

AD-A092 441

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
DIAGNOSTIC SOFTWARE DEVELOPMENT FOR A REAL-TIME SPECTRAL ANALYS--ETC(U)
APR 80 B J DONLAN
AFIT-CI-80-9T

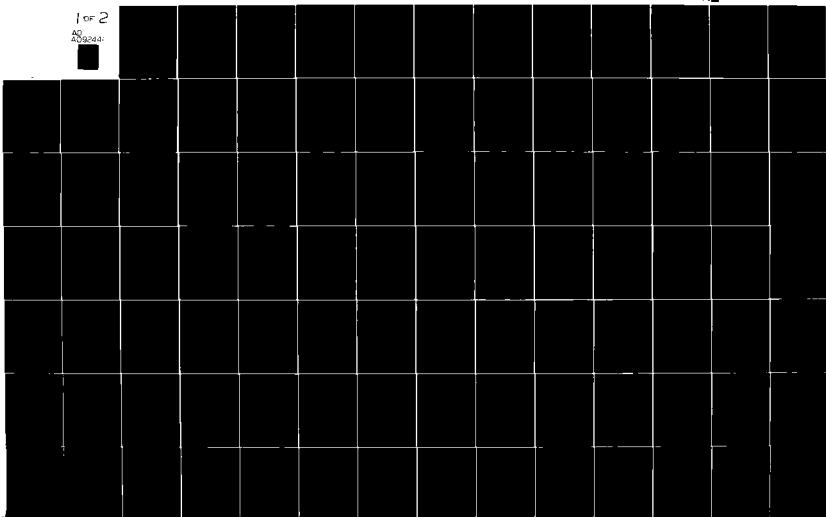
F/G 14/2

NL

UNCLASSIFIED

1 of 2

AD
A092441



AD A092441

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LEVEL II

①

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 80-9T	2. GOVT ACCESSION NO. AD-A092 441	3. RECIPIENT'S CATAL. NUMBER
4. TITLE (and Subtitle) Diagnostic Software Development For A Real-Time Spectral Analysis System		5. TYPE OF REPORT & PERIOD COVERED THESIS/DISSERTATION
6. PERFORMING ORG. REPORT NUMBER		7. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: Rensselaer Polytechnic Institute		10. PROGRAM ELEMENT PROJECT TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE Apr 80 1980
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		13. NUMBER OF PAGES 105
15. SECURITY CLASS. (of this report) UNCLASS		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) E		
18. SUPPLEMENTARY NOTES APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17 Fredric C. Lynch FREDRIC C. LYNCH, Major, USAF Director of Public Affairs Air Force Institute of Technology (ATC) Wright-Patterson AFB, OH 45433		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED		

DTIC

DEC 1 1980

E

DDC FILE COPY

DD FORM 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

This report deals with the development of diagnostic software for a real-time spectral analysis system. The report gives a description of the spectral analysis system and its associated maintainability and reliability problems. The diagnostic system is comprised of a read-only memory board containing a small operating system and diagnostic tests. Detailed descriptions and listings of the tests and operating system are provided.

Application For

CLASS - COMM

STATUS - TIB

REASON FOR CLASS

DATE OF REVIEW

APPROVAL

SPECIAL AGENT

A

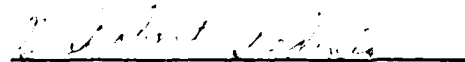
DIAGNOSTIC SOFTWARE DEVELOPMENT FOR A
REAL-TIME SPECTRAL ANALYSIS SYSTEM

by

Brian J. Donlan

A Project Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Approved by:



G. Robert Redinbo, Advisor

Rensselaer Polytechnic Institute
Troy, New York

April 1980

CONTENTS

	Page
LIST OF TABLES	vi
LIST OF FIGURES	vii
ACKNOWLEDGEMENT	viii
ABSTRACT	ix
1. INTRODUCTION-HISTORY	1
2. SYSTEM DESCRIPTION	3
2.1 General Description-Data Flow	3
2.2 IMSAI Microcomputer	3
2.3 Floating Point AP-120B Array Processor	4
2.4 Data Acquisition	5
2.5 Color Graphics Display	6
3. TECHNICAL DISCUSSION	8
3.1 Task	8
3.2 Design Details	8
3.3 Test Procedures	10
4. DIAGNOSTIC OPERATING SYSTEM	12
4.1 General Description	12
4.2 Detailed Description	12
4.3 Program Usage	12
4.4 Commands	13
4.5 Externally Referenced Subroutines	21
5. MINI-MEMORY TEST	24
5.1 General Description	24
5.2 Program Details	24
5.3 Operation	25
6. COMPREHENSIVE MEMORY TEST	34
6.1 General Description	34

