millisec delay provides the DAC output with a constant slope at an average rate of one cycle per second.

When the two loads are equal, the system will relieve the next load and repeat the above process until a "1AH" byte, located at the end of the random data, is sensed. This byte signals the end of the data, and the system checks to see if it is to be repeated. The random load data can be repeated up to 16 times, providing approximately ten hours of continuous testing. At the completion of the specified number of runs, the routine prompts the operator "M>" by automatically returning to the MAIN Executive routine.

B. EXECUTIVE ROUTINES

The Executive routines integrate the entire software package by using various combinations of subroutines to perform specific tasks. There are seven Executive routines: MAIN, SORCLD, ADCCEN, ADCMXM, READ, LOAD, and MASTER. The following paragraphs will discuss in detail the function of each Executive routine.

1. MAIN Executive

The MAIN Executive routine is designed to function as an access routine to the other Executive routines and the system 80/10 monitor. It is assumed that the system has been powered up and the punch paper tape containing all the prompt

messages has been read in via paper tape reader. To gain access, the operator types in at the console "G3C40." The system will respond with "M>" followed by "CONTROL G FOR INSTRUCTIONS." Each time the MAIN Executive is accessed, the stack pointer is initialized and the DAC's are set to 0 volts. This last point is very important because, if it becomes necessary to "RESET" the system, both DAC's are driven to minus full scale. This could be disastrous if a test specimen were hooked up. Continuing, should the operator choose to enter CONTROL G at the console, the following prompt would be displayed:

CONTROL A - ENTER ADCCHN EXECUTIVE
CONTROL B - ENTER ADCMXM EXECUTIVE
CONTROL C - ENTER SBC 80/10 MONITOR
CONTROL D - EXIT EXECUTIVE ROUTINE
CONTROL E - ENTER MASTER EXECUTIVE
CONTROL G - INSTRUCTIONS
CONTROL L - ENTER LOAD EXECUTIVE
CONTROL R - ENTER READ EXECUTIVE
CONTROL S - ENTER SURCLD EXECUTIVE
Decimal Point = DP
Space Bar = SB.

The operator now has access to the entire system via the above instructions. At this point the operator may wish to access the system 80/10 monitor to read in a random load paper tape. The operator accesses the monitor system by a "CONTROL C." The system will respond with "." At this time, the

operator would load the punch paper tape containing the random load data into tape reader, switch mode switch to "KT" and depress the "R" key, followed by carriage return (CR). The system will begin to read in the punched paper tape of random load data beginning at memory location 4400. This process of reading in load data need be done only once, since the SORCLD Executive will permanently record load data on a master load tape. After reading the punched paper tape, the tape movement will cease and the system will display "." at the console. The operator should next depress the "RESET" button, located on the front panel of the system 80/10 microcomputer (Fig. 13). (Note: The machine should have hydraulic drive off prior to depressing reset button, since DAC's are driven to minus full scale.)

It is necessary to insert a special byte value (1AH) at F700H at this time to signal the end of load data and prevent the system from possibly interfering with the stack area.

To gain access back to the MAIN Executive, the operator types in at the console "G3C40." The system will respond with "M>" followed by "CONTROL G FOR INSTRUCTIONS."

If the operator has just used a paper tape containing random loads, the next Executive routine selected would be "SORCLD."

2. SORCLD Executive

The SORCLD Executive is designed to make a copy on magnetic tape of the data between memory locations 4400-F700 HEX. Before entering SORCLD, ensure that the tape is rewound,

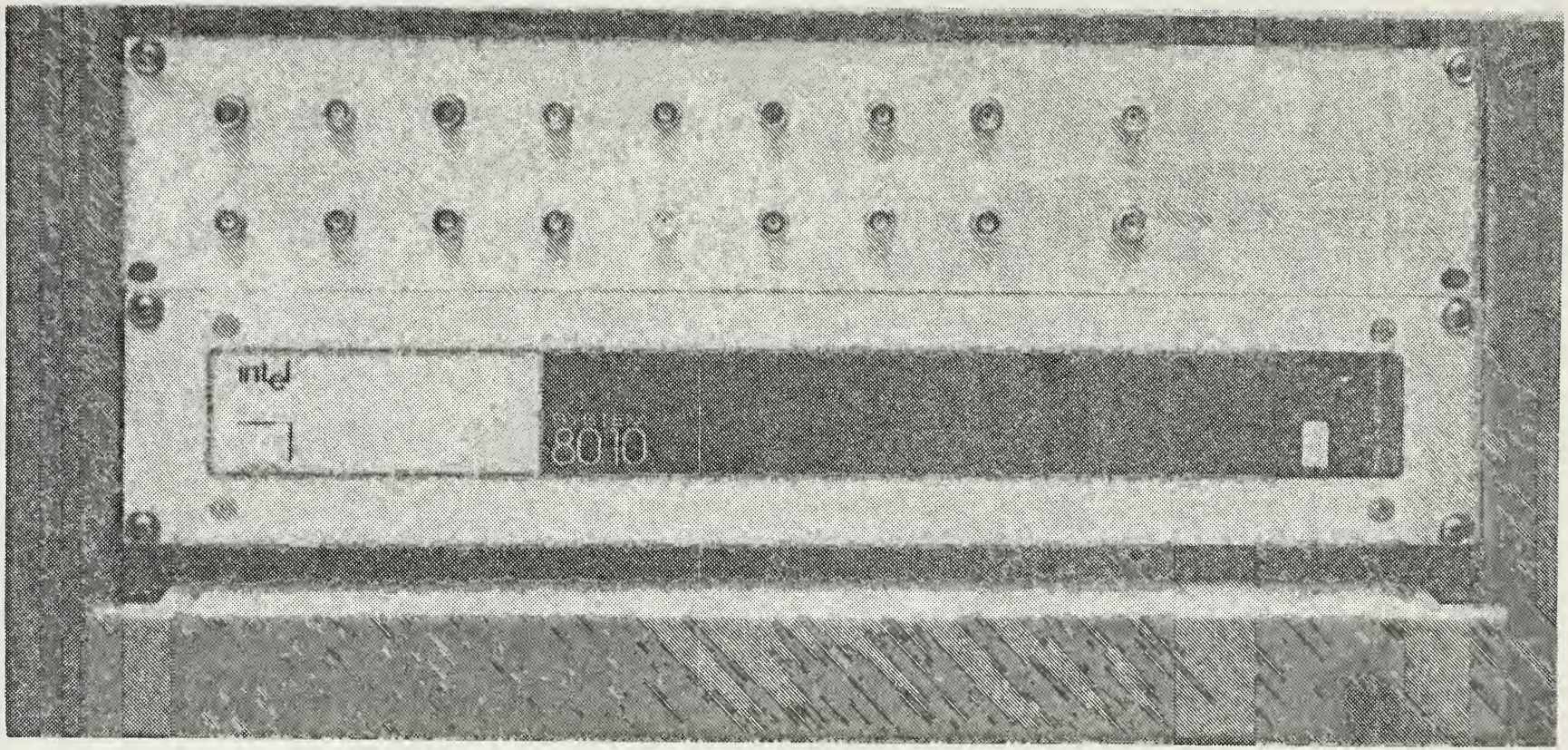

Figure 13. SBC-80/10 Front Panel with ADC and DAC BNC Connectors.

the "Write" head is engaged, and the Write/Read switch is in the "Write" mode. The routine is accessed from the MAIN Executive by holding down the "CONTROL" key and depressing the "S" key.

The cassette tape will begin to record load data located between 4400-7F00H. Upon completion, the system will display the following prompt and automatically return to the MAIN Executive:

LOAD DATA TRANSFER - COMPLETED

RELEASE WRITE HEAD

DEPRESS REWIND BUTTON

REMOVE LOAD DATA CASSETTE

ENGAGE WRITE HEAD

CHECK FOR M>.

"M>"

CONTROL G FOR INSTRUCTIONS

Had the random load data been recorded previously on magnetic tape, the operator could have elected to enter the LOAD Executive.

3. LOAD Executive

The LOAD Executive is designed to read random load data contained on cassette magnetic tape into memory, beginning at location 4400H.

Before entering the LOAD Executive, the operator should insure that the tape is rewound, the "Read head" is engaged, and the Write/Read switch is in the "Read" position. The LOAD Executive is accessed from the MAIN Executive by

CONTROL L command. The tape recorder will begin to read the random load data into memory. At the completion of this cycle, the system will indicate that the load data transfer is completed and will automatically return to the MAIN Executive. The prompt message is that given in the SORCLD Executive.

4. ADCCHN Executive

The ADCCHN Executive is designed to perform ADC on a selected channel at a 28 KHZ rate. The converted data is stored in a 256 byte buffer. When the buffer fills up, the data are dumped onto the cassette tape; thus freeing the buffer for more data. This process continues until the operator elects to terminate by entering a CONTROL D at the console, at which time the partially filled buffer is dumped onto cassette tape, the tape is stopped and the system is returned to the MAIN Executive routine.

The ADCCHN Executive is accessed from the MAIN Executive by entering a CONTROL A at the console. The system will respond with "A>" followed by LOAD DATA - ACTUAL OR SIMULATED (A/S)." Upon selecting "A" or "S" the operator will proceed through bookkeeping prompts as discussed under "Heading" subroutine. After heading information has been entered, the operator will be prompted with the following message, "SELECT CHANNEL NUMBER (0-F HEX)." The operator depresses the appropriate key, and the conversion process will begin. This routine is especially useful when testing and calibrating individual channels.

5. ADCMXM Executive

The ADCMXM Executive is designed to perform ADC on a selected channel. However, the only data that get stored in the data buffer, and eventually on cassette tape, are the maximum or minimum values of the converted data. The subroutine VALDT provides the logic for obtaining maximums and minimums. The significant data bytes are stored in a memory data buffer until the buffer is filled, at which time the buffer is emptied onto cassette tape. This process continues until the operator enters CONTROL D at the console, at which time the partially filled buffer is emptied onto tape, and the tape is stopped. The routine will automatically return the user to the MAIN Executive routine.

6. MASTER Executive

The MASTER Executive is designed to drive the materials testing system (MTS) with a random load, and simultaneously monitor a selected channel for strain-significant events. The random load data used to drive the MTS are located beginning at memory location 4400H. [See Appendix B].

The MASTER Executive is accessed from the MAIN Executive by entering a CONTROL E at the console. The system will respond "E>" prompt, followed by "LOAD DATA - ACTUAL OR SIMULATED (A/S)." Following the heading prompts, the system will prompt the operator with "NUMBER OF RUNS ENTER - 1 DIGIT (0-H HEX)." The operator enters the number of times that the load data sequence is to be repeated, i.e., if the operator wishes to repeat the load sequence four (4) times, then enter a "4."

This capability enables the operator to run the load sequence 16 times, which, with the given number of loads (7552) will provide approximately 10 hours of continuous testing. The next system prompt is "SELECT CHANNEL NUMBER (O-F HEX)." The operator types in the channel number and the system will begin to convert data and writing peaks and valleys on the cassette tape. The system automatically returns to the MAIN Executive following the last cycle through the random load data.

7. READ Executive

The READ Executive is designed to read (or play back) a file created by the Write program. The information is displayed at the console. The operator has the choice of displaying record type 3 data or continuing. To gain access to the READ Executive from the MAIN Executive, the operator types a CONTROL R at the console. Prior to the CONTROL R command, the cassette tape should be rewound, the Read head engaged, and the Write/Read switch set to the Read position. The system will respond with "R," then immediately begin reading tape records. The record 1 or 2 type data are displayed directly. The record 3 data are dumped in 256 byte blocks into RAM, the tape is stopped, and the operator is prompted, "HARD COPY STRAIN DATA (Y/N)." If the operator should elect "Y," then the specific data loaded in the 256 byte RAM buffer are typed out at the console. If the operator enters "N," the next block of data is read from the tape into RAM. This process is continued until the END-OF-FILE is typed at the console and the system automatically returns to the MAIN Executive.

C. PRELIMINARY SYSTEM QUALIFICATION

This section discusses the preliminary qualification tests conducted on the data acquisition system, intended to exercise the device throughout its expected range of operation so that actual performance limitations, if any, may be determined.

1. DC Calibration

Accompanying the SBC-732 analog input/output model was a voltage calibration and scan test software package [Ref.]. This program consists of a three-step sequence which must be performed in the following order: (1) P&A offset adjustment, (2) ADC offset adjustment, and (3) ADC range adjustment. Following the calibration test, the ADC system was verified by applying standard precision DC voltage. The ±10 volt range was chosen to coincide with the MTS machine operating range. The test also verified the subroutine associated with the ADCCHN Executive routine. For instance, 2 volts from a standard precision supply were read into the system and an interrogation of the memory showed that the system stored 99ADH, which corresponds to 2 volts.

2. Incremental DC Volt Step Test

This test was designed to verify the theoretical incremental voltage of each bit used in the conversion process. Using 12 bits for conversion, there are (16 x 16 x 16 = 4096) incremental steps possible. This means, using the ±10 volt range, that each incremental step is equivalent to 0.00488 volts. A standard precision voltage source was applied in steps of 5 millivolts. Changes in the bit pattern verified the theoretical predictions.

3. Sinusoidal Signal Reconstruction

This test was designed to evaluate the system's ability to accurately acquire, store, and reconstruct a sinusoidal signal with a known amplitude. The analog input signal was converted and recorded on cassette tape using the ADCCHN Executive routine. Recorded data were read into memory location beginning at 4400H and outputted via system DAC's to a strip chart recorder. The reconstruction signal was compared to the original signal and produced deviations of less than 1%.

4. Peak and Troughs Test

This test was designed to check the accuracy of the system to record only the peaks and valleys of a known sinusoidal signal. The amplitude of the sign wave was calibrated and measured. The peaks and valleys were reached on magnetic cassette tape using the ADCMXM Executive routine. The converted value from the tape was broken down into its voltage representation and compared with actual input voltage. For example, the converted value of FEEØH is equivalent to (256 x 15 + 16 x 14 + 14 = 4078) steps, which is the same as 9.9121 volts. This voltage was then compared with strip chart value of 9.94 volts, which was well within the accuracies of the instrumentation.

5. Driver Test

This test was designed to exercise the entire system using the random loads generated on the IBM-360 and to drive the MTS machine using the RAMP subroutine. Some initial concern was experienced with the speed of the drive signal and

the response of the MTS machine when the output signals were attenuated by 10%. The RAMP rate was set to an average of .7 cycles per second; the first list results indicated that the driver voltage was being attenuated at both the peaks and troughs by about 0.5 volts on a 0 to 10 volt signal.

Further investigation revealed that because of the incremental step technique used to create the driver signal, it was necessary to hold the RAMP signal longer at the DAC output port before proceeding to the next incremental step. Test results indicated that it was necessary to hold the signal at the DAC output about 8 millisec before proceeding to the next step. The resulting acceptable driver rate was approximately .6 cycles per sec. This test verified the subroutines used in the MASTER Executive routine.

V. CONCLUSIONS AND RECOMMENDATIONS

Based upon the performance tests discussed in the preceding section, it is concluded that the system is executing its several modes of operation accurately and reliably. Furthermore, during the course of the software interface development, modifications were constantly made with the intent of improving the performance and operator interface. The modularity of the software package makes it versatile and easily adapted to future changing needs.

APPENDIX A

GLOSSARY

1. ASCII: American Standard Code for Information Interchange. This is a seven-bit-plus-parity code established by the American National Standards Institute to achieve compatibility between data services. Also called USASCII.
2. Assembly: A listing which contains both source code and machine code.
3. BIT: BInary digiT. A single unit of information in a binary word.
4. Buffer: A group of memory locations used to store specific data (input data, constants, output data, etc.).
5. Byte: An eight-BIT word which is processed as a single quantity.
6. CPU: Central Processing Unit. The area of the microprocessor which computes and sequences all logic and arithmetic functions.
7. CRT: Cathode Ray Tube - A television-like picture tube used in visual display terminals.
8. D/A: The inverse of the A/D process.
9. EPROM: Erasable/programmable read only memory.
10. I/O: Input/output.

11. K: A suffix which indicates a group of 1024 (2^{10}) items as in "4K of memory" meaning 4096 memory locations.

12. MUX: A multiplexing device.

13. Nibble: The upper or lower four BITs in one byte.

14. RAM: Random access memory. Volatile memory used for variable storage and data manipulation.

15. Register: A storage location located in the CPU.

16. ROM: Read only memory, non-volatile.

17. Sample and Hold: A device for sampling the amplitude of a signal at a given time and holding that amplitude.

18. Software: The program which resides in the U-P's memory.

19. Source code: The program written by the user.

APPENDIX B

MEMORY MAP

Beginning Address	Description	Ending Address
0000H	EPROM #1 SBC-80/10 Monitor	Ø3FFH
0400H	EPROM #2 SBC-80/10 Monitor Subroutine	Ø7FFH
0800H	EPROM #3 Subroutines	OBFFH
0600H	EPROM #4 Subroutines + Executive Routines	OFFFH
1000H	11K Uncommitted Memory	3BFFH
3C00H	RAM Reserved for Monitor	363FH
3C40H	Start Software Program	
	Buffer #3	
3C80H	Messages/Variables	42FFH
4300H	256 Byte Data Buffer	43FFH
4400H	15,104 Byte Random Load Buffer	7F00H
7F00H	Stack Buffer	7FFFH
F700H	Memory Based Address for SBC-732 Board	F77FH

```
                        ;*****************************************************
                        ;
                        ;  NEWTWO VERSION 1.0  JUNE 1978
                        ;  THE FOLLOWING ASSEMBLY LANGUAGE PROGRAM
                        ;  INTEGRATES RANDOM LOAD TESTING ON
                        ;  THE MATERIALS TESTING SYSTEM MACHINE
                        ;
                        ;*****************************************************

3C40                        ORG 3C40H
3C40 31FF7F                 LXI  SP,7FFFH
3C43 C3860F                 JMP  MAIN ;

                        ;*****************************************************
                        ;
                        ;  SYSTEM EQUATES
                        ;
                        ;*****************************************************
                        ;  8255 #2 PORT ADDRESSES
                        ;*****************************************************
00E8 =                  PORTA   EQU  0E8H ;8255 #2 PORTA
00E9 =                  PORTB   EQU  0E9H ;8255 #2 PORTB
00EA =                  PORTC   EQU  0EAH ;8255 #2 PORTC
00EB =                  CONTROL EQU  0EBH ;8255 #2 CONTROL WORD
                        ;*****************************************************
                        ;  CONSOLE(TTY)
                        ;*****************************************************
00EC =                  CRTD    EQU  0ECH ;CONSOLE DATA PORT
00ED =                  CRTS    EQU  0EDH ;CONSOLE STATUS PORT
0002 =                  RBR     EQU  2    ;RECIEVER READY BIT
0001 =                  TRDY    EQU  1    ;TRANSMITTER READY BIT
0800 =                  PRMBRT  EQU  0800H      ;PROM 3 AND 4 BRANCH TABLE
```

```
0001 =          CNTRLA  EQU  01H  ;CONTROL A
0002 =          CNTRLB  EQU  02H  ;CONTROL B
0003 =          CNTRLC  EQU  03H  ;CONTROL C
0004 =          CNTRLD  EQU  04H  ;CONTROL D
0005 =          CNTRLE  EQU  05H  ;CONTROL E
0007 =          CNTRLG  EQU  07H  ;CONTROL G
000C =          CNTRLL  EQU  0CH  ;CONTROL L
0012 =          CNTRLR  EQU  12H  ;CONTROL R
0013 =          CNTRLS  EQU  13H  ;CONTROL S
                ;*****************************************************
                ;
                ;  SBC 732 BOARD
                ;
                ;*****************************************************

F7CC =          BASE    EQU  0F700H     ;BASE ADDR OF INTERFACE
F7CC =          SCA     EQU  BASE+0     ;A/D CMD REGISTER
F700 =          STAT    EQU  BASE+0     ;A/D STATUS REGISTER
F7C1 =          FCR     EQU  BASE+1     ;GAIN,MUX ADDR REGISTER
F702 =          LCR     EQU  BASE+2     ;LAST CHAN REG (WRITE ONLY)
F7C3 =          CLRI    EQU  BASE+3     ;CLEAR INTERRUPTS
F704 =          ADDATA  EQU  BASE+4     ;A/D DATA REGISTER
F708 =          DAC0    EQU  BASE+8     ;D/A OUTPUT CHANNEL ZERO
F70A =          DAC1    EQU  BASE+10    ;D/A OUTPUT CHANNEL ONE
8000 =          ADCHS   EQU  8000H      ;ADC HALF SCALE ( 0 VOLTS )
0001 =          GC EQU  1       ;CONV-ENABLE IN CMD REG.
0002 =          AIE     EQU  2      ;AUTO-INCREMENT ENABLE BIT.
0004 =          EIE     EQU  4      ;EXT-TRIG ENABLE BIT
0008 =          BUSY    EQU  8      ;MULTI-PROCESSOR BUSY BIT
001C =          ECSIE   EQU  10H    ;ECS INTERRUPT BIT.
0020 =          EOCIE   EQU  20H    ;EOC INTERRUPT ENABLE
004C =          ECS     EQU  40H    ;EOS STATUS
0080 =          EOC     EQU  80H    ;EOC STATUS
0000 =          CHAN0   EQU  0      ;CHANNEL ZERO
0000 =          X1 EQU  0       ;GAIN = 1.
                ;*****************************************************
                ;
                ;  SBC 80/10 MONITOR
                ;
                ;*****************************************************
0008 =          MONITOR EQU  0008H      ;ENTER MONITOR
03FD =          CI EQU  03FDH       ;CONSOLE INPUT
03FA =          CO EQU  03FAH       ;CONSOLE OUTPUT
000D =          CR EQU  0DH  ;CARRIAGE RETURN
000A =          LF EQU  0AH  ;LINE FEED
0481 =          DELAY   EQU  0481H      ;PROGRAMMMED DELAY OF 1 MILLISECOND
02AC =          HILO    EQU  02ACH      ;COMPARES TWO 16 BIT INTEGERS
                             ;IN HL AND DE REGISTERS
```

```
02C2 =          NMOUT   EQU  02C2H      ;CONV 8-BITS TO TWO ASCII CHARS
                                         ;TO THE CORRESPONDING ASCII CHAR
                                         ;( 0-9,A-F )
02D5 =          PRVAL   EQU  02D5H      ;CONVERTS A NUMBER IN THE RANGE
                                         ;( 0-F HEX )
01DF =          CNVBN   EQU  01DFH      ;CONVERT ASCII TO BINARY

                ;*****************************************************
                ;
                ;  VARIABLES
                ;
                ;*****************************************************
3C46            REPEAT: DS   1      ;( 1-A )
3C47            GAIN:   DS   1      ;(0,1,2, OR 3)
3C48            FSTCHN: DS   1      ;(0-31)
3C49            LSTCHN: DS   1      ;(0-31)
3C4A            KCUNTR: DS   1      ;DATA BUFFER COUNTER
3C4B            DATPTR: DS   2      ;POINTER TO DATA FIELD
3C4D            BFPTR1: DS   2      ;POINTER TO RANDOM LOAD DATA


3C4F            X: DS   2      ;PRESENT CONVERTOR DATA
3C51            XLST:   DS   2      ;LAST VALID CONVERTOR DATA
3C53            SIGN:   DS   1      ;SIGN ( 00 - OFF )
3C54            FLAG:   DS   1      ;FLAG ( 00 - OFF )


                ;*****************************************************
                ;
                ;  MESSAGES
                ;
                ;*****************************************************

3C80                    ORG  3C80H

3C8C 4143545541MSG1:    DB   'ACTUAL LOAD DATA ',CR,LF,LF
3C94 4A554C4941         DB   'JULIAN DATE     :'
3CA4 CCCCCCCC   DATE:   DB   0,0,0,0
3CA8 0D0A               DB   CR,LF
```

```
3CAA 4149524352MSG1A:   DB    'AIRCRAFT TYPE  :'
3CBA 00000000  TYPE:    DB    0,0,0,0
3CBE 0D0A               DB    CR,LF
3CC0 42554E4F20MSG1B:   DB    'BUNC           :'
3CD0 0C00000000SIDENO:  DB    0,0,0,0,0,0
3CD6 0D0A               DB    CR,LF
3CD8 434F4E4649MSG1C:   DB    'CONFIGURATION  :'
3CE8 00          CONFIG:DB    0
3CE9 0D0A               DB    CR,LF
3CEB 47524F5353MSG1D:   DB    'GROSS WEIGHT   :'
3CFB 0C00000000GSWT:    DB    0,0,0,0,0
3DCC 0D0A               DB    CR,LF
3DC2 4D49535349MSG1E:   DB    'MISSION        :'
3D12 0C          MSN:   DB    0
3D13 0D0A               DB    CR,LF
3D15 53494D554CMSG2:    DB    'SIMULATED LOAD DATA W'
3D2A 0D0A0A             DB    CR,LF,LF
3D2D 4D545320C53        DB    'MTS SCALE FACTOR   :'
3D41 0C00000000SCALE:   DB    0,0,0,0,0
3D46 0D0A               DB    CR,LF
3D48 4C494D4954         DB    'LIMIT LOAD         :'
3D5C 0C00000000LIMIT:   DB    0,0,0,0,0
3D61 0D0A               DB    CR,LF
3D63 43524F5353         DB    'CROSS SECT AREA    :'
3D77 0C000000  AREA:    DB    0,0,0,0
3D78 0D0A               DB    CR,LF
3D7C 52414E444F         DB    'RANDOM NUMBER SEED :'
3D91 0C00000000SEED:    DB    0,0,0,0,0,0
3D97 0D0A               DB    CR,LF
3D99 5354524149         DB    'STRAIN DATA FOLLOWS:'
3DAD 0D0A0A             DB    CR,LF,LF
3DB0 454E44204FMSG3:    DB    'END OF FILE',CR,LF


3DBC 0D0A434845MSG4:    DB    CR,LF,'CHECKSUM ERROR',CR,LF,'$'
3DD0 0D0A454E44MSG5:    DB    CR,LF,'END OF FILE',CR,LF,'$'
3DEC 0D0A484152MSG5A:   DB    CR,LF,'HARD COPY STRAIN DATA (Y/N)','$'
3DFE 4A554C4941MSG6:    DB    'JULIAN DATE ( ENTER 4 DIGITS + SB )',CR,LF,'$'
3E24 0D0A414952MSG7:    DB    CR,LF,'AIRCRAFT TYPE ( ENTER 4 CHARS + SB  )'
3E4B 0D0A24             DB    CR,LF,'$'
3E4E 0D0A42554EMSG8:    DB    CR,LF,'BUNC ( ENTER 6 DIGITS + SB )',CR,LF,'$'
3E6F 0D0A434F4EMSG9:    DB    CR,LF,'CONFIGURATION (ENTER 1-LETTER-A,B,C,OR'
3E97 44202B2053         DB    'D + SB)',CR,LF,'$'
3EA1 0D0A47524FMSGA:    DB    CR,LF,'GROSS WEIGHT ( ENTER 5 DIGITS + SB  )'
3EC8 0D0A24             DB    CR,LF,'$'
3ECB 0D0A4D4953MSGB:    DB    CR,LF,'MISSION ( ENTER 1-DIGIT--1,2,3,4,CR 5'
3EF2 2B20534229         DB    '+ SB)',CR,LF,'$'
```

```
3EFA 0D0A0A0A  MSGC:    DB    CR,LF,LF,LF
3EFE 4C4F414420    DB    'LOAD DATA TRANSFER-COMPLETED',CR,LF,LF
3F1D 52454C4541    DB    'RELEASE WRITE HEAD',CR,LF
3F31 4445505245    DB    'DEPRESS REWIND BUTTON',CR,LF
3F48 52454D4F56    DB    'REMOVE LOAD DATA CASSETTE',CR,LF
3F63 494E534552    DB    'INSERT CLEAN CASSETTE',CR,LF
3F7A 454E474147    DB    'ENGAGE WRITE HEAD',CR,LF
3F8D 434845434B    DB    'CHECK FOR M> PROMPT ',CR,LF,'$'
3FA4 0D0A4D5453MSGD:    DB    CR,LF,'MTS SCALE FACTOR ( ENTER 5-DIGITS + SB'
3FCC 290D0A24      DB    ')',CR,LF,'$'
3FD0 0D0A4C494DMSGD1:   DB    CR,LF,'LIMIT LOAD ( ENTER 5-DIGITS + SB )'
3FF4 0D0A24        DB    CR,LF,'$'
3FF7 0D0A43524FMSGD2:   DB    CR,LF,'CROSS SECT AREA ( ENTER 3-DIGITS +DP +'
401F 534220290D    DB      'SB )',CR,LF,'$'
4026 0D0A52414EMSGD3:   DB    CR,LF,'RANDOM NUMBER SEED ( ENTER 6-DIGITS +'
404D 5342202900D   DB    'SB )',CR,LF,'$'
4054 0D0A204C4FMSGE:    DB    CR,LF,' LOAD DATA-ACTUAL OR SIMULATED (A/S)'
407A 0D0A0A24      DB    CR,LF,LF,'$'
407E 0D0A53454CMSGF:    DB    CR,LF,'SELECT CHANNEL NUMBER ( 0-F HEX )',CR,LF
40A3 24            DB    '$'
40A4 0D0A4D3E0DMSGG:    DB    CR,LF,'M>',CR,LF,'$'
40AB 0D0A533E0DMSGG1:   DB    CR,LF,'S>',CR,LF,'$'
40B2 0D0A413E0DMSGG2:   DB    CR,LF,'A>',CR,LF,'$'
40B9 0D0A423E0DMSGG3:   DB    CR,LF,'B>',CR,LF,'$'
40C0 0D0A523E0DMSGG4:   DB    CR,LF,'R>',CR,LF,'$'
40C7 0D0A453E0DMSGG5:   DB    CR,LF,'E>',CR,LF,'$'
40CE 0D0A4C3E0DMSGG6:   DB    CR,LF,'L>',CR,LF,'$'
40D5 0D0A434F4EMSGH:    DB    CR,LF,'CONTROL G FOR INSTRUCTIONS',CR,LF,'$'
40F4 2C20434F4EMSGH1:   DB    '  CONTROL A - ENTER ANCCHN EXECUTIVE',CR,LF
411A 2020434F4E    DB    '  CONTROL B - ENTER ADCMXM EXECUTIVE',CR,LF
414C 2C20434F4E    DB    '  CONTROL C - ENTER SBC 80/10 MONITOR',CR,LF
4167 2020434F4E    DB    '  CONTROL D - EXIT EXECUTIVE ROUTINE',CR,LF
418D 2C20434F4E    DB    '  CONTROL E - ENTER MASTER EXECUTIVE',CR,LF
41B3 2C20434F4E    DB    '  CONTROL G - INSTRUCTIONS',CR,LF
41CF 2020434F4E    DB    '  CONTROL L - ENTER LOAD EXECUTIVE',CR,LF
41F3 2C20434F4E    DB    '  CONTROL R - ENTER READ EXECUTIVE',CR,LF
4217 2C20434F4E    DB    '  CONTROL S - ENTER SORCLC EXECUTIVE',CR,LF
423D 2C20444543    DB    '  DECIMAL POINT = DP',CR,LF
4253 2C20535041    DB    '  SPACE BAR     = SB',CR,LF,'$'
426A 0D0A4E554DMSGI:    DB    CR,LF,'NUMBER OF RUNS ENTER-1 DIGIT ( 0-A HEX'
4292 290D0A24      DB    ')',CR,LF,'$'



4296 0C90          UPPER:  DB    00H,90H
4298 0C70          LOWER:  DB    00H,70H
429A 0C            TYPDTA: DB    00H  ;ACTUAL=A OR SIMULATED=S TYPE DATA
```

```
                ;*****************************************************
                ;
                ;  BUFFERS
                ;
                ;*****************************************************

4300               ORG  4300H
4300            BUFR:   DS   256  ;DATA BUFFER

44CC               ORG  4400H
4400            BUFR1:  DS   15104     ;RANDOM LOAD DATA BUFFER
                        ;4400H TO 7F00H

                ;*****************************************************

0561               ORG  0561H

                RDLOAD:
0561 C600          MVI  B,00H     ;INITIALIZE CHECK SUM
                RDL1:
0563 CDC00A        CALL RDCHAR    ;READ FIRST BYTE
0566 FE00          CPI  00H  ;
0568 C26305        JNZ  RDL1 ;FIND FIRST NON ZERO CHAR

                RDL2:
056B CDC00A        CALL RDCHAR    ;
056E FE00          CPI  00H  ;
0570 CA6B05        JZ   RDL2 ;

0573 FE01          CPI  01H  ;
0575 CA8905        JZ   RDL5

0578 FE02          CPI  02H  ;
057A CA8905        JZ   RDL5

057D FE03          CPI  03H  ;
057F CA9005        JZ   RDL3 ;

0582 FE04          CPI  04H  ;
0584 CC7A0A        CZ   STOP ;STOP TAPE
```

```
0587 21FA3E        LXI  H,MSGC    ;
058A CD7F08        CALL MSG  ;
058D C3860F        JMP  MAIN ;


                   RDL3:
0590 57            MOV  D,A  ;STORE FILE TYPE
0591 CDC00A        CALL RDCHAR    ;GET FILE LENGTH
0594 4F            MOV  C,A  ;STORE FILE LENGTH
0595 F5            PUSH PSW  ;

                   RDL4:
0596 CDC00A        CALL RDCHAR    ;GET NEXT DATA BYTE
0599 77            MOV  M,A  ;
059A 80            ADD  B    ;
059B 47            MOV  B,A  ;
059C 23            INX  H    ;
059D 0D            DCR  C    ;
059E C29605        JNZ  RDL4 ;
05A1 CDC00A        CALL RDCHAR    ;GET LAST CHAR
05A4 77            MOV  M,A  ;
05A5 80            ADD  B    ;UPDATE CHECKSUM
05A6 47            MOV  B,A  ;
05A7 F1            POP  PSW  ;
05A8 4F            MOV  C,A  ;

05A9 CDC00A        CALL RDCHAR    ;
05AC A8            XRA  B    ;

05AD E5            PUSH H    ;
05AE 21BD3D        LXI  H,MSG4    ;
05B1 C47F08        CNZ  MSG  ;
05B4 E1            POP  H    ;

05B5 23            INX  H    ;

05B6 C36105        JMP  RDLOAD    ;


                   RDL5:
05B9 CDC00A        CALL RDCHAR    ;
05BC 4F            MOV  C,A  ;STORE RECORD LENGTH

                   RDL6:
05BD CDC00A        CALL RDCHAR    ;
05C0 0D            DCR  C    ;
05C1 C2BD05        JNZ  RDL6 ;
05C4 C36105        JMP  RDLOAD    ;
```

```
                        ;*****************************************************
                        ;
                        ;
                        ;   SUBROUTINES
                        ;
                        ;
                        ;*****************************************************
                        ;   PROM 3 AND 4 BRANCH TABLE
0800                        ORG   PRMBRT

08CC C36608                 JMP   CONIN     ;
0803 C37208                 JMP   CONOUT    ;
0806 C37F08                 JMP   MSG   ;
0809 C38A08                 JMP   GAP   ;
080C C39608                 JMP   HEADNG    ;
08CF C3A609                 JMP   OXICE     ;
0812 C3C709                 JMP   WRCHAR    ;
0815 C3E509                 JMP   WRREC     ;
0818 C3160A                 JMP   RANCHN    ;
081B C3300A                 JMP   LDDATA    ;
081E C3550A                 JMP   LDRNDT    ;
0821 C37A0A                 JMP   STOP  ;
0824 C37F0A                 JMP   DUMP  ;
0827 C3910A                 JMP   DELAY1    ;
082A C3A50A                 JMP   CLROX     ;
082D C3C00A                 JMP   RDCHAR    ;
0830 C3E30A                 JMP   RDREC     ;
0833 C3180B                 JMP   OSFILE    ;
0836 C39C0B                 JMP   CROUT     ;
0839 C3A70B                 JMP   GOUGE     ;
083C C3E00B                 JMP   DACZRO    ;

                        ;   PROM #4

083F C3000C                 JMP   SGLCHN    ;
0842 C3220C                 JMP   VALDT     ;
0845 C3BF0C                 JMP   STORE     ;
0848 C3E00C                 JMP   RAMP  ;
084B C3610D                 JMP   WDAC0     ;
084E C3650D                 JMP   WDAC1     ;
0851 C3690D                 JMP   SORCLD    ;
0854 C3B30D                 JMP   ADCCHN    ;
0857 C3320E                 JMP   ADCMXM    ;
```

```
085A C3B10E        JMP   READ ;
085D C3D30E        JMP   LOAD ;
086C C3E20E        JMP   MASTER     ;
0863 C3860F        JMP   MAIN ;


                   ;
                   ;
                   ;  FUNCTION:CONIN
                   ;  INPUTS   :CHARACTER FROM CONSOLE
                   ;  OUTPUTS  :A- CHARACTER FROM CONSOLE
                   ;  CALLS    :CI
                   ;  DESTROYS:
                   ;  DESCRITION: AWAITS UNTILL CHARACTER HAS BEEN ENTERED AT THE
                   ;              CONSOLE AND THEN RETURNS CHARACTERVIA THE A
                   ;              REGISTERTC THE CALLING RCUTINE.
                   ;
                   ;
                   CCNIN:
0866 C5                PUSH B
0867 D5                PUSH D
0868 E5                PUSH H
0869 CDFDC3            CALL CI
086C E67F              ANI  7FH   ;PARITY MASK
086E E1                POP  H
086F D1                POP  D
087C C1                POP  B
0871 C9                RET        ;EXIT CONIN
                   ;
                   ;
                   ;  FUNCTION: CONOUT