6.2 Program Details	34
6.3 Operation	34
7. FORMATTED DISK TEST	42
7.1 General Description	42
7.2 Program Details	42
8. TRACK WRITE ROUTINE	53
8.1 General Description	53
8.2 Program Details	53
8.3 Operation	54
9. TRACK READ ROUTINE	62
9.1 General Description	62
9.2 Program Details	62
9.3 Operation	63
10. UNIBUS PORT TEST.....	70
10.1 General Description	70
10.2 Program Details	70
10.3 Operation	71
11. UNIBUS COMMUNICATION TEST	76
11.1 General Description	76
11.2 Program Details	76
12. UNIBUS SNAPSHOT	86
12.1 General Description	86
12.2 Program Details	86
12.3 Operation	86
13. DIAGNOSTIC MEMORY BOARD	92
13.1 General Description	92
13.2 Detailed Description	92
14. CONCLUSIONS	94

15. LITERATURE	96
APPENDIX A: Memory Board Diagrams	97
APPENDIX B: Commonly Used Subroutines	101
APPENDIX C: PROM Program Listings	105

LIST OF TABLES

	Page
Table 4.1 Command Summary	
Table 5.1 Phase Messages	
Table 11.1 Unibus Addresses	
Table 13.1 Memory Map	

LIST OF FIGURES

	Page
Figure 2.1 System Block Diagram	
Figure 10.1 Unibus Adapter Ports	

ACKNOWLEDGEMENT

The author wishes to express his gratitude to Dr. G. Robert Redinbo and Dr. Lester A. Gerhardt for their advice and guidance.

Special thanks to the Air Force Institute of Technology for their academic sponsorship.

Finally to Corey and Joey for taking many naps and allowing their mother to type the manuscript.

Section 1

Introduction-History

Over the past few years the researchers here at Rensselaer have been designing and constructing a Real-Time Spectral Analysis System. This system was designed to be used by oceanographers in their studies of the tides and tidal erosion. The primary purpose of the system is to measure the very low frequency power spectra of the ocean waves. The power spectral density is obtained by taking the time varying input signals from the oceanographer's sensors and transforming it from the time domain to the frequency domain using a Fast Fourier Transform. The resulting spectral information is presented on a color graphics display.

In the summer of 1978 the newly completed system was taken to a beach in Florida for a field test. The test proved partially unsuccessful. The basic system ideas and functions were correct, but the system proved difficult to test and keep operational. Reliability and maintainability became the system's downfall.

In the haste to complete the system for the summer tests very little was done to facilitate testing of fault diagnosis of the system or individual components. The author accepted the task of remedying the situation by developing some diagnostic programs and methods which will aid in test and diagnosis of the Real-Time Spectral Analysis System.

This report presents the author's efforts and the diagnostic programs developed to solve the given problem.

Since no tests or test procedures existed, the entire diagnostic system was the author's responsibility.

Section 2

System Description

2.1 General Description-Data flow

The real-time spectral analysis system is composed of 4 major subsystems as presented in figure 2.1.

Analog data from up to four sensors is input to the data acquisition subsystem where it is amplified, filtered and digitized. Next, the digitized input data is transferred to the high speed array processor where a Discrete Fourier Transform is performed on the data string converting it from the time domain to the frequency domain.

The frequency domain spectral data is transferred to the color graphics display where it is displayed in a time verses frequency format with color encoding representing amplitude

Detailed descriptions of each subsystem and its functions follow.

2.2 IMSAI Microcomputer and Front Panel

The IMSAI Microcomputer containing an Intel 8080 CPU is the heart of the system. The IMSAI thru the Unibus adapter controls many of the systems and options. The 8080 operates the user front panel, receiving and displaying the many parameters and options, and passes them out to the required subsystem. The microcomputer also functions as a host for the Floating Point Array Processor, which has the

task of loading the signal processing programs into the array processor.