                   ;  INPUTS   :C- CHARACTER TO OUTPUT TO CONSOLE
                   ;  OUTPUTS  :C- CHARACTER TO CONSOLE
                   ;  CALLS    :CO
                   ;  DESTROYS:
                   ;  DESCRIPTION: WAITS FOR INPUT CHARACTER AND THEN
                   ;               SENDS CHARACTER TC CONSOLE
                   ;
                   ;
                   CCNOUT:
0872 F5                PUSH PSW
0873 C5                PUSH B
0874 D5                PUSH D
0875 E5                PUSH H
0876 4F                MOV  C,A
```

```
0877 CDFA03          CALL  CO
087A E1              POP   H
087B D1              POP   D
087C C1              POP   B
087D F1              POP   PSW
087E C9              RET          ;EXIT CONOUT



                   MSG:
087F 7E              MOV   A,M    ;CHAR TO ACCM
0880 FE24            CPI   '$'    ;CHECK FOR MESSAGE DELIMITER
0882 C8              RZ

0883 CD7208          CALL  CONOUT     ;OUTPUT CHAR TO CONSOLE
0886 23              INX   H      ;INCREMENT MESSAGE POINTER
0887 C37F08          JMP   MSG    ;GET NEXT CHAR


                   ;
                   ;
                   ;
                   GAP:
088A F5              PUSH  PSW    ;SAVE ACCM FLAGS
088B 3E20            MVI   A,20H      ;00100000
088D D3EA            OUT   PORTC      ;LWD/FWD BLANK TAPE
088F 3EFF            MVI   A,0FFH     ;256MSEC DELAY
0891 CD910A          CALL  DELAY1     ;
0894 F1              POP   PSW    ;RESTORE ACCM AND FLAGS
0895 C9              RET          ;EXIT GAP
                   ;
                   ;
                   ;
                   ;  FUNCTION: HEADNG
                   ;  INPUT   : A- CHARACTERS FROM CONSOLE
                   ;  OUTPUT  :
                   ;  CALLS   : CONIN CONOUT
                   ;  DESTROYS:
                   ;  DESCRIPTION: INPUTS JULIAN DATE, AIRCRAFT TYPE, EUNO,
                   ;       CONFIGURATION, GROSS WEIGHT,AND MISSION.
                   ;
                   ;
                   ;
                   HEADNG:
0896 215440          LXI   H,MSGE     ;DISPLAY SOURCE OF LOAD DATA
0899 CD7F08          CALL  MSG    ;
```

```
089C 219A42        LXI  H,TYPDTA  ;POINT TO TYPE DATA BUFFER
089F CD6608        CALL CONIN     ;
08A2 CD7208        CALL CONOUT    ;
08A5 329A42        STA  TYPDTA    ;STORE TYPE DATA
08A8 FE41          CPI  'A'  ;A-ACTUAL  S-SIMULATED
08AA C24409        JNZ  REC7 ;IF SIMULATED JMP

08AD CD9C0B        CALL CROUT     ;OUTPUT CR LF
08B0 21FE3D        LXI  H,MSG6    ;DISPLAY JULIAN DATE
08B3 CD7F08        CALL MSG  ;
08B6 21A43C        LXI  H,DATE    ;POINT TO JULIAN DATE ADDR
08B9 CD6608    JDATE:  CALL CONIN     ;INPUT CHAR FRM CONSOLE
08BC CD7208        CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
08BF FE20          CPI  ' '  ;IF SPACE BAR JMP A/C TYPE
08C1 CAC908        JZ   REC2 ;
08C4 77            MOV  M,A  ;STORE JULIAN DATE
08C5 23            INX  H    ;INCREMENT ADDRESS
08C6 C3B908        JMP  JDATE     ;GET NEXT DIGIT

08C9 21243E    REC2:   LXI  H,MSG7    ;DISPLAY AIRCRAFT TYPE
08CC CD7F08        CALL MSG  ;
08CF 21BA3C        LXI  H,TYPE    ;POINT TO A/C TYPE ADDR
08D2 CD6608    ACTYPE: CALL CONIN     ;INPUT CHAR FM CONSOLE
08D5 CD7208        CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
08D8 FE20          CPI  ' '  ;IF SPACE BAR JMP SIDENO
08DA CAE208        JZ   REC3 ;
08DD 77            MOV  M,A  ;STORE TYPE
08DE 23            INX  H    ;INCREMENT ADDRESS
08DF C3D208        JMP  ACTYPE    ;GET NEXT CHAR

08E2 214E3E    REC3:   LXI  H,MSG8    ;DISPLAY BUNC
08E5 CD7F08        CALL MSG  ;
08E8 21D03C        LXI  H,SIDENO  ;POINT TO SIDE NUMBER ADDR
08EB CD6608    SDNUM:  CALL CONIN     ;INPUT CHAR FM CONSOLE
08EE CD7208        CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
08F1 FE20          CPI  ' '  ;IF SPACE BAR JMP CONFIGURATION
08F3 CAFB08        JZ   REC4 ;
08F6 77            MOV  M,A  ;STORE SIDE NUMBER
08F7 23            INX  H    ;INCREMENT ADDR
08F8 C3EB08        JMP  SDNUM     ;GET NEXT CHAR

08FB 216F3E    REC4:   LXI  H,MSG9    ;DISPLAY CONFIGURATION
08FE CD7F08        CALL MSG  ;
0901 21E83C        LXI  H,CONFIG  ;POINT TO CONFIG ADDR
0904 CD6608    CNFIG:  CALL CONIN     ;INPUT CHAR FM CONSOLE
0907 CD7208        CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
090A FE20          CPI  ' '  ;IF SPACE BAR JMP GROSS WEIGHT
090C CA1409        JZ   REC5 ;
```

```
090F 77          MOV  M,A  ;STORE CONFIG.
0910 23          INX  H    ;INCREMENT ADDR
0911 C30409      JMP  CNFIG     ;GET NEXT CHAR

0914 21A13E  REC5:   LXI  H,MSGA    ;DISPLAY GROSS WEIGHT
0917 CD7F08      CALL MSG  ;
091A 21FB3C      LXI  H,GSWT    ;POINT TO GROSS WEIGHT ADDR
091D CD6608  GWT:    CALL CONIN     ;INPUT CHAR FM CONSOLE
0920 CD7208      CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
0923 FE20        CPI  ' '  ;IF SPACE BAR JMP MISSION
0925 CA2D09      JZ   REC6 ;
0928 77          MOV  M,A  ;STORE GROSS WEIGHT
0929 23          INX  H    ;INCREMENT ADDR
092A C31D09      JMP  GWT  ;GET NEXT CHAR

092D 21CB3E  REC6:   LXI  H,MSGB    ;DISPLAY MISSION
0930 CD7F08      CALL MSG  ;
0933 21123D      LXI  H,MSN     ;POINT TO MISSION ADDR
0936 CD6608  MSSN:   CALL CONIN     ;INPUT CHAR FM CONSOLE
0939 CD7208      CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
093C FE20        CPI  ' '  ;RETURN IF SPACE BAR
093E C8          RZ        ;
093F 77          MOV  M,A  ;STORE MISSION
0940 23          INX  H    ;INCREMENT ADDR
0941 C33609      JMP  MSSN ;GET NEXT CHAR

0944 21A43F  REC7:   LXI  H,MSGD    ;DISPLAY SCALE FACTOR
0947 CD7F08      CALL MSG  ;
094A 21413D      LXI  H,SCALE   ;POINT TO SCALE ADDRESS
094D CD6608  SCLE:   CALL CONIN     ;INPUT CHAR FM CONSOLE
0950 CD7208      CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
0953 FE20        CPI  ' '  ;IF SPACE BAR JMP LIMIT
0955 CA5D09      JZ   REC8 ;
0958 77          MOV  M,A  ;STORE SCALE FACTOR
0959 23          INX  H    ;INCREMENT ADDR
095A C34D09      JMP  SCLE ;GET NEXT CHAR

095D 21D03F  REC8:   LXI  H,MSGD1   ;DISPLAY LIMIT LOAD
0960 CD7F08      CALL MSG  ;
0963 215C3D      LXI  H,LIMIT   ;POINT TO LIMIT ADDR
0966 CD6608  LMTLD:  CALL CONIN     ;INPUT CHAR FM CONSOLE
0969 CD7208      CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
096C FE20        CPI  ' '  ;IF SPACE BAR JMP AREA
096E CA7609      JZ   REC9 ;
0971 77          MOV  M,A  ;STORE LIMIT LOAD
0972 23          INX  H    ;INCREMENT ADDR
0973 C36609      JMP  LMTLD     ;GET NEXT CHAR
```

```
0976 21F73F     REC9:   LXI  H,MSGD2   ;DISPLAY CROSS SECT AREA
0979 CD7F08        CALL MSG  ;
097C 21773D        LXI  H,AREA    ;POINT TO AREA ADDR
097F CD6608     AR:     CALL CONIN     ;INPUT CHAR FM CONSOLE
0982 CD7208        CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
0985 FE20          CPI  ' '  ;IF SPACE BAR JMP SEED
0987 CA8F09        JZ   RECA ;
098A 77            MOV  M,A  ;STORE CROSS SECT AREA
098B 23            INX  H    ;INCREMENT ADDR
098C C37F09        JMP  AR   ;GET NEXT CHAR

098F 212640     RECA:   LXI  H,MSGD3   ;DISPLAY RANDOM NUMBER SEED
0992 CD7F08        CALL MSG  ;
0995 21913D        LXI  H,SEED    ;POINT TO SEED ADDR
0998 CD6608     SD:     CALL CONIN     ;INPUT CHAR FM CONSOLE
099B CD7208        CALL CONOUT    ;OUTPUT CHAR TO CONSOLE
099E FE20          CPI  ' '  ;RETURN IF SPACE BAR
09A0 C8            RZ        ;
09A1 77            MOV  M,A  ;STORE RANDOM NUMBER SEED
09A2 23            INX  H    ;INCREMENT ADDR
09A3 C39809        JMP  SD   ;GET NEXT CHAR
                   ;
                   ;
                   ;
                   ;THE OXIDE SUBROUTINE POSITIONS THE RECR. HEAD
                   ; ON THE OXIDE IN PREPARATION TO WRITE DATA ON TO
                   ; THE TAPE.
                   ;
                   ;
09A6 3E83       OXIDE:  MVI  A,83H     ; 10000011
09A8 D3E8          OUT  CONTROL   ; LOAD CONTROL WORD
                   ;
09AA DBEA       CIP1:   IN   PORTC
09AC E6C4          ANI  04H  ; 000001CC
09AE C2AA09        JNZ  CIP1 ; CIP
                   ;
09B1 3E20          MVI  A,20H     ; 00100000
09B3 D3EA          OUT  PORTC     ; LOAD FORWARD
                   ;
09B5 DBEA       BOT1:   IN   PORTC
09B7 E608          ANI  08H  ; 00001000
09B9 CAB509        JZ   BOT1 ; BOT
                   ;
09BC CD8A08        CALL GAP  ;BLANK TAPE
09BF CD8A08        CALL GAP  ;BLANK TAPE
                   ;
09C2 3E30          MVI  A,30H     ; 00110000
09C4 D3EA          OUT  PORTC     ; STOP LOAD FORWARD
```

```
09C6 C9            RET
                   ;
                   ;
                   ;
                   ;  FUNCTION: WRCHAR
                   ;  INPUT    : A-CHAR
                   ;  OUTPUT   : A-CHAR VIA PORTA
                   ;  CALLS    : DELAY1
                   ;  DESTROYS: A,F/F
                   ;  DESCRIPTION:OUTPUTS CHAR IN ACCM TO CASSETTE TAPE
                   ;
                   ;
                   WRCHAR:
09C7 D3E8          OUT  PORTA      ;CHAR TO CASSETTE INPUT PORT
09C9 F5            PUSH PSW
09CA 3EF0          MVI  A,0F0H     ; 11110000
09CC D3EA          OUT  PORTC      ; WRITE/START
                   ;
09CE 3E01          MVI  A,01H
09D0 CD910A        CALL DELAY1     ;PROVIDES 1MS START PULSE
                   ;
09D3 3EB0          MVI  A,0B0H     ; 1C11C000
09D5 D3EA          OUT  PORTC      ;START PULSE LOW
                   ;
09D7 DBEA          STATHI: IN   PORTC
09D9 E601          ANI  1H    ; 00000001
09DB C2D709        JNZ  STATHI     ; STATUS HI  (STILL WRITING)
                   ;
09DE 3E07          MVI  A,07H
09EC CD910A        CALL DELAY1     ; 5MS DELAY
                   ;
09E3 F1            POP  PSW
09E4 C9            RET
                   ;
                   ;
                   ;
                   ; 'WRREC' SUBROUTINE WRITES A RECORD (1,2 OR 3) ON
                   ;TAPE
                   ;   D=RECORD TYPE
                   ;ENRTY  C=LENGTH OF RECORD
                   ;  H,L=START ADDR. OF RECORD
                   ;EXIT   B=CHECKSUM (CHKSUM)
                   ;
                   ;
                   ;
09E5 CD8A08        WRREC:  CALL GAP  ;BLANK TAPE
09E8 0600          MVI  B,0  ;INITIALIZE CHKSUM ZERO
```

```
09EA 3E00          MVI   A,0H ;WRITE THREE ZEROS
09EC CDC709        CALL  WRCHAR
09EF CDC709        CALL  WRCHAR
09F2 CDC709        CALL  WRCHAR
                   ;
09F5 7A            MOV   A,D  ;WRITE FILE TYPE
09F6 CDC709        CALL  WRCHAR
                   ;
09F9 79            MOV   A,C  ;WRITE RECORD LENGTH
09FA CDC709        CALL  WRCHAR
                   ;
09FD 7E          NEXTCH: MOV  A,M  ;ADDR OF FIRST CHAR
09FE CDC709        CALL  WRCHAR    ;OUTPUT CHAR ON TAPE
                   ;
0AC1 80            ADD   B    ;UPDATE CHECK SUM
0A02 47            MOV   B,A
                   ;
0A03 23            INX   H    ;MOVE POINTER
0AC4 CD            DCR   C    ;DECREASE COUNTER
0A05 C2FD09        JNZ   NEXTCH    ;JMP NEXT CHAR

0A08 7E            MOV   A,M  ;PUT CHAR IN ACCM
0A09 CDC709        CALL  WRCHAR    ;OUTPUT CHAR ON TAPE

0A0C 80            ADD   B    ;UP DATE CHECKSUM
0A0D 47            MOV   B,A  ;

0A0E 7E            MOV   A,B  ;MOVE CHKSUM INTO ACCM
0A0F CDC709        CALL  WRCHAR    ;OUTPUT CHKSUM ON TAPE

CA12 CD8A08        CALL  GAP  ;BLANK TAPE
0A15 C9            RET
                   ;
                   ;
                   ;
                   ;SUBROUTINE 'RANCHN' USES THE GLOBAL VARIABLES 'FSTCHN' AND
                   ;'GAIN'TO DEFINE THE CHANNEL GAIN TO BE USED IN THE CONVERSION
                   ;OF THE ANALOG INPUT.  THE RESULT IS STORED AT THE LOCATION
                   ;POINTED BY 'DATPTR'.
                   ;
                   ;
                   ;
0A16 2101F7      RANCHN: LXI  H,FCR     ;POINT HL TO FIRST CHANNEL REGISTER.
0A19 3A473C        LDA   GAIN ;LOAD GAIN
0A1C 4F            MOV   C,A
0A1D 3A483C        LDA   FSTCHN    ;LOAD CHANNEL.
0A20 B1            ORA   C    ;ADD GAIN BITS
0A21 77            MOV   M,A  ;LOAD FIRST CHANNEL REG.
```

```
0A22 2B          DCX  H     ;POINT TO COMMAND/STATUS REG.
0A23 3EC1        MVI  M,GO  ;START CONVERSION

0A25 7E        RAN1: MOV  A,M   ;READ STATUS
0A26 07          RLC         ;CHECK EOC STATUS
0A27 D2250A      JNC  RAN1  ;JMP IF CONVERSION NOT DONE

0A2A 3E00        MVI  M,0   ;RESET CONV-ENABLE IN CMD REG.
0A2C CD300A      CALL LODATA     ;LOAD DATA INTO MEMORY

0A2F C9          RET         ;EXIT RANCHN
                 ;
                 ;
                 ;
                 ;SUBROUTINE 'LODATA' TAKES THE A/D CONV DATA AND
                 ;LOADS IT INTO THE MEMORY LOCATION POINTED BY 'DATPTR'.
                 ;
                 ;
                 ;
                 LODATA:
0A30 2A04F7      LHLD ADDATA     ;LOAD CONVERTED DATA INTO HL
0A33 EB          XCHG        ;PUT CONVERTED DATA INTO DE

0A34 F5          PUSH PSW   ;SAVE ACCM AND FLAGS
0A35 3A01F7      LDA  FCR   ;LOAD GAIN,MUX ADDR REG
0A38 E60F        ANI  0FH   ;SAVE CHANNEL NUMBER
0A3A B3          ORA  E     ;ADD CHANNEL NUMBER TO DATA
0A3B 5F          MOV  E,A   ;STORE LSBYTE + CHAN NUMB IN REG E.
0A3C F1          POP  PSW   ;RESTORE ACCM AND FLAGS

0A3D 2A4B3C      LHLD DATPTR     ;LOAD ADDR OF DATA POINTER INTO HL
0A40 72          MOV  M,D   ;STORE DATA INTO MEMORY
0A41 2C          INR  L     ;INCREMENT DATA POINTER
0A42 7D          MOV  A,L   ;MOV BUFFER COUNTER INTO ACCM
0A43 324A3C      STA  KOUNTR     ;STORE BUFFER COUNTER IN KOUNTR

0A46 73          MOV  M,E   ;STORE LSBYTE DATA INTO MEMORY

0A47 FEFF        CPI  0FFH  ;BUFFER FULL
0A49 CC7F0A      CZ   DUMP  ;CALL DUMP IF BUFFER FULL

0A4C 2C          INR  L     ;INCREMENT DATA POINTER
0A4D 7D          MOV  A,L   ;MOV BUFFER COUNTER INTO ACCM
0A4E 324A3C      STA  KOUNTR     ;STORE BUFFER COUNTER IN KOUNTR

0A51 224B3C      SHLD DATPTR     ;SAVE NEW ADDR OF DATA POINTER
```

```
0A54 C9             RET         ;EXIT LDDATA.

                    ;
                    ;
                    ;
                    ;
                    ;   FUNCTION    :LDRNDT
                    ;   INPUT       :NONE
                    ;   OUTPUT        :NONE
                    ;   CALLS       :DUMP
                    ;   DESTROYS    :A,D,E,H,L,F/F
                    ;   DESCRIPTION:MOVES RANDOM LOAD DATA LOCATED IN BUFR1
                    ;                INTO BUFR IN 256 BYTE BLOCKS AND WRITES
                    ;                DATA ON CASSETTE FOR STORAGE
                    ;
                    ;
                    ;
                    LDRNDT:
0A55 010043         LXI   B,BUFR      ;LOAD BEGINING ADDR OF BUFR

0A58 210044         LXI   H,BUFR1     ;LOAD BEGINING ADDR OF BUFR1

                    LDRND1:
0A5B 7E             MOV   A,M   ;LOAD DATA BYTE INTO ACCM
0A5C FE1A           CPI   1AH   ;CHECK FOR END OF DATA SEED
0A5E CA760A         JZ    LDRND2      ;IF ZERO DUMP BUFFER

0A61 C2             STAX B      ;STORE ACCM AT ADDR IN BC
0A62 CC             INR   C     ;INCREMENT BUFR POINTER
0A63 23             INX   H     ;INCREMENT BUFR1 POINTER

0A64 7E             MOV   A,M   ;GET NEXT DATA BYTE
0A65 02             STAX B      ;STORE ACCM AT ADDR IN BC

0A66 C5             PUSH B      ;SAVE ADDR IN BC
0A67 79             MOV   A,C   ;LOAD ACCM WITH LSBYTE OF BC
0A68 324A3C         STA   KOUNTR      ;STORE LSBYT IN KOUNTR
0A6B FEFF           CPI   OFFH  ;CHECK FOR END OF BUFR
0A6D CC7F0A         CZ    DUMP  ;IF ZERO DUMP BUFR ON CASSETTE TAPE

0A70 C1             POP   B     ;RESTORE ADDR IN BC
0A71 23             INX   H     ;INCREMENT RANDOM DATA BUFFER
0A72 0C             INR   C     ;INCREMENT DATA BUFFER
0A73 C35B0A         JMP   LDRND1      ;GET NEXT RANDOM DATA BYTE

                    LDRND2:
0A76 CD7F0A         CALL  DUMP  ;DUMP BUFR ON CASSETTE TAPE
0A79 C9             RET         ;EXIT LDRNDT
```

```
                        ;
                        ;
                        ;
                        ;  FUNCTION: STOP
                        ;  INPUT    : NONE
                        ;  OUTPUT   : A-CHAR VIA PORTC
                        ;  CALLS    : NONE
                        ;  DESTROYS: A
                        ;  DESCRIPTION: REMOVES START SIGNAL FROM CASSETTE DRIVE
                        ;
                        ;
                        STOP:
0A7A 3EB0                  MVI  A,0B0H     ;REMOVE START SIGNAL
0A7C D3EA                  OUT  PORTC      ;OUTPUT TO PORTC
0A7E C9                    RET             ;EXIT STOP
                        ;
                        ;
                        ;
                        ;  FUNCTION: DUMP
                        ;  INPUT    : NONE
                        ;  OUTPUT   : D-RCORD TYPE
                        ;             C-RECORD LENGTH
                        ;             H,L-START ADDR OF RECORD
                        ;  CALLS    : WRREC
                        ;  DESTRCYS:
                        ;  DESCRIPTION: WRITE A RCORD OF DATA ON TAPE-
                        ;               RECORD LENGTH DETERMINED BY KCUNTR.
                        ;
                        ;
                        DUMP:
0A7F E5                    PUSH H     ;SAVE ADDRESS OF DATA POINTER