In its present configuration, the IMSAI system contains 24K bytes of semiconductor random access memory, an 8" floppy disk drive and interface, a serial input/output port for console communication, and two parallel input/output boards used in the Unibus adapter. The IMSAI also hosts a small numerical processor. With this project, an additional 16K byte read-only memory board was added to the IMSAI to contain the diagnostic tests. The 8080 also contains an adapter board which enables it to simulate a DEC PDP-11 computer and perform data transfers over the Unibus.

2.3 Floating Point AP-1208 Array Processor

The Floating Point Array Processor performs the actual signal processing. The digitized input data is transferred to the array processor where it is first multiplied by a user selected window. The windowing helps minimize any distortion caused by the time limiting of the input data. The data is then transformed into the frequency domain using a 1024 point Fast Fourier Transform (FFT). After the Fast Fourier Transform (FFT) the spectral data may be filtered or further processed before being presented to the color graphics display.

The AP-120B is a very fast and versatile floating point array processor with a basic cycle time of 167 nanoseconds. The AP-120B has a pipeline structure with a 38-bit

floating point format. The instruction words are 64 bits long so many functions can be performed in one machine cycle. This type of speed and instructions format is ideal for the reiterative additions and multiplication often used in digital signal processing. For example, a 512 point FFT can be performed in 3.2 MSEC.

The array processor is a slave computer and requires a host to operate it. In this system the IMSAI 8080 functions as the Host. The AP has no front panel and all access to the AP is thru a Unibus interface.

2.4 Data Acquisition

The present system has four analog input channels. The amplified analog inputs are filtered by a programmable anti-aliasing filter to band limit the input high frequency components. The input signal high frequencies must be limited to prevent aliasing. The filter cutoff frequency is set by the IMSAI 8080 depending on the sampling frequency selected on the user front panel. The analog signal is then sampled by an A/D converter which converts the continuous input data to sampled digital data. The system is capable of sampling frequencies of from .001 HZ to 15 KHZ. The input circuitry contains a programmable real-time clock used to control the sampling rate. This clock is set by the IMSAI in response to the front panel selected frequency. The digitized data is transferred directly to the array processor data memory via an AP input/output port interface.

2.5 Color Graphics Display

The color graphics display system was built especially for this system. The graphics display portion consists of a 512 x 512 point screen format which is stored in a large Intel refresh memory.

The frequency spectrum of each input block of 1024 input samples is displayed on one horizontal line with frequency increasing from left to right. A time history of the spectrum is shown in the vertical direction as each new line is added. As each FFT computation is completed in the array processor its spectrum is output into one horizontal line on the display. The new line is added to the bottom of the screen and the older lines are scrolled up presenting the history of the spectrum. Each point of the line is color amplitude encoded with one of 128 possible colors.

Two lines of characters for anotation purposes are provided at the top of the screen. Under the two lines of anotation are two color bars used in the color amplitude encoding. The top most bar is called the menu and it presents all 128 possible colors. The lower bar is the color map which contains 64 locations into which a color can be loaded. Each map position represents an increment of magnitude of the frequency spectrum. The color placed in each map location representing a certain level of magnitude is selected on the user front panel. The entire map may be stored on the floppy disk for quick reloading.