0A80 214A3C                LXI  H,KCUNTR  ;POINT TO ADDR OF COUNTER
0A83 4E                    MOV  C,M   ;RECORD LENGTH
0A84 210043                LXI  H,BUFR     ;POINT TO INITIAL ADDR OF MEMORY
0A87 1603                  MVI  D,03H      ;RECORD TYPE

0A89 CDE509                CALL WRREC      ;WRITE RECORD ON TAPE

0A8C CD7A0A                CALL STOP ;STOP CASSETTE TAPE
0A8F E1                    POP  H     ;RESTORE HL

0A90 C9                    RET             ;EXIT DUMP
                        ;
                        ;
```

```
                        ;
                        ;  FUNCTION: DELAY1
                        ;  INPUTS   : A-BYTE OF DATA REPRESENTING DESIRED DELAY
                        ;  OUTPUT   : NONE
                        ;  CALLS    : NONE
                        ;  DESTROYS:
                        ;  DESCIPTION: PROVIDES TIMING DELAYS-96H IN ACCM FOLLOWED
                        ;              BY CALL DELAY PROVIDES 100MS DELAY.
                        ;
                        ;
                        DELAY1:
0A91 C5                    PUSH B     ;SAVE BC
0A92 FE00               DEL0:  CPI  0     ;FINISHED
0A94 CAA30A                JZ   RETDEL    ;JMP ZERO TO RETURN
0A97 0669                  MVI  B,69H     ;MAGIC CONSTANT
0A99 00                 DEL1:  NOP
0A9A 00                    NOP
0A9B C5                    DCR  B
0A9C C2990A                JNZ  DEL1
0A9F 3D                    DCR  A     ;1MS GONE
0AA0 C3920A                JMP  DEL0
0AA3 C1                 RETDEL: POP  B     ;RESTORE BC
0AA4 C9                    RET        ;EXIT DELAY

                        ;
                        ;
                        ;THIS PROGRAM IS DESIGNED TO TEST THE MEMODYNE
                        ;TAPE RECORDER IN THE READ MODE OF OPERATION.
                        ;
                        ;
                        ;
                        ;
                        ;THE CLROX SUBROUTINE POSITIONS THE RECORDER HEAD
                        ;NEAR THE LEADER/OXIDE JUNCTION.
                        ;
                        ;
                        CLROX:
0AA5 3E83                  MVI  A,83H     ; 10000011
0AA7 D3EB                  OUT  CONTROL   ; LOAD CONTROL WORD
                        ;
0AA9 DBEA               CIP:   IN   PORTC
0AAB E604                  ANI  04H   ; 00000100
0AAD C2A90A                JNZ  CIP   ; CIP
                        ;
0AB0 3E20                  MVI  A,20H     ; 00100000
0AB2 D3EA                  OUT  PORTC     ; LOAD FORWARD
```

```
                        ;
0AB4 DBEA       BOT:    IN    PORTC
0AB6 E608               ANI   08H   ; 00001000
0AB8 CAB40A             JZ    BOT   ; BOT
                        ;
0ABB 3E30               MVI   A,30H       ; 00110000
0ABD D3EA               OUT   PORTC       ; STOP LOAD FORWARD
                        ;
0ABF C9                 RET
                        ;
                        ;
                        ;
                        ; THE READ SUBROUTINE TAKES A BYTE OF DATA FROM THE TAPE
                        ; AND STUFFS IT IN THE ACCUMULATOR.
                        ;
                        ;
0AC0 3E70       RDCHAR: MVI   A,70H       ; 01110000
0AC2 D3EA               OUT   PORTC       ; READ/START
                        ;
0AC4 3E01               MVI   A,01H       ;PROVIDES 1MS START PULSE
0AC6 CD910A             CALL  DELAY1
                        ;
0AC9 3E30               MVI   A,030H      ;00110000
0ACB D3EA               OUT   PORTC       ;START PULSE LOW
                        ;
0ACD DBEA       HI:     IN    PORTC
0ACF E602               ANI   02H   ; 00000010
0AD1 CACD0A             JZ    HI    ; TAPE SYNC HIGH
                        ;
0AD4 DBEA       LO:     IN    PORTC
0AD6 E602               ANI   02H   ; 00000010
0AD8 C2D40A             JNZ   LO    ; TAPE SYNC LOW
                        ;
0ADB 3E0A               MVI   A,0AH
0ADD CD910A             CALL  DELAY1      ;5MSEC DELAY
                        ;
0AE0 DBE9               IN    PORTB       ;DATA TO ACCUM.
0AE2 C9                 RET
                        ;
                        ;
                        ;  FUNCTION:RDREC
                        ;  ENTRY: H,L=START OF READ BUFFER
                        ;  EXIT:  B=CHKSUM
                        ;         D=RECORD TYPE
                        ;         C=RECORD LENGTH
                        ;         ACCM=ERROR FLAG
                        ;
```

```
0AE3 0600       RDREC:  MVI   B,0H ;INITIALIZE CHKSUM
0AE5 CDC00A     RDC1:   CALL  RDCHAR     ;READ FIRST BYTE
0AE8 FE00           CPI   0H   ;
0AEA C2E50A         JNZ   RDC1 ;CONTINUE FIRST NON ZERO BYTE

0AED CDC00A     RDC2:   CALL  RDCHAR     ;GET FILE TYPE
0AF0 FE00           CPI   0H   ;
0AF2 CAED0A         JZ    RDC2 ;CONTINUE NOT ZERO

0AF5 57             MOV   D,A  ;STORE FILE TYPE
0AF6 CDC00A         CALL  RDCHAR     ;GET FILE LENGTH
0AF9 4F             MOV   C,A  ;STORE     FILE LENGTH
0AFA F5             PUSH  PSW  ;SAVE FILE LENGTH ON STACK

0AFB CDC00A     RDC3:   CALL  RDCHAR     ;GET NEXT DATA BYTE
0AFE 77             MOV   M,A  ;STORE DATA IN RAM
0AFF 80             ADD   B    ;UPDATE CHECKSUM
0B00 47             MOV   B,A  ;STORE NEW CHECKSUM
0B01 23             INX   H    ;INCREMENT DATA BUFFER
0B02 0D             DCR   C    ;DECREMENT BUFFER COUNTER
0B03 C2FB0A         JNZ   RDC3 ;COUNTER NOT ZERO GET NEXT DATA BYTE
0B06 CDC00A         CALL  RDCHAR     ;GET LAST BYTE
0B09 77             MOV   M,A  ;
0B0A 80             ADD   B    ;UPDATE CHECK SUM
0B0B 47             MOV   B,A  ;
0B0C F1             POP   PSW  ;GET FILE LENGTH FROM STACK
0B0D 4F             MOV   C,A  ;FILE LENGTH INTO C

0B0E CDC00A         CALL  RDCHAR     ;GET CHECKSUM FROM TAPE
0B11 A8             XRA   B    ;IF ERROR ACCM NOT ZERO

0B12 E5             PUSH  H    ;SAVE RAM ADDRESS
0B13 213D0B         LXI   H,MSG4     ;ADDR CHECKSUM ERROR MSG
0B16 C47F08         CNZ   MSG  ;ACCM NOT ZERO PRINT MSG2
0B19 E1             POP   H    ;RESTOR RAM ADDRESS

0B1A C9             RET        ;EXIT RDREC
                    ;
                    ;
                    ;
                    ; FUNCTION: DSFILE
                    ;  INPUT: B-CHECKSUM
                    ;         C-FILE LENGTH
                    ;         D-FILE TYPE
                    ;  OUTPUT:C-CHARACTER ON CONSOLE
                    ;  CALLS: CONOUT, NMOUT
                    ;
                    ;
```

```
                        ;
                        DSFILE:
0B1B 15                     DCR   C     ;
0B1C CA2B0B                 JZ    DSFIL1      ;JMP FILE TYPE ONE

0B1F 15                     DCR   D     ;
0B20 CA2B0B                 JZ    DSFIL2      ;JMP FILE TYPE TWO

0B23 15                     DCR   C     ;
0B24 CA430B                 JZ    DSFIL3      ;JMP FILE TYPE THREE

0B27 15                     DCR   D     ;
0B28 CA930B                 JZ    DSFIL4      ;JMP FILE TYPE FOUR

                        DSFIL1:
                        DSFIL2:
0B2B 21C043                 LXI   H,BUFR      ;POINT START OF MEMORY BUFFER
0B2E 224B3C                 SHLD  DATPTR      ;STORE ADDREES OF BUFFER

0B31 2A4B3C             DSF1A:  LHLD DATPTR     ;POINT ADDRESS OF BUFFER
0B34 7E                     MOV   A,M   ;PUT DATA IN ACCM AT ADDRESS IN HL
0B35 C5                     PUSH  B     ;SAVE CHECKSUM AND FILE LENGTH
0B36 CD7208                 CALL  CONOUT      ;OUTPUT CHAR TO CONSOLE
0B39 C1                     POP   B     ;RESTORE CHECKSUM AND FILE LENGTH
0B3A 2C                     INR   L     ;INCREMENT MEMORY BUFFER POINTER
0B3B 224B3C                 SHLD  DATPTR      ;SAVE NEW POINTER
0B3E 0D                     DCR   C     ;DECREMENT MEMORY COUNTER
0B3F C2310B                 JNZ   DSF1A       ;JMP COUNTER NOT ZERO
0B42 C9                     RET         ;RETURN

                        DSFIL3:
0B43 21CC43                 LXI   H,BUFR      ;POINT TO START MEMORY BUFFER
0B46 224B3C                 SHLD  DATPTR      ;STORE ADDRESS OF BUFFER IN HL

0B49 21E03D                 LXI   H,MSG5A     ;PROMPT-HARD COPY STRAIN DATA
0B4C CD7F08                 CALL  MSG   ;DISPLAY MESSAGE

0B4F CD6608                 CALL  CONIN       ;INPUT CHAR FROM CONSOLE
0B52 CD7208                 CALL  CONOUT      ;OUTPUT CHAR TO CONSOLE
0B55 FE59                   CPI   'Y'   ;IF YES DATA DISPLAYED
0B57 C28C0B                 JNZ   DSF3C       ;IF NOT ZERO SKIP DISPLAY
0B5A CD9C0B                 CALL  CROUT       ;OUTPUT CR LF
                        DSF3A:

0B5D 3E20                   MVI   A,' '       ;MOV ASCII SPACE CHAR INTO ACCM
0B5F CD7208                 CALL  CONOUT      ;OUTPUT SPACE TO CONSOLE

0B62 2A4B3C             DSF3B:  LHLD DATPTR     ;POINT TO BUFFER ADDRESS
```

```
0B65 7E           MOV   A,M  ;PUT DATA IN ACCM
0B66 C5           PUSH  B    ;SAVE CHECKSUM AND FILE LENGTH
0B67 F5           PUSH  PSW  ;SAVE ACCM - FLAGS
0B68 CDC202       CALL  NMOUT      ;CONVERT 8-BIT BINARY TO TWO ASCII CHAR
0B6B F1           POP   PSW  ;RESTORE ACCM - FLAGS.
0B6C C1           POP   B    ;RESTORE CHECHSUM - FILE LENGTH

0B6D 23           INX   H    ;INCREMENT MEMORY BUFFER POINTER
0B6E 224B3C       SHLD  DATPTR     ;SAVE NEW DATA BUFFER POINTER
0B71 0D           DCR   C    ;DECREMENT MEMORY BUFFER COUNTER
0B72 CA8C0B       JZ    DSF3C      ;IF ZERO OUTPUT CR LF

0B75 7D           MOV   A,L  ;LOAD LSBYTE OF HL
0B76 E60F         ANI   0FH  ;CHECK FOR END OF LINE
0B78 C2810B       JNZ   DSF3D      ;NO-CHECK FOR SECOND BYTE
0B7B CD9C0B       CALL  CROUT      ;YES-OUTPUT CR LF
0B7E C35D0B       JMP   DSF3A      ;JMP OUTPUT SPACE

                  DSF3D:
0B81 7D           MOV   A,L  ;LOAD LSBYTE OF HL
0B82 E601         ANI   01H  ;CHECK FOR SECOND BYTE
0B84 FE01         CPI   01H  ;
0B86 CA620B       JZ    DSF3B      ;NO-GET NEXT BYTE
0B89 C35D0B       JMP   DSF3A      ;YES-OUTPUT SPACE

                  DSF3C:
0B8C CD9C0B       CALL  CROUT      ;OUTPUT CR LF
0B8F CD9C0B       CALL  CROUT      ;
0B92 C9           RET        ;


                  DSFIL4:
0B93 21D03D       LXI   H,MSG5     ;LOAD MSG3 ADDRESS
0B96 CD7F08       CALL  MSG  ;

0B99 C3860F       JMP   MAIN ;


                  CROUT:
0B9C 3E0D         MVI   A,CR ;OUTPUT CR
0B9E CD7208       CALL  CONOUT     ;OUTPUT TO CONSOLE

0BA1 3E0A         MVI   A,LF ;OUTPUT LF
0BA3 CD7208       CALL  CONOUT     ;OUTPUT TO CONSOLE

0BA6 C9           RET        ;EXIT CROUT

                  GOUGE:
```

```
0BA7 21F440        LXI  H,MSGH1    ;CISPLAY INSTRUCTIONS
0BAA CD7F08        CALL MSG  ;
0BAC C3920F        JMP  MAIN1      ;

                   CACZRO:
0BB0 210080        LXI  H,ADCHS    ;SET EOTH DAC TO ZERO
0BB3 2208F7        SHLD DACO ;
0BB6 220AF7        SHLD DAC1 ;
CBB9 C9            RET
                   ;**********************************************************
                   ;
                   ;
                   ;  PROM 4
                   ;
                   ;
                   ;**********************************************************


0C00               ORG  0C00H
                   SGLCHN:
CC00 F5            PUSH PSW  ;SAVE ALL REGISTERS
0C01 C5            PUSH B    ;
0C02 D5            PUSH D    ;
0C03 E5            PUSH H    ;
0C04 2101F7        LXI  H,FCR      ;
CC07 3A473C        LDA  GAIN ;
0C0A 4F            MOV  C,A  ;
0CC8 3A483C        LDA  FSTCHN     ;
0C0E B1            ORA  C    ;
0C0F 77            MOV  M,A  ;

CC1C 2E            DCX  H    ;
0C11 3601          MVI  M,GO ;

0C13 7E          SGLCH1: MOV  A,M  ;
0C14 07            RLC
0C15 D2130C        JNC  SGLCH1     ;

0C18 3600          MVI  M,0  ;
0C1A CD22CC        CALL VALDT      ;
0C1D E1            POP  H    ;RESTORE ALL REGISTERS
0C1E D1            POP  D    ;
0C1F C1            POP  B    ;
0C20 F1            POP  PSW  ;

0C21 C9            RET       ;
```

```
                VALDT:
0C22 2AC4F7         LHLD ADDATA     ;LOAD CONVERTOR DATA INTO HL
0C25 EB             XCHG        ;CONVERTOR DATA INTO DE

0C26 2A9642         LHLD UPPER      ;CHECK UPPER BOUNDARY
0C29 CDAC02         CALL HILO ;CARRY BIT ( 0-H DE)
                              ;              ( 1-HL> DE)
0C2C DA4D0C         JC    VALDT2    ;CHECK LOWER BOUNDARY

0C2F 3EFF           MVI   A,0FFH    ;SET FLAG
0C31 32543C         STA   FLAG ;

0C34 2A513C         LHLD XLST ;LOAD LAST X INTO HL
0C37 CDA002         CALL HILO ;CARRY BIT ( 0-HDE)
                              ;              ( 1-HL>DE)
0C3A D2AE0C         JNC   VALDT7    ;JMP UPDATE LASTX

0C3D 7A             MOV   A,D   ;
0C3E 94             SUB   H     ;
0C3F CABD0C         JZ    VALDT9    ;

0C42 3A533C         LDA   SIGN ;LOAD SIGN
0C45 FE01           CPI   01H   ;
0C47 CAAE0C         JZ    VALDT7    ;JMP UPDATE UPDATE LASTX

0C4A C39E0C         JMP   VALDT5    ;JMP IF LASTX >= X

                VALDT2:
0C4D 2A9842         LHLD LOWER      ;CHECK LOWER BOUNDARY
0C50 CDA002         CALL HILO ;CARRY BIT ( 0-HL DE )
0C53 D26F0C         JNC   VALDT4    ;JMP IF X LOWER

0C56 3EFF           MVI   A,0FFH    ;SET FLAG OFFH
0C58 32543C         STA   FLAG ;

0C5B 2A513C         LHLD XLST ;LOAD LASTX
0C5E CDA002         CALL HILO ;CARRY BIT ( 0- HDE )
                              ;              ( 1- HL>DE )
0C61 DAAE0C         JC    VALDT7    ;JMP UPDATE LASTX

0C64 3A533C         LDA   SIGN ;LOAD SIGN FLAG
```

```
0C67 FEC0          CPI   00H  ;
0C69 CAAE0C        JZ    VALDT7     ;JMP UPDATE LASTX

0C6C C3A60C        JMP   VALDT6     ;CHANGE SIGN-UPDATE LASTX-STORE

                   VALDT4:
0C6F 3A543C        LDA   FLAG ;POINT TO FLAG BUFFER
0C72 FEFF          CPI   0FFH ;IF FLAG SET CONTINUE
0C74 C0            RNZ           ;RETURN IF FLAG NOT SET

0C75 2A513C        LHLD  XLST ;HL_LASTX
0C78 CDA002        CALL  HILO ;CARRY BIT ( 0- HDE)
                              ;              ( 1- HL>DE)

0C7B DA8E0C        JC    NEGSGN     ;CHECK FOR SIGN CHANGE

                   POSSGN:
0C7E 3A533C        LDA   SIGN ;PUT SIGN IN ACCM.
0C81 FE00          CPI   00H  ;COMPARE NEGATIVE SIGN
0C83 CAAE0C        JZ    VALDT7     ;JMP UPDATE LASTX
0C86 3E00          MVI   A,00H      ;SET FLAG 00H
0C88 32543C        STA   FLAG ;
0C8B C3A60C        JMP   VALDT6     ;CHG SIGN-UPDATE LASTX-STORE

                   NEGSGN:
0C8E 3A533C        LDA   SIGN ;PUT SIGN IN ACCM.
0C91 FE01          CPI   01H  ;COMPARE NGATIVE SIGN
0C93 CAAE0C        JZ    VALDT7     ;JMP UPDATE LASTX
0C96 3E00          MVI   A,00H      ;SET FLAG 00H
0C98 32543C        STA   FLAG
0C9B C39E0C        JMP   VALDT5     ;CHG SIGN-UPDATE LASTX-STORE

                   VALDT5:
0C9E 3E01          MVI   A,01H      ;CHANGE SIGN (NEGATIVE)
0CA0 32533C        STA   SIGN ;STORE IN SIGN BUFFER
0CA3 C3B50C        JMP   VALDT8     ;JMP UPDATE LASTX-STORE

                   VALDT6:
0CA6 3E00          MVI   A,00H      ;CHANGE SIGN (POSITIVE)
0CA8 32533C        STA   SIGN ;STORE IN SIGN BUFFER
0CAB C3B50C        JMP   VALDT8     ;JMP UPDATE LASTX-STORE

                   VALDT7:
0CAE EB            XCHG          ;EXCHANGE DE AND HL
0CAF 22513C        SHLD  XLST ;UPDATE LASTX
0CB2 C3BD0C        JMP   VALDT9     ;RETURN

                   VALDT8:
```

```
0CB5 EB          XCHG       ;EXHANGE DE AND HL
0CB6 22513C      SHLD XLST  ;UPDATE LASTX
0CB9 EB          XCHG       ;EXCHANGE DE AND HL
0CBA CDBF0C      CALL STORE      ;STORE SIGNIFICANT DATA

0CBD 0C        VALDT9: NOP
0CBE C9          RET        ;EXIT VALDT



               STORE:
0CBF F5          PUSH PSW   ;SAVE ACCM AND FLAGS
0CC0 3A01F7      LDA  FCR   ;LOAD GAIN,MUX ADDR REG
0CC3 E60F        ANI  0FH   ;SAVE CHANNEL NUMBER
0CC5 B3          ORA  E     ;ADD CHN NUMBER TO DATA
0CC6 5F          MOV  E,A   ;STORE LSBYTE + CHAN NUMBER IN REG E
0CC7 F1          POP  PSW   ;RESTORE ACCM AND FLAGS

0CC8 2A4B3C      LHLD DATPTR     ;LOAD ADDR. OF DATA POINTER INTO HL
0CCB 72          MOV  M,D   ;STORE DATA INTO MEMORY
0CCC 2C          INR  L     ;INCREMENT DATA POINTER
0CCD 7D          MOV  A,L   ;MOV BUFFER COUNTER INTO ACCM
0CCE 324A3C      STA  KOUNTR     ;STORE COUNTER

0CD1 73          MOV  M,E   ;STORE LSBYTE INTO MEMORY
0CD2 FEFF        CPI  0FFH  ;BUFFER FULL
0CD4 CC7F0A      CZ   DUMP  ;DUMP MEMORY IF BUFFER FULL

0CD7 2C          INR  L     ;INCREMENTDATA POINTER
0CD8 7D          MOV  A,L   ;MOVE BUFFER COUNTER INTO ACCM
0CD9 324A3C      STA  KOUNTR     ;STORE COUNTER

0CDC 224B3C      SHLD DATPTR     ;SAVE DATA POINTER

0CDF C9          RET        ;EXIT STORE



               RAMP:
0CE0 210044      LXI  H,BUFR1    ;POINT RANDOM LOAD BUFFER ADDR
0CE3 224D3C      SHLD BFPTR1     ;STORE RANDOM LOAD BUFFER ADDR

0CE6 2A4D3C      LHLD BFPTR1     ;LOAD RANDOM LOAD BUFFER ADDR INTO HL
0CE9 56          MOV  D,M   ;LOAD RANDOM DATA INTO DE
0CEA 23          INX  H     ;INCREMENT BUFFER ADDR
```

```
0CEB 5E           MOV   E,M   ;

                  RMP1:
0CEC 23           INX   H     ;INCREMENT BUFR1 ADDR
0CED 46           MOV   B,M   ;LOAD NEXT RANDOM DATA INTO BC
0CEE 7E           MOV   A,M   ;CHECK FOR 1AH E O D SEED
0CEF FE1A         CPI   1AH   ;
0CF1 CA590D       JZ    RMP9  ;
0CF4 23           INX   H     ;INCREMENT BUFR1 ADDR
0CF5 4E           MOV   C,M   ;

0CF6 224D3C       RMP3:   SHLD BFPTR1     ;STORE RANDOM LOAD BUFFER ADDR

0CF9 78           MOV   A,B   ;MOV MSBYTE BC INTO ACCM.
0CFA BA           CMP   D     ;COMPARE MSBYTE OF BC WITH DE
0CFB CA510D       JZ    RMP2  ;  D=B  JMP RESTART 1
0CFE DA2C0D       JC    RMP8  ;  D > B  JMP
                              ;  D B  CONTINUE
0D01 62           MOV   H,D   ;LOAD DE INTO HL
0D02 69           MOV   L,C   ;SET LSBYTE IN BC EQUAL IN HL

                  RMP4:
0D03 CD610D       CALL  WDAC0     ;OUTPUT DAC0
0D06 24           INR   H     ;INCREMENT MSBYTE HL
0D07 7C           MOV   A,H   ;LOAD MSBYTE INTO ACCM
0D08 B8           CMP   B     ;COMPARE WITH MSBYTE BC
0D09 CA240D       JZ    RMP5  ;  H=B  JMP GET NEXT DATA BYTE
0D0C CDE104       CALL  DELAY     ;EACH DELAY = 1MSEC
0D0F CDB104       CALL  DELAY     ;
0D12 CDE104       CALL  DELAY     ;

0D15 CD000C       CALL  SGLCHN    ;CHECK SELECTED CHAN FOR DATA

0D18 CDB104       CALL  DELAY     ;
0D1B CDB104       CALL  DELAY     ;
0D1E CDB104       CALL  DELAY     ;

0D21 C3030D       JMP   RMP4 ;JMP CONTINUE TO INCREMENT MSBYTE

                  RMP5:
0D24 54           MOV   D,H   ;UPDATE DE MAINTAIN DAC0 OUTPUT
0D25 5D           MOV   E,L   ;
0D26 2A4D3C       LHLD  BFPTR1    ;LOAD HL WITH DATA BUFFER ADDRESS
0D29 C3EC0C       JMP   RMP1 ;JMP GET NEXT DATA BYTE

                  RMP8:
0D2C 62           MOV   H,D   ;MOV DE INTO HL
0D2D 69           MOV   L,C   ;
```

```
                        RMP7:
0D2E CD610D                CALL WDAC0      ;OUTPUT DAC0
0D31 F5                    PUSH PSW  ;SAVE FLAGS
0D32 25                    DCR  H    ;DECREMENT MSBYTE
0D33 F1                    POP  PSW  ;RESTORE FLAGS
0D34 7C                    MOV  A,H  ;MSBYTE INTO ACCM
0D35 B8                    CMP  B    ;COMPARE MSBYTE BC
0D36 CA240D                JZ   RMP5 ;JMP GET NEXT DATA BYTE

0D39 CDB104                CALL DELAY      ;
0D3C CDB104                CALL DELAY      ;
0D3F CDB104                CALL DELAY      ;

0D42 CDC00C                CALL SGLCHN     ;

0D45 CDB104                CALL DELAY      ;
0D48 CDB104                CALL DELAY      ;
0D4B CDB104                CALL DELAY      ;

0D4E C32E0D                JMP  RMP7 ;CONTINUE TO DECREMENT

0D51 62                 RMP2:   MOV  H,D  ;MOVE DE REG INTO HL
0D52 69                    MOV  L,C  ;

0D53 CD610D                CALL WDAC0      ;OUTPUT DAC0
0D56 C3240D                JMP  RMP5 ;JMP GET NEXT LOAD DATA

                        RMP9:
0D59 2680                  MVI  H,80H      ;SET DAC0 ZERO VOLTS
0D5B 2E00                  MVI  L,00H      ;
0D5D CD610D                CALL WDAC0      ;

0D60 C9                    RET             ;EXIT RAMP


0D61 2208F7             WDAC0:  SHLD DAC0 ;WRITE TO DAC0
0D64 C9                    RET

0D65 22CAF7             WDAC1:  SHLD DAC1 ;WRITE TO DAC1
0D68 C9                    RET

                        ;***************************************************
                        ;
                        ;
                        ;  EXECUTIVE ROUTINES
```

```
                        ;
                        ;
                        ;**********************************************************
                        SORCLD:
0D69 21AB40                LXI   H,MSGG1    ;CISPLAY S>
0D6C CD7F08                CALL  MSG  ;

0D6F CD9608                CALL  HEADNG     ;INPUT SIMULATED OR ACTUAL A/C DATA
0D72 3A9A42                LDA   TYPDTA     ;PUT TYPE DATA IN ACCM
0D75 FE41                  CPI   'A'  ;A=ACTUAL DATA
0D77 C28A0D                JNZ   SRCLD1     ;
0D7A CDA609                CALL  OXIDE      ;PUT RECORD HEAD ON OXIDE

0D7D 21803C                LXI   H,MSG1     ;LOAD A/C CHARACTERISTICS
0D80 1601                  MVI   D,01H      ;RECORD TYPE ( 01 )
0D82 0E95                  MVI   C,95H      ;RECORD LENGTH

0D84 CDE509                CALL  WRREC      ;WRITE RECORD ON TAPE
0D87 C3970D                JMP   SRCLD2     ;JMP STOP CASSETTE TAPE

                        SRCLD1:
0D8A CDA609                CALL  OXIDE      ;PUT RECORD HEAD ON OXIDE
0D8D 21153D                LXI   H,MSG2     ;LOAD SIMULATED LOAD CHARACTERISTICS
0D90 1602                  MVI   D,02H      ;RECORD TYPE ( C2 )
0D92 0E9A                  MVI   C,9AH      ;RECORD LENGTH

0D94 CDE509                CALL  WRREC      ;WRITE RECORD ON TAPE
0D97 CD7A0A             SRCLD2: CALL STOP ;STOP CASSETTE TAPE

0D9A CD550A                CALL  LDRNDT     ;LOAD SOURCE DATA ON CASSETTE

0D9D 21B03D                LXI   H,MSG3     ;LOAD END OF FILE RECORD
0DA0 1604                  MVI   D,04H      ;RECORD TYPE ( C4 )
0DA2 0E0D                  MVI   C,0DH      ;RECORD LENGTH

0DA4 CDE509                CALL  WRREC      ;WRITE RECORD ON TAPE

0DA7 CD7A0A                CALL  STOP ;STOP CASSETTE TAPE

0DAA 21FA3E                LXI   H,MSGC     ;POINT MSGC ADDREES
0DAD CD7F08                CALL  MSG  ;DISPLAY MESSAGE ON CONSOLE

0DB0 C3860F                JMP   MAIN ;RETURN MAIN
```

```
                ACCCHN:
0DB3 21B240         LXI   H,MSGG2    ;CISPLAY A> PROMPT
0DB6 CD7F08         CALL  MSG  ;

0DB9 CD9608         CALL  HEADNG     ;

0DBC 3A9A42         LDA   TYPDTA     ;PUT DATA TYPE IN ACCM
0DBF FE41           CPI   'A'  ;
0DC1 C2D40D         JNZ   ADC01      ;ACTLAL=A  SIMULATED=S

0DC4 CDA609         CALL  CXIDE      ;
0DC7 21803C         LXI   H,MSG1     ;
0DCA 1601           MVI   D,01H      ;
0DCC CE95           MVI   C,95H      ;

0DCE CDE509         CALL  WRREC      ;
0DD1 C3E10D         JMP   ADC02      ;

                ADC01:
0DD4 CDA609         CALL  OXIDE      ;
0DD7 21153D         LXI   H,MSG2     ;
0DDA 1602           MVI   D,02H      ;
0DDC CE9A           MVI   C,9AH      ;

0DDE CDE509         CALL  WRREC      ;

                ADC02:
0DE1 CD7A0A         CALL  STOP ;

0DE4 217E40         LXI   H,MSGF     ;DISPLAY SELECT CHANNEL #
0DE7 CD7F08         CALL  MSG  ;
0DEA 21483C         LXI   H,FSTCHN   ;POINT TO FIRST CD
HANNEL REG
0DED CD6608         CALL  CONIN      ;INPUT CHAR FRM CONSOLE
0DFC CD7208         CALL  CONOUT     ;ECHC CHAR TO CCNSOLE
0DF3 4F             MOV   C,A  ;
0DF4 CDDF01         CALL  CNVBN      ;
0DF7 32483C         STA   FSTCHN     ;STORE IN FIRST CHANNEL REG

0DFA 3ECC           MVI   A,X1 ;SET GAIN ECUAL TO ONE
0DFC 32473C         STA   GAIN ;STORE IN GAIN BUFFER

0DFF 3E00           MVI   A,0H ;INITIALIZE BUFFER CCUNTER TC ZERO
0E01 324A3C         STA   KOUNTR     ;STCRE IN COUNTER BUFFER

0E04 210043         LXI   H,BUFR     ;POINT TC DATA BUFFER
0E07 224B3C         SHLD  DATPTR     ;SORCLC POINTER FCR 'RANCHN'
```

```
0ECA CD160A     ADC03:  CALL RANCHN     ;CONVERT SELECTED CHANNEL

CECD DBED           IN   CRTS ;DEPRESS ANY KEY TO END MISSION
0ECF E6C2           ANI  RBR  ;
0ED1 CA0A0E         JZ   ADC03      ;IF ZERO CONTINUE TO CONVERT SELECTED
                         ;CHANNEL

0ED4 CD6608         CALL CONIN      ;READ DEPRESSED KEY
0ED7 CD7208         CALL CONOUT     ;ECHO DEPRESSED KEY
0EDA FE04           CPI  CNTRLD     ;IF CNTRLD DUMP BUFFER ON TAPE
                         ;TERMINATE MISSION
0EDC C20A0E         JNZ  ADC03      ;NOT CONTROL D CONTINUE MISSION

0EDF CD7F0A         CALL DUMP ;DUMP CONVERTED DATA ON TAPE
0EE2 213D03         LXI  H,MSG3     ;LOAD END OF FILE
0EE5 1604           MVI  D,04H      ;
0EE7 0E0D           MVI  C,0DH      ;

0EE9 CDE509         CALL WRREC      ;
0EEC CD7A0A         CALL STOP ;

0EEF C3860F         JMP  MAIN ;RETURN TO MAIN ROUTINE


                ADCMXM:
0EF2 218940         LXI  H,MSGG3    ;
0EF5 CD7F08         CALL MSG  ;

0EF8 CD9608         CALL HEADNG     ;

0EFB 3A9A42         LDA  TYPDTA     ;PUT TYPE DATA IN ACCM
0EFE FE41           CPI  'A'  ;
0F00 C2530E         JNZ  ADCR1      ;

0F03 CDA609         CALL OXIDE      ;
0F06 21803C         LXI  H,MSG1     ;
0F09 1601           MVI  D,01 ;
0F0B 0E95           MVI  C,95H      ;

0F0D CDE509         CALL WRREC      ;
0F10 C3600E         JMP  ADCR2      ;

                ADCR1:
0F13 CDA609         CALL CXIDE      ;
0F16 21153D         LXI  H,MSG2     ;
0F19 1602           MVI  D,02H      ;
0F1B CE9A           MVI  C,9AH      ;
```

```
0E5C CDE509        CALL WRREC      ;

                ADCR2:
0E6C CD7A0A        CALL STOP ;

0E63 217E40        LXI  H,MSGF     ;DISPLAY SELECT CHANNEL NUMBER
0E66 CD7F08        CALL MSG  ;
0E69 21483C        LXI  H,FSTCHN   ;POINT TO FIRST CHANNEL BUFFER
0E6C CD6608        CALL CONIN      ;
0E6F CD7208        CALL CONOUT     ;
0E72 4F            MOV  C,A  ;
0E73 CDDF01        CALL CNVBN      ;CONVERT ASII TO BINARY
0E76 32483C        STA  FSTCHN     ;

0E79 3E00          MVI  A,X1 ;
0E7B 32473C        STA  GAIN ;STORE IN GAIN BUFFER

0E7E 3E00          MVI  A,0H ;BUFFER COUNTER ZERO
0E8C 324A3C        STA  KOUNTR     ;STORE IN KOUNTR BUFFER

0E83 210043        LXI  H,BUFR     ;POINT TO DATA BUFFER
0E86 224B3C        SHLD DATPTR     ;LOAD POINTER FOR RANCHN

                ADCR3:
0E89 CD000C        CALL SGLCHN     ;CONVERT SINGLE CHN (PEAKS/VALLEYS)

0E8C DBED          IN   CRTS ;DEPRESS CONTROL D TO END MISSION
0E8E E602          ANI  RBR  ;
0E90 CA890E        JZ   ADCR3      ;IF ZERO CONVERT SELECTED CHANNEL

0E93 CD6608        CALL CONIN      ;READ DEPRESSED KEY
0E96 CD7208        CALL CONOUT     ;ECHO DEPRESSED KEY
0E99 FE04          CPI  CNTRLD     ;IF CNTRLD DUMP BUFFER
0E9B C2890E        JNZ  ADCR3      ;NOT CONTROL C CONVERT SELECTED CHANNEL

0E9E CD7F0A        CALL DUMP ;DUMP CACNVERTED DATA ON TAPE
0EA1 21803D        LXI  H,MSG3     ;LOAD END OF FILE
0EA4 1604          MVI  D,04H      ;
0EA6 0E0D          MVI  C,0CH      ;

0EA8 CDE509        CALL WRREC      ;

0EAB CD7A0A        CALL STOP ;

0EAE C3860F        JMP  MAIN ;RETURN MAIN
```

```
                READ:
0EB1 21C040         LXI  H,MSGG4    ;DISPLAY R> PROMPT
0EB4 CD7F08         CALL MSG  ;

0EB7 CDA50A         CALL CLROX      ;PUT READ HEAD NEAR OXICE
0EBA 210043     READ1:  LXI  H,BUFR     ;POINT TC CATA BUFFER
0EBD CDE30A         CALL RDREC      ;READ FILE INTO MEMORY

0ECC CD7A0A         CALL STOP ;

0EC3 CD1B0B         CALL CSFILE     ;DISPLAY FILE

0EC6 DBEA           IN   PORTC      ;CHECK PORTC STATUS FOR END OF TAPE
0EC8 E608           ANI  08H  ;00001000
0ECA C2BA0E         JNZ  READ1      ;EOT

0ECD CD7A0A         CALL STOP ;STOP TAPE
0ED0 C3860F         JMP  MAIN ;RETURN MAIN


                LOAD:
0ED3 21CE40         LXI  H,MSGG6    ;DISPLAY L> PROMPT
0ED6 CD7F08         CALL MSG  ;

0ED9 CDA50A         CALL CLROX      ;PLACE READ HEAD ON OXIDE
0EDC 210044         LXI  H,BUFR1    ;POINT TO RANDOM LOAD BUFFER
                                ;BEGINING ADDR 4400H

0EDF C36105         JMP  RDLOAD     ;READ LOAD FRM CASSETTE TAPE
                                ;INTO ADDR 4400-7F00H
                                ;RDLOAD WILL RETURN USER TO MAIN



                MASTER:
0EE2 21C740         LXI  H,MSGG5    ;DISPLAY E> PROMPT
0EE5 CD7F08         CALL MSG  ;

0EE8 CD9608         CALL HEADNG     ;

0EEB 3A9A42         LDA  TYPCTA     ;
0EEE FE41           CPI  'A'  ;
```

```
0EF0 C2030F        JNZ   MSTR1      ;

0EF3 CDA609        CALL  OXIDE      ;
0EF6 21803C        LXI   H,MSG1     ;
0EF9 1601          MVI   D,01H      ;
0EFB CE95          MVI   C,95H      ;

0EFD CDE509        CALL  WRREC      ;
0F00 C3100F        JMP   MSTR2      ;

                MSTR1:
0F03 CDA609        CALL  OXIDE      ;
0F06 21153D        LXI   H,MSG2     ;
0F09 1602          MVI   D,02H      ;
0F0B 0E9A          MVI   C,9AH      ;

0F0D CDE509        CALL  WRREC      ;

                MSTR2:
0F10 CD7A0A        CALL  STOP ;
0F13 216A42        LXI   H,MSGI     ;DISPLAY NUMBER OF RUNS
0F16 CD7F08        CALL  MSG  ;
0F19 21463C        LXI   H,REPEAT   ;POINT TO REPEAT ADDR
0F1C CD6608        CALL  CONIN      ;INPUT CHAR FRM CONSOLE
0F1F CD7208        CALL  CONOUT     ;OUTPUT CHAR TO CONSOLE
0F22 4F            MOV   C,A  ;
0F23 CDDF01        CALL  CNVBN      ;CONVERT ASCII TO BINARY
0F26 32463C        STA   REPEAT     ;

0F29 217E40        LXI   H,MSGF     ;
0F2C CD7F08        CALL  MSG  ;
0F2F 21483C        LXI   H,FSTCHN   ;
0F32 CD6608        CALL  CONIN      ;
0F35 CD7208        CALL  CONOUT     ;
0F38 4F            MOV   C,A  ;
0F39 CDDF01        CALL  CNVBN      ;
0F3C 32483C        STA   FSTCHN     ;

0F3F 3E00          MVI   A,X1 ;
0F41 32473C        STA   GAIN ;

0F44 3E00          MVI   A,0H ;
0F46 324A3C        STA   KOUNTR     ;

0F49 21C043        LXI   H,BUFR     ;
0F4C 224B3C        SHLD  DATPTR     ;

0F4F 214F3C        LXI   H,X  ;
```

```
0F52 36C0          MVI   M,00H      ;
0F54 2C            INR   L      ;
0F55 3600          MVI   M,0H ;
0F57 2C            INR   L      ;
0F58 36C0          MVI   M,0H ;
0F5A 2C            INR   L      ;
0F5B 3680          MVI   M,80H      ;
0F5D 2C            INR   L      ;
0F5E 3600          MVI   M,00H      ;
0F60 2C            INR   L      ;
0F61 3600          MVI   M,00H      ;

                MSTR3:
0F63 CDE00C        CALL  RAMP ;
0F66 CD7A0A        CALL  STOP ;
0F69 21463C        LXI   H,REPEAT  ;POINT TO REPEAT STORAGE ADDR
0F6C 7E            MOV   A,M  ;PUT NUMBER OF RUNS IN ACCM
0F6D 3D            DCR   A      ;
0F6E 32463C        STA   REPEAT     ;
0F71 FE00          CPI   0H   ;
0F73 C2630F        JNZ   MSTR3      ;


0F76 21B03D        LXI   H,MSG3     ;
0F79 1604          MVI   D,04H      ;
0F7B 0E0D          MVI   C,0DH      ;

0F7D CDE509        CALL  WRREC      ;
0F80 CD7A0A        CALL  STOP ;

0F83 C3860F        JMP   MAIN ;




                MAIN:
0F86 21A440        LXI   H,MSGG     ;
0F89 CD7F08        CALL  MSG  ;

0F8C 21D540        LXI   H,MSGH     ;DISPLAY CONTROL G FOR INSTRUCTIONS
0F8F CD7F08        CALL  MSG  ;

0F92 31FF7F     MAIN1:  LXI   SP,7FFFH  ;INITIALIZE STACK POINTER

0F95 CDB00B        CALL  DACZRO     ;SET DAC'S TO ZERO
```

```
0F98 DBED          IN    CRTS ;CHECK FOR DEPRESSED KEY
0F9A E602          ANI   RBR  ;02H
0F9C CA920F        JZ    MAIN1      ;LOOP UNTIL KEY DEPRESSED

0F9F CD6608        CALL  CONIN      ;READ DEPRESSED KEY
0FA2 CD7208        CALL  CONOUT     ;ECHO KEY

0FA5 FE07          CPI   CNTRLG     ;CHECK FOR CONTROL G
0FA7 CAA70B        JZ    GOUGE      ;

0FAA FEC3          CPI   CNTRLC     ;
0FAC CA0800        JZ    MONITOR    ;ENTER MONITOR

0FAF FE13          CPI   CNTRLS     ;CHECK FOR CONTROL S
0FB1 CA690D        JZ    SORCLD     ;

0FB4 FEC1          CPI   CNTRLA     ;CHECK FOR CONTROL A
0FE6 CAB30D        JZ    ADCCHN     ;

0FB9 FE02          CPI   CNTRLB     ;CHECK FOR CONTROL B
0FBB CA320E        JZ    ADCMXM     ;

0FBE FE12          CPI   CNTRLR     ;CHECK FOR CONTROL R
0FC0 CAB10E        JZ    READ ;

0FC3 FE0C          CPI   CNTRLL     ;CHECK FOR CONTROL L
0FC5 CAD30E        JZ    LOAD ;


0FC8 FE05          CPI   CNTRLE     ;CHECK FOR CONTROL E
0FCA CAE20E        JZ    MASTER     ;

0FCD C3920F        JMP   MAIN1      ;

CFDC               END        ;
```

```
;"HEXLINK" IS A MODIFICATION OF "LINK" PROGRAM DEVELOPED BY
; LT. MACK T. ELLIOTT,USN. "LINK" IS DESIGNED TO INTERFACE THE
;MDS 800 (AND MODEL 40 PRINTER) WITH CP/CMS THRUGH A 1200
;BAUD TELEPHONE LINE. MODIFICATIONS TO "LINK" ARE OUTLINE
;WITH ASTERISKS (****).

     ORG   100H
     JMP   START
     BDOS  EQU   5H    ;ENTRY POINT
     XON   EQU   11H   ; END OF LINE FROM VIRTUAL MACHINE
     XOFF  EQU   13H   ;END OF LINE TO VIRTUAL MACHINE
     CR    EQU   0DH   ;CARRIAGE RETURN
     LF    EQU   0AH   ;LINE FEED
     FF    EQU   0CH   ;FORM FEED
     EOF   EQU   1AH   ;END OF FILE CHAR FOR DISK WRITE
     RUB   EQU   7FH   ;DELETE CHARACTER
     CNTLC       EQU   03H   ;WARM BOOT
     CNTLD       EQU   04H   ;RESTORES "DIRECT LINKUP" MODE
     CNTLG       EQU   07H   ;PRINT INSTRUCTIONS
     CNTLI       EQU   09H   ;TAB CHARACTER
     CNTLP       EQU   10H   ;CONTROL P TURNS PRINTER ON AND OFF
     CNTLR       EQU   12H   ;CONTROL R FOR RECEIVE FILE
     CNTLT       EQU   14H   ;CONTROL T FOR TRANSMIT FILE
     CNTLU       EQU   15H   ;DELETE LINE
     FLIMIT      EQU   2A0H  ;ALLOWS 416 RECORDS OF 128 BYTES
     BUFFMAX EQU       0D000H    ;MAX SIZE OF TRANSFERRED FILE
     COUNT:      DS    2     ;COUNT OF RECORDS TRANSFERRED
     FCOUNT:     DS    2     ;FILE COUNT RECORD
     PPREG       DS    1     ;PRINTER CONTROL REG;0 OFF,1 ON
     MSG1:       DB    CR,LF,'DIAL 2721 FOR LINE -- CONTROL G FOR INSTRUCTIONS',CR,L
     MSG2:       DB    '                    CONTROL C - REBOOT',CR,LF
          DB     '                    CONTROL D - RETURN TO DIRECT LINKUP',CR,LF
          DB     '                    CONTROL G - INSTRUCTIONS',CR,LF
          DB     '                    CONTROL I - TAB',CR,LF
          DB     '                    CONTROL P - PRINTER ON/OFF',CR,LF
          DB     '                    CONTROL R - RECEIVE FILE',CR,LF
          DB     '                    CONTROL T - TRANSMIT FILE',CR,LF
          DB     '                    CONTROL U - DELETE LINE',CR,LF
          DB     '                    RUBOUT    - DELETE CHARACTER',CR,LF