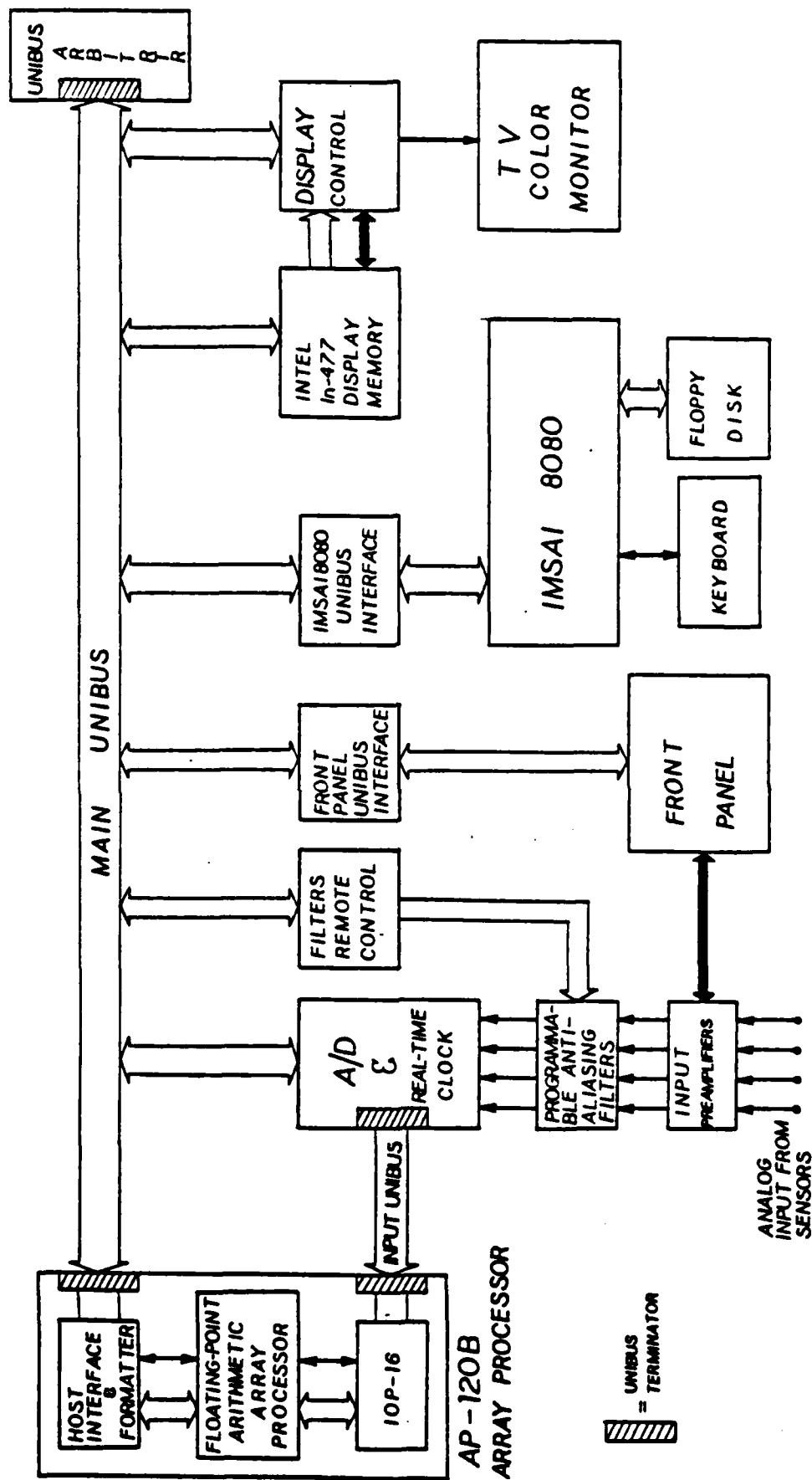


FIG 2.1 SYSTEM BLOCK DIAGRAM

Section 3

Technical Discussion

3.1 Task

The purpose of this project is to identify potential system malfunctions and to design software and hardware aids which will help in isolating system faults. History has helped in locating many problem areas. The spectral analysis system has a number of weak hardware links which are prone to failure. These areas received attention first.

3.2 Design Details

The floppy disk system has been a consistent problem area. The disk drive is a very delicate device which was never mounted or placed in a case, leaving it exposed to contamination and physical abuse.

In the original system the disk was the keystone of the spectrum analysis system. All memory was volatile and once powered-down left the system void of intelligence. The only mass storage device from which to load any programs or diagnostics was the disk drive, making a disk failure catastrophic. A programmable read-only memory containing a small operating system and diagnostics, was added to the IMSAI 8080 computer. This enables the computer system to always have a small operating system available, even in the event of a disk failure.

The floppy disk is probaby one of the most difficult parts of the system to troubleshoot. The disk controller requires many driver routines to read or write to the disk drive. A large disk diagnostic was written and stored on the diagnostic memory board to aid in troubleshooting and testing the disk system. Two smaller programs were also written to aid in disk drive alignment and testing.

The IMSAI 8080 computer is another area where a possible failure leads to complete system failure. The IMSAI uses an Intel 8080 microprocessor as CPU. Because of the one chip simplicity and cheap replacement, diagnostics on the actual CPU instuctions did not seem prudent. Diagnostics to test the computer memory were written however. The computer system presently has 24K of MOS RAM memory. The MOS memory is very sensitive to static charges and power supply voltage variations. A comprehensive memory test using an advanced test algorithm by Knaizut and Hartmann [2] was implemented. A simpler mini-memory test has the advantage of not requiring a console device or scratch pad memory. These tests are all resident on the diagnostic memory board.