          DB     '                    XMIT      - INTERRUPT CMS',CR,LF,'$'
     MSG3:       DB    CR,LF,'DISK:FILENAME.FILETYPE',CR,LF,'$'
     MSG4:       DB    CR,LF,'REPEAT',CR,LF,'$'
     MSG5:       DB    'EDIT $'
     MSG5A:      DB    'FILE NOT FOUND',CR,LF,'>$'
     MSG6:       DB    'TRANSMITTING',CR,LF,'$'
```

```
MSG7:     DB   'TRANSMISSION COMPLETE',CR,LF,'$'
MSG8:     DB   'FILE$'
MSG10:    DB   'DUMPF $'
MSG11:    DB   'NO DIRECTORY SPACE AVAILABLE',CR,LF,'>$'
MSG12:    DB   'RECEIVING',CR,LF,'$'
MSG13:    DB   'DISK FULL',CR,LF,'$'
MSG14:    DB   ' RECORDS TRANSFERRED',CR,LF,'>$'
MSG15:    DB   'CMS FILENAME FILETYPE   ',CR,LF,'$'
MSG17:    DB   'FILE EXCEEDS BUFFER - ONLY 52K BYTES TRANSFERRED',CR,LF,'$'
MSG18:    DB   'RELOADING',CR,LF,'$'
MSG19:    DB   'SAVE$'
STACK:    DS   20
STKBTM    EQU  $
START:    LXI  SP,  STKBTM
          MVI  A,   0
          STA  PPREG          ;INITIALLY PRINTER IS OFF
          LXI  D,   MSG1
          CALL MESSAGE        ;PROMPTS USER TO CALL COMPUTER CENTER
          CALL BOARD          ;INITIALIZES SBC 534 BOARD AND USARTS
          IN   60H
          ;
          ;TRANSMIT MODE
          ;
TX:
          IN   61H            ;CHECKS LINE FOR MESSAGE
          ANI  2
          JNZ  CRCV1
          IN   0F7H           ;CHECKS KEYBOARD FOR DEPRESSED KEY
          ANI  2
          JZ   TX             ;LOOPS UNTIL ONE OF THE ABOVE OCCURS
          MVI  C,   1
          CALL BDOS           ;READ CHAR FROM CONSOLE
          CPI  CR             ;CHECK FOR CR
          JZ   RCV            ;SWITCH TO RECEIVE MODE
          CPI  CNTLP
          JZ   PRTCONT        ;TURN PRINTER ON/OFF
          CPI  CNTLR
          JZ   FILERX         ;RECEIVE FILE MODE
          CPI  CNTLT
          JZ   FILETX         ;TRANSMIT FILE MODE
          CPI  CNTLC
          JZ   00H            ;ESCAPE BY REBOOTING
          CPI  CNTLG          ;PRINT INSTRUCTIONS
          JZ   GOUGE
          CPI  CNTLI
          CZ   CHNG4          ;TRANSMIT TAB CHAR "     "
          CPI  RUB
          CZ   CHNG2          ;TRANSMIT DELETE CHAR SYMBOL "
```

```
    CPI   CNTLU
" AND XOFF
    MOV   C,    A
    CPI   XON
    JZ    CTX
    LDA   PPREG           ;CHECK IF PRINTER ON
    CPI   0
    JZ    CTX
    MOV   A,    C
    CALL  DRIVER
CTX:
    MOV   A,    C
    CALL  SEND            ;SENDS CHAR TO VIRTUAL MACHINE
    JMP   TX              ;LOOPS FOREVER
CHNG1:
    MVI   A,    '     '
    RET
CHNG2:
    MVI   A,    08H       ;BACKSPACE
    CALL  CONOUT
    MVI   A,    '
    RET
CHNG3:
'
    CALL SEND
    JMP  RCV
CHNG4:
    MVI   A,    '     '
    CALL  CONOUT
    RET
    ;
    ;RECEIVE MODE
    ;
RCV:
    LDA   PPREG           ;CHECK IF PRINTER ON
    CPI   0
    JZ    CRCV
    MVI   A,    CR        ;START NEW LINE ON PRINTER
    CALL  DRIVER
    MVI   A,    LF
    CALL  DRIVER
CRCV:
    MVI   A,    XOFF      ;END OF LINE CHAR
    CALL  SEND
CRCV1:
    ;HL REGISTER POINTS TO ADDR FOR NEXT WORD RECEIVED
    ;DE REGISTER POINTS TO ADDR OF NEXT WORD TO BE PRINTED
    LXI   h,    BUFF      ;FIFO BUFFER ADDR FOR RECEIVED DATA
```

```
     LXI   D,    BUFF
RX1:
     CALL  BREAK
     IN    61H          ;CHECK LINE FOR CHAR
     ANI   02H
     JZ    CKPRT              ;IF LINE NOT READY, CHECK IF BUFFER CAUGHT UP

RX:
     IN    60H          ;INPUT WORD FROM LINE
     ANI   7FH
     CPI   XON
     JZ    CATCH              ;IF END OF LINE, LET BUFFER CATCH UP
     CPI   XOFF
     JZ    RX1          ;FILTER OUT XOFF CHAR
     MOV   M,    A      ;STORE CHAR
     INX   H
     JMP   RX1          ;LOOP UNTIL END OF LINE
CATCH:
     MOV   M,    A      ;STORE LAST WORD
LOOOP:
     LDAX  D            ;NEXT WORD TO BE PRINTED
     CPI   XON
     JZ    TX           ;IF CAUGHT UP, GO BACK TO TRANSMIT MODE
     CALL  CONOUT             ;PRINT ON CONSOLE
     LDA   PPREG              ;CHECK IF PRINTER ON
     CPI   0
     JZ    BACK
     LDAX  D
     CALL  DRIVER
BACK:
     INX   D
     JMP   LOOOP              ;LOOP UNTIL CAUGHT UP
GOUGE:
     LXI   D,    MSG2
GLOOP:
     LDAX  D
     CPI   '$'
     JZ    TX
     CALL  CONOUT
     MOV   B,    A
     LDA   PPREG
     CPI   0
     JZ    GLP
     MOV   A,    B
     CALL  DRIVER
GLP:
     INX   D
     JMP   GLOOP
```

```
PRTCONT:
    LDA   PPREG           ;CHECK IF PRINTER ON OR OFF
    CPI   0
    JNZ   PRTOFF          ;IF ON, WANT TO TURN OFF
    CALL  USART2
    MVI   A,   1
    STA   PPREG           ;SUBSEQUENT ROUTINES CHECK THIS ADDR
    MVI   A,   CR    ;START PRINTER ON NEW LINE
    CALL  DRIVER
    MVI   A,   LF
    CALL  DRIVER
    JMP   TX          ;RETURN TO TRANSMIT MODE
PRTOFF:
    MVI   A,   30H   ;CONTROL WORD TO TURN PRINTER OFF
    OUT   63H
    MVI   A,   0
    STA   PPREG           ;SUBSEQUENT ROUTINES CHECK THIS ADDR
    JMP   TX
DRIVER:       ;DRIVES PRINTER USART
    PUSH  PSW
SLO:  IN    63H           ;WAIT UNTIL XMITTER READY
    RRC
    JNC   SLO
    POP   PSW
    OUT   62H
    RET
CONOUT:       ;DRIVES CONSOLE USART
    PUSH  PSW
SLO2:
    IN    0F7H
    RRC
    JNC   SLO2
    POP   PSW
    OUT   0F6H
    RET
CKPRT:        ;KEEPS TRACK OF WHICH RECEIVED DATA HAS BEEN PRINTED
    MOV   A,   L
    CMP   E
    JZ    RX1         ;IF CAUGHT UP, NO NEED TO PROCEED
    IN    0F7H
    RRC
    JNC   RX1         ;IF CONSOLE NOT READY, NO NEED TO PROCEED
    LDA   PPREG           ;CHECK IF PRINTER ON
    CPI   0
    JZ    CKP2        ;IF PRINTER NOT ON, NO NEED TO PROCEED
    IN    63H
    RRC
    JNC   RX1         ;IF PRINTER NOT READY, NO NEED TO PROCEED
```

```
CKP2:
     LDAX D          ;NEXT WORD TO BE PRINTED
     OUT  0F6H       ;OUT TO CONSOLE
     OUT  62H        ;OUT TO PRINTER
     INX  D
     MOV  A,   L     ;CHECK AGAIN TO SEE IF BUFFER IS
     CMP  E          ;CAUGHT UP - IF SO, RESET BUFFER ADDR
     JNZ  RX1
     LXI  H,   BUFF
     LXI  D,   BUFF
     JMP  RX1
SEND:                ;DRIVES USART ON HIGH SPEED LINE
     PUSH PSW
WAIT:
     IN   61H
     RRC
     JNC  WAIT
     POP  PSW
     OUT  60H
     RET
BREAK:               ;CHECKS KEYBOARD FOR INTERRUPT
     IN   0F7H
     ANI  2
     RZ              ;IF NONE, GO BACK TO RECEIVE MODE
     IN   0F6H       ;IF INTERRUPT PRESENT, CHECK FOR BREAK
     ANI  7FH
     CPI  XON
     RNZ             ;IGNORE IF NOT BREAK
     MVI  A,   3FH   ;CONTROL WORD DRIVES XMIT LINE LOW
     OUT  61H        ;-WANT TO HOLD LINE LOW FOR 2 WORDLENGTHS
     LXI  B,   400H  ;WAIT 10 MILLISECS
DLA1:
     DCX  B
     MOV  A,   B
     CPI  0
     JZ   DLA3
     IN   61H        ;CHECK LINE FOR CHAR
     ANI  2
     JZ   DLA1
     IN   60H
     MOV  M,   A
     INX  H
DLA3:
     LXI  B,   5A0H  ;DELAY 16 MILLISEC
DLA2:
     DCX  B
     MOV  A,   B
     CPI  0
```

```
    JNZ  DLA2
RESET:
    MVI  A,   37H
    OUT  61H
    RET

BOARD:
;
;
;   THIS ROUTINE INITIALIZES THE 534 BOARD, THE TIMERS, AND THE TWO USARTS
;   NEEDED TO DRIVE THE IBM HIGH SPEED LINE AND THE MODEL 40 PRINTER
;
;   BASE ADDR OF 534 BOARD          60H
;   CMD ADDR OF LINE USART          61H
;   DATA ADDR OF LINE USART         60H
;   CMD ADDR OF PTR USART           63H
;   DATA ADDR OF PTR USART          62H
;
;   TWO MORE USARTS AND ONE 8255 PARALLEL INTERFACE AND THEIR TIMERS ARE
;   AVAILABLE ON THE 534 BOARD.  NEW INTERFACES MUST BE PROGRAMMED BEFORE USE
;
;
    DI              ;DISABLES 8080 INTERRUPTS
    OUT  6FH        ;RESETS BOARD
    OUT  6CH        ;SELECTS BOARD CONTROL BLOCK
    CALL TIMER          ;INITIALIZE PIT CHIPS
    CALL USART          ;INITIALIZE USARTS FOR IBM LINE AND PTR
    EI              ;REENABLES INTERRUPTS
    RET
;
;MUST SET UP TIMER CHIPS ACCORDING TO PAGE 3-12 OF 534 MANUAL
;  CHIP 0 HAS THREE TIMERS ON IT
;  TIMERS 0 AND 1 OF CHIP 0 ARE CONNECTED TO USARTS 1 AND 2
;  RESPECTIVELY, DRIVING THE IBM LINE AND THE PRINTER
;
TIMER:
    OUT  6CH            ;PUTS BOARD INTO CONTROL BLOCK
    MVI  A,   36H   ;SELECT TIMER 0 FOR LINE USART
    OUT  63H        ;ADDR OF COUNTER 0 MODE CONTROL
    MVI  A,   40H   ;
    OUT  60H        ;SET N=40H IN TIMER C
    MVI  A,   0H    ;CCLK/N=19.2KHZ FOR 1200 BAUD,
    OUT  60H        ;BRF=16X
    MVI  A,   76H   ;
    OUT  63H        ;SELECT TIMER 1 FOR PTR USART
    MVI  A,   8H    ;
```

```
        OUT   61H           ;SET N=8 IN TIMER 1
        MVI   A,    0H      ;CCLK/N=153.6KHZ FOR 9600 BAUD,
        OUT   61H           ;BRF=16X
        OUT   6DH           ;PUTS BOARD IN DATA BLOCK
        RET
;
;
;SET UP BOTH USARTS WITH RESETS AND MODE WORDS
;
USART:
        MVI   A,    0CAH ;2 STOP, ODD PAR DISABLED, 7 BITS, BRF=16
        OUT   61H
        MVI   A,    5AH  ;1 STOP, ODD PAR DISABLED, 7 BITS, BRF=16
        OUT   63H
        MVI   A,    37H
        OUT   61H
        RET
USART2:
        MVI   A,    33H
        OUT   63H
        RET
              ;THIS SECTION PERTAINS TO TRANSFERRING COMPLETE
              ;FILES BETWEEN MDS AND IBM 360
              ;
        FCB   EQU   5CH         ;FCB ADDR
        FCBDN       EQU   FCB+0             ;DISK NAME
        FCBFN       EQU   FCB+1             ;FILENAME(8CHAR)
        FCBFT       EQU   FCB+9             ;FILETYPE (3CHAR)
        FCBRL       EQU   FCB+12            ;REEL NUMBER
        FCBRC       EQU   FCB+15            ;FILE RECORD COUNT (0-127)
        FCB2:       DS    33          ;NEW FILENAME AND FILETYPE
        FCBCR       EQU   FCB+32            ;NEXT RECORD NUMBER
FILETX:       ;USER PROMPTS CONSLE FOR FILE TO BE XMITTED, SETS UP FILE
              ;CONTROL BLOCK, OPENS NEW CMS FILE, TRANSMITS FILE, AND
              ;RETURNS USER TO DIRECT CMS LINKUP
              ;
        MVI   A,    0
        STA   COUNT
        STA   COUNT+1
        CALL  RESTRT            ;SETS UP FILE CONTROL BLOCK
        CALL  CRLF
        CALL  CPNAME            ;CP/CMS FILENAME, FILETYPE
        CALL  CRLF
        CALL  OPEN          ;OPENS DISK FILE
        CALL  FILERD            ;READS DISK FILE
        CALL  CMS           ;PREPARES CMS TO RECEIVE FILE
        CALL  ANS           ;WAITS FOR ANSWER
        CALL  XMIT          ;TRANSMITS FILE
```

```
     CALL ANS
     CALL FILE       ;"FILES" FILE IN CMS
     CALL ANS
     CALL TALLY           ;PRINTS OUT RECORD COUNT
     JMP  TX         ;RETURNS TO TRANSMIT MODE
FILERX:         ;SUBR PROMPTS CONSOLE FOR FILE TO BE RECEIVED, SETS UP FILE
          ;CONTROL BLOCK AND CREATES FILE ON FLOPPY DISK, RECEIVES FILE
          ;FROM CMS AND ECHOES ON CONSOLE, CLOSES FILE AND RESTORES
          ;USER TO DIRECT CMS LINKUP
          ;
     MVI  A,   0
     STA  COUNT
     STA  COUNT+1
     CALL CPNAME
     CALL CRLF
     CALL RESTRT          ;SETS UP FILE CONTROL BLOCK
     CALL CRLF
     CALL MAKE       ;DELETES AND CREATES DISK FILE
     CALL BETA       ;PREPARES CMS TO TRANSMIT FILE
     CALL CRLF
     CALL HAUL       ;RECEIVES FILE FROM CMS
     CALL FILEWR          ;WRITES FILE ON DISK
     CALL CLOSE           ;CLOSES DISK FILE
     CALL TALLY           ;PRINTS RECORD COUNT;
     JMP  TX         ;RETURNS TO TRANSMIT MODE
RESTRT:         ;CLEARS OUT OLD FILE CONTROL BLOCK AND SETS UP NEW ONE
     LXI  D,   MSG3
     CALL MESSAGE         ;PROMPTS "FILENAME.FILETYPE"
     MVI  A,   0
     STA  FCB2       ;PADS NEW FCB WITH "C(11 BLANKS)C000"
     LXI  H,   FCB2+1
     MVI  A,   20H        ;BLANK CHAR
     MVI  B,   11
PAD1:
     MOV  M,   A
     INX  H
     DCR  B
     JNZ  PAD1
     MVI  A,   0
     MVI  B,   4
     LXI  H,   FCB2+12
PAD2:
     MOV  M,   A
     INX  H
     DCR  B
     JNZ  PAD2
     MVI  C,   1
     CALL BDOS
```

```
     CPI   'A'          ;ASKS FOR DESIRED DISK AND NOTIFIES DISK DRIVE
     JZ    AONE
     CPI   'B'
     JZ    BONE
     JMP   REPEAT
AONE:
     MVI   E,    0
     JMP   DSK

BONE:
     MVI   E,    1
     JMP   DSK
DSK:
     MVI   C,    14    ;CHANGES DISK DRIVE SELECTION
     CALL  BDOS
     MVI   C,    1
     CALL  BDOS
     CPI   ':'          ;NEXT CHAR MUST BE ":"
     JNZ   REPEAT            ;IF NOT, START OVER
     MVI   B,    9
     LXI   H,    FCB2+1
FNAME:
     PUSH  B
     PUSH  H
     MVI   C,    1
     CALL  BDOS
     POP   H
     POP   B
     CPI   CNTLC
     JZ    00
     CPI   CNTLD
     JZ    DIRECT
     CPI   CNTLU
     JZ    DUMMY
     CPI   '.'
     JZ    FTYPE
     MOV   M,    A
     INX   H
     DCR   B            ;IF FILENAME EXCEEDS 8 CHAR, START OVER
     JZ    REPEAT
     JMP   FNAME
FTYPE:
     MVI   B,    4
     LXI   H,    FCB2+9
FTYPE1:
     PUSH  B
     PUSH  H
     MVI   C,    1
```

```
        CALL  BDOS
        POP   H
        POP   B
        CPI   CNTLC
        JZ    00
        CPI   CNTLD
        JZ    DIRECT
        CPI   CNTLU
        JZ    DUMMY
        CPI   CR
        RZ
        MOV   M,    A
        INX   H
        DCR   B             ;IF FILETYPE EXCEEDS 3 CHAR, START OVER
        JZ    REPEAT
        JMP   FTYPE1
DUMMY:
        CALL  CRLF
        JMP   RESTRT
REPEAT:
        LXI   D,    MSG4 ;PROMPTS "REPEAT"
        CALL  MESSAGE
        JMP   RESTRT            ;START OVER
CPNAME:
        LXI   D,    MSG15       ;PROMPTS "CMS FILENAME FILETYPE
        CALL  MESSAGE
        LXI   D,    BUFF40
NAME2:
        PUSH  D
        MVI   C,    1
        CALL  BDOS
        POP   D
        CPI   CNTLC
        JZ    00
        CPI   CNTLD
        JZ    DIRECT
        CPI   CNTLU
        JZ    DUMMY2
        CPI   CR
        JZ    NAME3
        STAX  D
        INX   D
        JMP   NAME2
NAME3:
        MVI   A,    '$'
        STAX  D
        RET
DUMMY2:
```

```
     CALL CRLF
     JMP  CPNAME
DIRECT:
     LXI  SP,  STKBTM
     MVI  A,   XOFF
     CALL SEND
     JMP  CRCV1
MAKE:
     MVI  C,   19    ;DELETES ANY OLD DISK FILE HAVING
     LXI  D,   FCB2  ;FILENAME, FILETYPE LISTED IN NEW FCB
     CALL BDOS
     MVI  C,   22    ;CREATES NEW DISK FILE NAMED ABOVE
     LXI  D,   FCB2
     CALL BDOS
     CPI  255
     JZ   NOROOM          ;ZERO INDICATES FULL DISK
     XRA  A
     STA  FCB2+32         ;ZEROES FILE RECORD COUNTER
     RET
NOROOM:
     LXI  D,   MSG11      ;PROMPTS "DISK FULL"
     CALL MESSAGE
     JMP  TX

CRLF:
     MVI  C,   2     ;STARTS NEW LINE ON CONSOLE
     MVI  E,   CR
     CALL BDOS
     MVI  C,   2
     MVI  E,   LF
     CALL BDOS
     RET
OPEN:
     LXI  D,   FCB2 ;OPENS DISK FILE FOR READING
     MVI  C,   15
     CALL BDOS
     CPI  255
     JZ   BADF       ;ZERO INDICATES NO SUCH FILE ON DISK
     XRA  A
     STA  FCB2+32         ;ZEROES FILE RECORD COUNTER
     CALL CRLF
     RET
BADF:
     LXI  D,   MSG5A      ;PROMPTS "FILE NOT FOUND"
     CALL MESSAGE
     INX  SP         ;ADJUSTS STACK POINTER
     INX  SP
     JMP  TX         ;RETURNS TO TRANSMIT MODE
```

```
MESSAGE:        ;PRINTS MESSAGE AT ADDR IN DE ON CONSCLE
        MVI   C,    9
        CALL  BDOS
        RET
FILERD:         ;READS ENTIRE DISK FILE INTO RAM STARTING AT
                ;BUFF (LIMITED TO 52K BYTES)
FILERDO:
        LXI   H,    FLIMIT
        SHLD  FCOUNT
        LXI   D,    BUFF
FILERD1:
        PUSH  D
        MVI   C,    26
        CALL  BDOS              ;CHANGES DMA BUFFER ADDR
        LXI   D,    FCB2
        MVI   C,    20
        CALL  BDOS              ;READ FILE RECORD
        POP   D
        PUSH  PSW
        CALL  COUNTER
        LXI   H,    80H  ;INCREMENTS BUFF BY 80H
        DAD   D
        XCHG
        POP   PSW
        CPI   0
        RNZ                 ;IF NOT ZERO, EOF CONTAINED IN LAST RECORD
        LHLD  FCOUNT
        DCX   H
        SHLD  FCOUNT
        MOV   A,    H
        CPI   0
        JNZ   FILERD1           ;ZERO IF BUFFER HAS BEEN EXCEEDED
        INX   D
        MVI   A,    XOFF ;TEMPORARY EOF -- PROGRAM WILL TRANSMIT
        STAX  D             ;FIRST 52K BYTES OF FILE, THEN
        RET                 ;COME BACK TO READ MORE
FILEWR:         ;WRITES DISK FILE BY SAME ALGORITHM AS ABOVE
        LXI   D,    BUFF
CONT:
        MVI   B,    80H  ;MUST CHECK EACH RECORD FOR EOF CHAR
        CALL  COUNTER
INLOOP:
        PUSH  D
INLOOP2:
        LDAX  D
        CPI   EOF           ;IF EOF, THIS WILL BE LAST RECORD WRITTEN
        JZ    LAST
        INX   D
```

```
        DCR   B
        JNZ   INLOOP2
        POP   D
        PUSH  D
        MVI   C,    26    ;CHANGE DMA BUFFER ADDR
        CALL  BDOS
        LXI   D,    FCB2
        MVI   C,    21    ;WRITE ONE DISK RECORD
        CALL  BDOS
        POP   D
        PUSH  PSW
        LXI   H,    80H   ;INCREMENT BUFF BY 80H
        DAD   D
        XCHG
        POP   PSW
        CPI   1
        JZ    ERR1        ;1 INDICATES DISK FULL
        JMP   CONT
LAST:                     ;WRITE LAST DISK RECORD
        POP   D
        MVI   C,    26
        CALL  BDOS
        LXI   D,    FCB2
        MVI   C,    21
        CALL  BDOS
        CPI   1
        JZ    ERR1
        RET
ERR1:
        LXI   D,    MSG13       ;PROMPTS "DISK FULL"
        CALL  MESSAGE
        RET
CLOSE:                    ;CLOSES DISK FILE
        LXI   D,    FCB2
        MVI   C,    16
        CALL  BDOS
        LXI   D,    MSG7 ;PROMPTS "TRANSMISSION COMPLETE"
        CALL  MESSAGE
        RET
TALLY:                    ;PRINTS OUT RECORD COUNT
        LDA   COUNT
        RAR
        RAR
        RAR
        RAR
        ANI   0FH
        ADI   30H