The majority of the devices in the spectral analysis system are handwired prototype devices connected to the Pseudo-Unibus. The IMSAI microcomputer has control of the Unibus via a Unibus to S-100 bus adapter. The adapter

board uses 6 I/O ports in the IMSAI 8080. Since problems with shorted Unibus lines have been common in the past, a diagnostic was written specifically to test the I/O ports and Unibus Lines.

In order to communicate with the many devices on the Unibus, a Unibus communication test was written. This is probably the most useful of all tests written; the user need only input the device address and a transfer command. A data word can be transfered to or from any device on the Unibus. Timers are also included in the program to test for device 'no answers', a common problem with Unibus devices.

The last major component of the system is the Floating-Point array processor. At present, a complete diagnostic package for the AP exists. This package was delivered with the AP and runs on the DEC PDP-11. The spectral analysis system also has a small AP debug program as part of the real-time program. No further diagnostics were written for the AP. The Unibus communication test can be used to read and write to the array processor front panel.

3.3 Test Procedures

Since the heart of the system is the IMSAI 8080 computer, no part of the system will function without it, making it the first area to be treated.

The following is a recommended test sequence:

1. Mini-Memory test 0 to 100 hex version — this tests the scratch pad memory used by other tests and the diagnostic operating system. This test needs no console. (sec.5)
2. Mini-Memory Test 24K version — this provides a quick test of all memory. Don't forget to check the power supplies if you have problems. (sec.5).
3. Diagnostic Operating System — try some simple command for overall CPU operation. This also tests the console for overall CPU operation. This also tests the console I/O operation. (sec.4)
4. Comprehensive Memory Test — this provides a good test on the memory and should catch most problems. (sec.6)
5. CPM disk operating system — boot the disk system and see what happens. This provides a good indication of overall disk operation. Most disk problems show up here.
6. Formatted disk test — this test tests the normal formatted operation of the disk. This test can take a long time to check all 77 tracks and a bad diskette media can cause a failure; use a good blank disk.
7. Unibus Port Test — This checks for shorted unibus lines and broken wires. If this fails look for an incorrectly inserted card in the lower card cage. (sec.10)
8. Unibus Communication Test — try to communicate with the various unibus devices. (sec.11)
9. Real-time Spectral Analysis program — the ultimate test.

Section 4

Diagnostic Operating System

4.1 General Description

In order to enable the computer system to have some capabilities and intelligence during a major system failure, a small self-contained operating system is included on the diagnostic memory board. This operating system can be used to run the diagnostic program as well as perform a number of standalone function such as memory and register examine.

4.2 Detailed Description

The diagnostic operating system is a modified and expanded version of the SSM 8080 Monitor V-1 supplied with the Prom Memory board. The operating system supplied was modified to handle our I/O requirements and the scratch pad area used by the operating system was fixed to the first 256 bytes of RAM memory. A diagnostic test directory and controller were added to ease diagnostic program execution. A Help command was also added to remind the casual user of the various commands and options available.

4.3 Program Usage

The starting address of the operating system is F000 Hex and it can be started directly from the IMSAI 8080 front panel. The operating system is located in a

readonly memory on the diagnostic memory board and requires only a minimum system of 256 byte of ram memory and a console device to function.

The operator communication with the monitor consist of a single alphabetic character input on the console device and may be followed by one or more parameters. When two or more parameters are used they are separated by a coma or a space. Parameters are hexadecimal values consisting of four or two hexadecimal characters. Leading zeros are assumed. The command line is terminated by carriage return in most cases.

4.4 Commands

The following is a modified and expanded explanation of the SSM monitor commands

D Command- (Display memory)

D'Low Address', 'High Address'

Memory from 'Low Address' through 'High Address' is displayed on the console device. If 'High Address' is equal to or smaller than 'Low Address', only the 'Low Address' byte is displayed. Data bytes are displayed in hexadecimal, 16 bytes per line. The beginning address of each line is displayed.

S Command - (Subsitute memory)

S'Address'

The byte at location 'Address' is displayed on the console device followed by a - character. The operator responds with one or more characters from the console. If the input character is a space or comma, the contents of the next location is displayed. If one or more hexadecimal digits are inputted before the space or comma, the specified value will replace the displaced value in the memory location. A carriage return terminates the command.