        CALL  CONOUT
```

```
     LDA   COUNT
     ANI   0FH
     ADI   30H
     CALL  CONOUT
     LDA   COUNT+1
     RAR
     RAR
     RAR
     RAR
     ANI   0FH
     ADI   30H
     CALL  CONOUT
     LDA   COUNT+1
     ANI   0FH
     ACI   30H
     CALL  CONOUT
     LXI   D,    MSG14
     CALL  MESSAGE
     RET
COUNTER:        ;KEEPS TRACK OF RECORDS READ/WRITTEN
     LDA   COUNT+1
     ADI   1
     DAA
     STA   COUNT+1
     LDA   COUNT
     ACI   0
     STA   COUNT
     RET
HAUL:           ;RECEIVES WORDS FROM LINE USART AND STORES AT BUFF
     LXI   D,    MSG12       ;PROMPTS "RECEIVING"
     CALL  MESSAGE
     LXI   D,    BUFF ;FIRST TWO WORDS WILL BE CR AND LF
                      ;DON'T WANT THEM ON DISK
     LXI   B,    BUFFMAX     ;BUFF LIMIT IS 52K BYTES
     MVI   A,    XOFF
     CALL  SEND          ;TELL CMS TO START SENDING
FRX1:           ;CHECK USART FOR CHAR
;*************************************************
     PUSH  B
     CALL  GETWD
     CPI   'R'   ;CHECK FOR R IN RECORD
     CZ    OMIT  ;
     CALL  0FEC7H
     JC    FRX1+1      ;CARRY ON RETURN INDICATES BAD HEX
     ANI   0FH
     RAL
     RAL
     RAL
```

```
     RAL
     MOV  B,A
FRX2:
     CALL GETWD
     CALL 0FEC7H
     JC   FRX2
     CRA  B
     POP  B
;*****************************************************
     STAX D          ;STORE IN BUFFER
     INX  D          ;INCREMENT BUFFER ADDR
     DCX  B          ;CHECK BUFFER LIMIT NOT EXCEEDED
     MOV  A,   B
     CPI  0
     JZ   EXCEED
     CALL BREAK2
     JMP  FRX1       ;LOOP FOREVER
GETWD:
     IN   61H
     ANI  2
     JZ   GETWD
     IN   60H
     CPI  XON
     JZ   MARK
     RET
OMIT:
     CALL GETWD      ;
     CPI  LF    ;
     RZ         ;
     JMP  OMIT  ;
;*****************************************************************
BREAK2:              ;CHECK KEYBOARD FOR INTERRUPT--
               ;IF INTERRUPT EXISTS, RESET STACK POINTER
               ;AND JUMP TO DIRECT LINKUP MODE
               ;WHERE INTERRUPT CONDITION WILL BE NOTED
               ;AND A SIGNAL SENT TO CMS
     IN   0F7H
     ANI  2
     RZ
     LXI  SP,  STKBTM
     JMP  CRCV1
MARK:          ;MARK END OF FILE WITH "EOF"
          ;LAST CHARS RECEIVED ARE CR,LF,NULL,R;>
          ;WANT TO BACK UP TO LAST VALID WORD
     POP  B
     POP  PSW  ;DUMMY POP
     MVI  A,   EOF
     STAX D
```

```
     RET
EXCEED:
     LXI   D,    MSG17      ;PROMPTS "BUFFER LIMIT EXCEEDED"
     CALL MESSAGE
     MVI   A,    EOF   ;MARKS END OF FILE-REMAINDER OF FILE IS LOST
     STAX D
     RET
BETA:             ;SENDS "PRINT " TO CMS
     LXI   D,    MSG10
GAMMA:
     LDAX D
     CPI   '$'
     JZ    DELTA
     CALL CONOUT
     CALL SEND
     INX   D
     JMP   GAMMA
DELTA:            ;SENDS "FILENAME FILETYPE" TO CMS
     LXI   D,    BUFF40
EPSILON:
     LDAX D
     CPI   '$'
     RZ
     CALL CONOUT
     CALL SEND
     INX   D
     JMP   EPSILON
CMS:              ;SETS UP CMS TO RECEIVE FILE BY COMMANDING
          ;"EDIT FILENAME FILETYPE"
     LXI   D,    MSG5
CMS2:
     LDAX D
     CPI   '$'
     JZ    CMS3
     CALL CONOUT
     CALL SEND
     INX   D
     JMP   CMS2
CMS3:
     LXI   D,    BUFF40
CMS4:
     LDAX D
     CPI   '$'
     JZ    CMS5
     CALL CONOUT
     CALL SEND
     INX   D
     JMP   CMS4
```

```
CMS5:
     MVI   A,    XOFF
     CALL  SEND
     RET
ANS:              ;ECHOES CMS ANSWER TO CONSOLE
     IN    61H
     ANI   2
     JZ    ANS
     IN    60H
     CPI   XON
     RZ
     CPI   XOFF         ;FILTERS OUT XOFF
     JZ    ANS
     CALL  CONOUT
     JMP   ANS
ANS2:             ;RECEIVES CMS ANSWERS AND ECHOES TO CONSOLE
                  ;FILTERS OUT XOFF,CR,LF,AND >
     IN    61H
     ANI   2
     JZ    ANS2
     IN    60H
     CPI   XON
     RZ
     CPI   XOFF
     JZ    ANS2
     CPI   CR
     JZ    ANS2
     CPI   LF
     JZ    ANS2
     CPI   '>'
     JZ    ANS2
     CALL  CONOUT
     JMP   ANS2
           ;TRANSMITS FILE TO CMS
XMIT:
     LXI   D,    MSG6 ;PROMPTS "TRANSMITTING"
     CALL  MESSAGE
     CALL  PAUSE            ; DELAY 100 MICROSECS AT BEGINNING OF EACH LINE
     LXI   D,    BUFF
     MVI   C,    83H  ;132 BYTES
XMIT2:
     LDAX  D
     CPI   EOF          ;IF EOF, TRANSMISSION IS FINISHED
     JZ    XMIT3
     CPI   XOFF         ;IF TEMPORARY EOF, MORE DISK FILE REMAINS
     JZ    XMIT4
     CPI   CR           ;CLOSE OUT LINE AT CARRIAGE RETURN
     JZ    ENDLN
```

```
     CPI   LF          ;FILTER OUT LINEFEECS
     JZ    SKIP
     CPI   09H         ;CHANGE TAB CHAR TO "    "
     CZ    CHNG1
     MOV   B,    A
     CALL  SEND
     DCR   C
     JZ    ENDLN2            ;IF 132 CHARS EXCEEDED, CMS BUFFER CHCKES
SKIP:
     INX   D
     CALL  BREAK3
     JMP   XMIT2
XMIT3:
     LXI   D,    MSG7 ;PROMPTS "TRANSMISSION COMPLETE"
     CALL  MESSAGE
XMIT35:
     CALL  PAUSE
     MVI   A,    XOFF ;SENDS DOUBLE XOFF TO SHIFT
     CALL  SEND       ;CMS FROM INPUT TO EDIT MODE
     CALL  ANS2       ;WAIT FOR ANSWER AND DELAY IN BETWEEN
     CALL  PAUSE
     MVI   A,    XOFF
     CALL  SEND
     RET
XMIT4:           ;FOR FILES EXCEEDING 52K, PROGRAM SHIFTS
          ;CMS TO EDIT MODE AND ISSUES "SAVE" COMMAND
          ;AT THIS POINT - CMS SAVES TRANSMITTED DATA
          ;AND RETURNS TO INPUT MODE, AT WHICH TIME
          ;PROGRAM READS NEXT SECTION OF FILE AND TRANSMITS
     CALL  XMIT35
     CALL  ANS
     CALL  PAUSE
     LXI   D,    MSG19
XMIT5:
     LDAX  D
     CPI   '$'
     JZ    XMIT6
     CALL  CONOUT
     CALL  SEND
     INX   D
     JMP   XMIT5
XMIT6:
     MVI   A,    XOFF
     CALL  SEND
     CALL  ANS
     LXI   D,    MSG18       ;PROMPTS "RELOADING"
     CALL  MESSAGE
     CALL  FILERDO           ;READ NEXT PART OF FILE FROM DISK
```

```
     JMP   XMIT        ;TRANSMIT NEXT PART OF FILE
BREAK3:
     IN    0F7H
     ANI   2
     RZ
     IN    0F6H
     ANI   7FH
     CPI   CNTLD
     RNZ
     JMP   DIRECT
ENDLN:           ;SENDS XOFF AFTER EACH LINE
     CMP   B           ;IF LAST CHAR SENT WAS A CR, IGNORE-
     JZ    SKIP        ;EFFECTIVELY CANCELS SKIPPED LINES
ENDLN2:
     MOV   B,    A
     MVI   A,    XOFF
     CALL  SEND
     CALL  ANS2
     CALL  PAUSE
     MVI   C,    83H         ;132 BYTES
     JMP   SKIP        ;CONTINUE TRANSMITTING
PAUSE:           ;DELAY APPROX 100 MICROSECONDS
     LXI   H,    200H
PAUSE2:
     DCX   H
     MOV   A,    H
     CPI   0
     JNZ   PAUSE2
     RET
FILE:            ;COMMANDS CMS TO "FILE" TRANSMITTED DATA
     CALL  PAUSE
     LXI   D,    MSG8
FILE2:
     LDAX  D
     CPI   '$'
     JZ    FILE3
     CALL  CONOUT
     CALL  SEND
     INX   D
     JMP   FILE2
FILE3:
     MVI   A,    XOFF
     CALL  SEND
     RET
     BUFF40:   DS   20
               DS   2
     BUFF  EQU  $      ;BUFFER STARTS AT END OF PROGRAM
     END   100H
```

```
C                                                                       POS00010
C     RANDOMIZED MIL SPEC 8866 SPECTRUM A LOADS                         POS0003C
C
      INTEGER*2 KONST                                                   POS00040
      INTEGER*2 KSTEP1                                                  POS00050
      INTEGER*2 KSTEP2                                                  POS00060
      INTEGER*2 KSTEP3                                                  POS00070
      DIMENSION SMAX(4205),SMIN(4205),VOLTS1(4205),VOLTS2(4205),        POS00080
     1KSTEP1(4205),KSTEP2(4205),KSTEP3(8410)                            POS00090

C     INPUT DATA

      READ(5,32)NBLOCK,JLEVEL,SCALE,TLL,AREA                            POS00100
   32 FORMAT(2I10,2F12.0,F12.3)                                         POS00110
      WRITE(5,34)NBLOCK,JLEVEL                                          POS00120
   34 FORMAT(//I10,23H TIMES THROUGH BLOCK OF,I10,6H LOADS)             POS00130
      WRITE(5,33)SCALE,TLL,AREA                                         POS00140
   33 FORMAT(//10X,18H SCALE FACTOR     =,F12.0,1X,6H (PSI),//10X,18H LIMPOS00150
     1IT LOAD       =,F12.0,1X,6H (PSI),//10X,18H CROSS SECT AREA =,F12.3POS00160
     1,1X,13H (SQR INCHES))                                             POS00170

C     INITIALIZE PARAMETERS

   61C IA=0                                                             POS00180
       IB=0                                                             POS00190
       IC=0                                                             POS00200
       ID=0                                                             POS00210
       IE=0                                                             POS00220
       IK=0                                                             POS00230
       IG=0                                                             POS00240
       IH=0                                                             POS00250
       II=0                                                             POS00260
       IJ=0                                                             POS00270
       I=1                                                              POS00280
       ITOTAL=0                                                         POS00290
       KONST=32768                                                      POS00300
       PLL=TLL*AREA                                                     POS00310

C     BEGIN RANDOMIZATION PROCESS

       IX=583                                                           POS00320
   700 CALL RANDU(IX,IY,YFL)                                            POS00330
       IX=IY                                                            POS00340
       IF(ITOTAL.EQ.4201) GO TO 790                                     POS00350
       IF((YFL.GE.0.).AND.(YFL.LT..1)) GO TO 750                        POS00360
       IF((YFL.GE..1).AND.(YFL.LT..2)) GO TO 751                        POS00370
       IF((YFL.GE..2).AND.(YFL.LT..3)) GO TO 752                        POS00380
```

```
      IF((YFL.GE..3).AND.(YFL.LT..4)) GO TO 753                         POS00390
      IF((YFL.GE..4).AND.(YFL.LT..5)) GO TO 754                         POS00400
      IF((YFL.GE..5).AND.(YFL.LT..6)) GO TO 755                         POS00410
      IF((YFL.GE..6).AND.(YFL.LT..7)) GO TO 756                         POS00420
      IF((YFL.GE..7).AND.(YFL.LT..8)) GO TO 757                         POS00430
      IF((YFL.GE..8).AND.(YFL.LT..9)) GO TO 758                         POS00440
      IF((YFL.GE..9).AND.(YFL.LT.1.)) GO TO 759                         POS00450
  750 IF(IA.EQ.2) GO TO 700                                             POS00460
      ITOTAL=ITOTAL+1                                                   POS00470
      IA=IA+1                                                           POS00480
      SMAX(I)=1.25*PLL                                                  POS00490
      SMIN(I)=.11*PLL                                                   POS00500
      GO TO 785                                                         POS00510
  751 IF(IB.EQ.4) GO TO 700                                             POS00520
      ITOTAL=ITOTAL+1                                                   POS00530
      IB=IB+1                                                           POS00540
      SMAX(I)=1.15*PLL                                                  POS00550
      SMIN(I)=.11*PLL                                                   POS00560
      GO TO 785                                                         POS00570
  752 IF(IC.EQ.15) GO TO 700                                            POS00580
      ITOTAL=ITOTAL+1                                                   POS00590
      IC=IC+1                                                           POS00600
      SMAX(I)=1.05*PLL                                                  POS00610
      SMIN(I)=.11*PLL                                                   POS00620
      GO TO 785                                                         POS00630
  753 IF(ID.EQ.44) GO TO 700                                            POS00640
      ITOTAL=ITOTAL+1                                                   POS00650
      ID=ID+1                                                           POS00660
      SMAX(I)=.95*PLL                                                   POS00670
      SMIN(I)=.11*PLL                                                   POS00680
      GO TO 785                                                         POS00690
  754 IF(IE.EQ.136) GO TO 700                                           POS00700
      ITOTAL=ITOTAL+1                                                   POS00710
      IE=IE+1                                                           POS00720
      SMAX(I)=.85*PLL                                                   POS00730
      SMIN(I)=.11*PLL                                                   POS00740
      GO TO 785                                                         POS00750
  755 IF(IK.EQ.250) GO TO 700                                           POS00760
      ITOTAL=ITOTAL+1                                                   POS00770
      IK=IK+1                                                           POS00780
      SMAX(I)=.75*PLL                                                   POS00790
      SMIN(I)=.11*PLL                                                   POS00800
      GO TO 785                                                         POS00810
  756 IF(IG.EQ.450) GO TO 700                                           POS00820
      ITOTAL=ITOTAL+1                                                   POS00830
      IG=IG+1                                                           POS00840
      SMAX(I)=.65*PLL                                                   POS00850
      SMIN(I)=.11*PLL                                                   POS00860
```

```
      GC TC 785                                                          POS00870
757   IF(IH.EQ.650) GO TC 700                                            POS00880
      ITCTAL=ITCTAL+1                                                    POSC089C
      IH=IH+1                                                            POS00900
      SMAX(I)=.55*PLL                                                    POS00910
      SMIN(I)=.11*PLL                                                    POSC092C
      GO TO 785                                                          POS00930
758   IF(II.EQ.950) GO TO 700                                            PCS0094C
      ITOTAL=ITOTAL+1                                                    POS00950
      II=II+1                                                            POSC0960
      SMAX(I)=.45*PLL                                                    POS0097C
      SMIN(I)=.11*PLL                                                    POS00980
      GO TO 785                                                          POSCC99C
759   IF(IJ.EC.1700) GO TO 700                                           POS0100C
      ITOTAL=ITOTAL+1                                                    POS0101C
      IJ=IJ+1                                                            POS01020
      SMAX(I)=.35*PLL                                                    POS01C3C
      SMIN(I)=.11*PLL                                                    POS01040
      GC TO 785                                                          POS0105C
 785  CONTINUE                                                           POS01060

C     FORM TWO VECTORS (PCSITIVE LOADS AND 11 PERCENT LCADS

      VCLTS1(I)=(SMAX(I)/SCALE)*10.                                      POS01070
      KSTEP1(I)=(INT(VOLTS1(I)*(204.8)))*16+KONST                        POSC108C
      VCLTS2(I)=(SMIN(I)/SCALE)*10                                       POS01090
      KSTEP2(I)=(INT(VOLTS2(I)*(204.8)))*16+KONST                        POS01100
      I=I+1                                                              POS0111C
      GC TO 700                                                          POS01120
790   CONTINUE                                                           POS0113C

C     ALTERNATE PCSITIVE LOADS WITH 11 PERCENT LOADS

      K=0                                                                POS0114C
      J=-1                                                               POS01150
      DC 99 I=1,4205                                                     POS01160
      J=J+2                                                              POS01170
      K=K+2                                                              POSC1180
      KSTEP3(J)=KSTEP1(I)                                                POS01190
      KSTEP3(K)=KSTEP2(I)                                                POS01200
 99   CONTINUE                                                           POS01210

WRITE RESULTANT VECTOR AS DEFINED BY DATA DEFINITION

      WRITE(9,120)(KSTEP3(I),I=1,8410)                                   POS0122C
      REWIND 9                                                           POS01230
 120  FORMAT(255A2,1A2)                                                  POS0124C
      STCP                                                               POS01250
```