F Command (fill memory)

F'Low Address',High Address','Data'

Memory from 'Low Address' through 'High Address' is filled with 'data'. If 'High Address' is equal to or smaller than 'Low Address' on the 'Low Address' is changed.

M Command (move memory)

M'Low Address',High Address','Dest Address'

Data from 'Low Address' through 'High Address' are moved to memory beginning at 'Dest Address'. If 'High Address' is equal to or smaller than 'Low Address' only the byte at 'Low Address' is moved. If 'Dest Address' is between 'Low Address' and 'High Address' the data from 'Low Address' to 'Dest Address' is repeated to fill the destination field.

B Command (binary dump memory)

B'Low Address','High Address'

Data from 'Low Address' through 'High Address'

are output to the logical punch device in a binary format compatible with mits paper tape format. 'High Address' must be equal to or greater than 'Low Address', with one exception: If 'High Address' is zero, an end-of-file-record is output specifying 'Low Address' as the entry point address

L Command (load memory, binary)

L'Bias Address'

Data in binary format are read from the logical reader device and stored in memory at the load address specified in the input file plus 'Bias Address'. When an end-of-file record is encountered control is transferred to the specified entry point, address of zero terminates loading and the monitor remains in control.

W Command (write memory, Hexadecimal)

W'Low Address','High Address'

Data from 'Low Address' through 'High Address' are output to the logical punch device in a hexadecimal format compatible with Intel paper tape format. 'High Address' must be equal to or greater than 'Low Address' with one exception: If 'High Address' is zero an end-of-file record is output specifying 'Low Address' as the entry point address.

R Command (read to memory, hexadecimal)

R'Bias Address'

Data in hexadecimal format are read from the logical reader device and stored in memory at the load address specified in the input file plus 'Bias Address'. When an end-of-file record is encountered control is transferred to the specified entry point address if it is non-zero. An entry point address of zero terminates loading and the monitor remains in control.

N Command (null output)

N

Sixty null bytes (OOH) are output to the logical punch device.

K Command (copy files)

K

Bytes are continuously read from the logical reader device and output to the logical punch device. This process continues until manually interrupted, I.E., by resetting the system.

G Command (goto)

G'Address','Breakpoint 1','Breakpoint 2'

If 'Address' is specified, control is transferred to 'Address'. If 'Address' is not specified, control is transferred to the address of the last encountered breakpoint, after program status (CPU registers and flags) is restored.

If 'Breakpoint 1' or 'Breakpoint 2' is specified, breakpoints (RST 1) replace the bytes at corresponding addresses. These addresses must contain the first byte of an instruction. If breakpoints are specified, a jump instruction is stored at location 0008H to return control to the monitor when a breakpoint, or any RST 1 instruction is executed. At this point, the monitor will save the program status and restore the bytes replaced by any known breakpoints. The program counter in the saved program status is decremented, so that program execution may be resumed with the instruction formerly replaced by the breakpoint. Monitor commands may then be used to display/modify memory or CPU registers, etc.

When the monitor is entered normally, i.e. by other than breakpoint execution, recording of existing breakpoints is destroyed. Therefore, if breakpoints are set, but not executed before the monitor is re-entered, the contents of the bytes containing those breakpoints must be manually restored.

RST 1 instructions other than known breakpoints may be used as pseudo-breakpoints, subject to certain restrictions. The jump instruction must be stored at location 0008H by previously setting a normal breakpoint. RST 1 instructions other than known breakpoints may be executed through normal program execution (RST 1 stored as part executing program) or instruction jam (interrupt).

When such a RST 1 instruction is encountered, the monitor saves the program status and resets known breakpoints. However, the program counter in the saved program status is not decremented, so program execution may be resumed at the next instruction.

X Command (register display/modify)

X'Register'

Register contents as of the last encountered breakpoint are displayed. 'Register' may be specified as A,B,C,D,E,F (flags), H,L,M (H and L combined), P (program counter) or S (stack pointer). The registers are displayed, in the above order, beginning with specified 'Register'. After each register content is displayed, the operator may change it by supplying the new value followed by a space or comma. If no new value is entered the old value is retained and the next register is displayed. The command is terminated by a carriage return, or display/modification of register S.

If 'Register' is not specified, all registers are displayed without operator intervention.

C Command (hexadecimal arithmetic)

C'Operand1','Operand2'

The sum and difference of 'Operand1' and Operand2' are displayed in hexadecimal on the console device.

A Command (assign I/O devices)

A'Logical'='Physical'

Physical device 'Physical' is assigned to logical device 'Logical'. 'Logical' may be any of the four system logical devices, I.E., console, reader, punch, or list. Only the first character of the device name is required. 'Physical' may be 0,1,2, or 3. This option is not fully implemented due to the lack of I/O devices.

H Command (Help)

H

This program lists a summary of all of these commands.

T Command (Test Controller)

T

This command executes the test controller and test directory. The test directory printout can be suppressed by raising sense switch '0'. If the type-out is not suppressed the program will list the tests available and request the test to be run. If the type-out is suppressed, the test code can be input immediately following the 'T'.

A	Assigns I/O device (physical to logical)
B	Dump memory in binary on punch device
C	Hexadecimal arithmetic
D	Display a block of memory
F	Fill a block of memory with a constant
G	Go to address and execute, optional break-pt.
H	Help, this directory
K	Copy from reader to punch
L	Load a binary tape, optional bias
M	Move a block of memory to another location
N	Outputs 60 nulls to punch
R	Loads a hex tape from reader
S	Display and changes any memory location
T	Test list and execution program
W	Dumps memory in hex on punch
X	Cpu register display and change

Table 4.1 COMMAND SUMMARY

4.5 Externally Referenced Subroutines

Several externally reference subroutines are available for program usage. These routines, their starting address, and function are outlined below:

MONTRA 'F000'

Normal entry point to the monitor

CI 'F003'

Console input. One character is read from the logical console device and returned in register A. All registers other than A and F are preserved.

RI 'F006'

Reader input. One byte is read from the logical reader device and returned in reg A. All registers other than A and F are preserved. If no byte is available from the reader, the carry flag is set upon return.

CO 'F009'

Console output. The byte in register C is output to the logical console device. All registers other than A and F are preserved.

PO 'F00C'

Punch output. The byte in register C is output to the logical punch device. All registers other than A and F are preserved.

LO 'FOOF'

List output. The byte in register C is output to the logical list device. All registers other than A and F are preserved.

CSTS 'F012'

Console status. The logical console input device is checked for availability. Register A is set to zero and the zero flag is set true if no input is available. Register A is set non-zero and the zero flag set false if a character is available. All registers other than A and F are preserved.

IOCHK 'F015'

The current setting of IOBYT (I/O byte) is returned in register A. IOBYT is the byte of ram used to record the current logical device to physical device assignments.

- | | |
|----------|--|
| Bits 0,1 | Record the physical device currently assigned to the logical console device. |
| Bits 2,3 | Record the physical device currently assigned to the logical reader device. |
| Bits 4,5 | Record the physical device currently assigned to the logical punch device. |
| Bits 6,7 | Record the physical device currently assigned to the logical list device. |

IOSET 'F018'

The contents of register C are stored in ICBYT, thus altering the logical to physical device assignments. All registers are preserved.

STRNG 'F01E'

The string of characters pointed to by registers H and L is output to the logical console device. The character string is terminated before a null character or after a character with bit 7 set. Registers B,D,E are preserved.

REENT 'F021'

Alternate entry point to the monitor. The current I/O configuration is not altered when the monitor is entered at this point.

Section 5

Mini-Memory Test

5.1 General Description

The Mini-Memory test is a small memory diagnostic test. The test is completely self-contained and requires no scratch pad memory or I/O devices. Since the test has a fixed test address range three different copies of the test are provided, each with a different address range.

5.2 Program Details

The Mini-Memory test is a modified implementation of the memory test supplied with the IMSAI 8080 computer system. To provide flexibility three different versions of the test are provided with a (0 to 256), (0 to 8K), (0 to 24K) address ranges. All versions are stored in programmable memory on the diagnostic memory board.

The memory test consists of three phases. Phase one consists of an incremented bit pattern, where each address is tested with the 256 different patterns. In phase two and three the lower and upper bytes respectively of the address are stored in that location. Phase two and three are designed to help locate addressing problems.

5.3 Operation

The 0 to 100 hex version of this test was designed to test the scratch pad area used by the diagnostic operating system and the comprehensive memory test. This test should be run before these programs to verify this area of memory. Although this test can be run by the operating system test controller, it should normally be started directly from the IMSAI 8080 front panel at a starting address of 'C290' hex.