LIST OF REFERENCES

1. Intel Corporation, System 80/10 Hardware Reference Manual, 1976.

2. Intel Corporation, SBC 732 Combination Analog Input/Output Board Hardware Reference Manual, 1977.

3. Atkinson, J.S., Jr., A Study of Spectrum Loading and Range-Pair Counting Method Effects on Cumulative Fatigue Damage, Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1977.

4. IBM, Control Program-67/Cambridge Monitor System (CP-67/CMS) Version 3.2 Program Number 360 D-05.2.005 User's Guide, 1974.

5. W. R. Church Computer Center, User's Manual, 1974.

6. Intel Corporation, MDS-800 Intellec MDS Microcomputer Development System Hardware Reference Manual, 1975.

7. Intel Corporation, Diskette Operating System Microcomputer Development System MDS-DOS Operator's Manual, 1975.

8. Elliott, M.T., Master's Thesis to be published September 1978, Naval Postgraduate School, Monterey, CA.

9. Digital Research Corp., An Introduction to CP/M Features and Facilities, 1976.

10. Butler, C.L., Software Design for a Fatigue Monitoring Data Acquisition System, Master's Thesis, Naval Postgraduate School, Monterey, CA, September 1976.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 67 Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
4.	Assoc. Professor G. H. Lindsey, Code 67Li Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
5.	LCDR Frederick Martin Blakely, USN 14834 North Ashdale Avenue Woodbridge, VA 22193	1