The Mini-Memory tests require no console device so all communication with the test is through the IMSAI 8080 sense lites (address lites 8-15) and the sense switches (address switches 8-15). Once the test is started, the status of the test as it proceeds through the various phases is displayed in the sense lites (see table 5.1). If an error is encountered, an error message is also read out in the sense lites. The following is the procedure used to locate the faulty memory location:

1. Change any sense switch
2. Sense lites will display 8 high-order address bits at the failing location.
3. Change any sense switch.
4. Sense lites will display 8 low-order address bits at the failing location
5. Change any sense switch.
6. Sense lites will display data test pattern.

7. Change any sense switch.
8. Sense lites will display the actual data at the failing location.
9. Change any sense switch.
10. The test will start over at the beginning of phase one.

The 0 to 8K version of the Mini-Memory begins at 'D600' hex and the 0 to 24K version begins at 'D700' hex.

Sense Lite Display Hex	Meaning
01	Phase 1 Running-no errors yet
02	Phase 2 Running-no errors yet
03	Phase 3 Running-no errors yet
F1	Error Phase 1
F2	Error Phase 2
F3	Error Phase 3
FF	Test complete-no errors, move any sense switch to restart

Table 5.1
Phase messages

```

; MINI-MEMORY TEST
; PROM VERSION FOR 0 TO 100H
;
; BRIAN J. DONLAN
;
;
; ORG      0C290H
ENTER:  DI
        MVI      A, OFEH
        OUT      OFFH      ;OUTPUT PHASE I LITES
        LXI      H, 000H    ;START ADDRESS
LP2:    XRA      A          ;ZERO ACC
LP1:    MOV      M, A        ;STORE TEST PATTERN IN MEM.
        MOV      B, M        ;READ BACK TO B
        CMP      B          ;COMPARE FOR OK
        JNZ      ERR1       ;JUMP IF ERROR
        INR      A          ;NEW TEST PATTERN
        JNZ      LP1
        INX      H
        LXI      D, OFF00H      ;STOP ADDRESS
        XCHG
        DAD      D            ;ADD TWO'S COMPLIMENT
        XCHG
        JNC      LP2
;
; PHASE II
        MVI      A, OFDH      ;PHASE II LITES
        OUT      OFFH
        LXI      H, 000H
LP3:    MOV      M, H        ;LOW ADDRESS TO MEM
        INX      H
        LXI      D, OFF00H    ;STOP ADDRESS
        XCHG
        DAD      D
        XCHG
        JNC      LP3
; READ MEMORY
        LXI      H, 000H
LP4:    MOV      A, M        ;READ MEMORY
        SUB      H          ;COMPARE
        JNZ      ERR2       ;JUMP IF ERROR
        INX      H
        LXI      D, OFF00H
        XCHG
        DAD      D
        XCHG
        JNC      LP4
;
; PHASE III
;
        MVI      A, OFCH      ;PHASE THREE LITES
        OUT      OFFH
        LXI      H, 000H
LP5:    MOV      M, L        ;STORE HIGH ADDRESS IN ALL MEM
        INX      H
        LXI      D, OFF00H
        XCHG
        DAD      D
        XCHG
        JNC      LP5
; READ MEM
        LXI      H, 000H
LP6:    MOV      A, M        ;READ MEMORY
        SUB      L          ;COMPARE
        JNZ      ERR3
        INX      H
        LXI      D, OFF00H
        XCHG
        DAD      D
        XCHG
        JNC      LP6

```

```

;
;
; ALL PHASE COMPLETE
MVI A,OFFH
LXI H,ENTER
JMP LITES ;GO TO LITES PROG

;
; PHASE I ERROR
ERR1: XCHG
      MOV C,A ;SAVE BAD DATA
      LXI H,COMERR ;RETURN
      MVI A,OF1H ;PHASE I ERROR LITES
      JMP LITES

;
; COMMON ERROR OUTPUT ROUTINE
COMERR: MOV A,D ;HIGH ADDRESS
        LXI H,LOADD ;RETURN
        JMP LITES
LOADD: MOV A,E ;LOW ADDRESS TO LITES
        LXI H,IPAT ;RETURN
        JMP LITES
IPAT: MOV A,C ;TEST PATTERN TO LITES
      LXI H,ACTDAT ;RETURN
      JMP LITES
ACTDAT: MOV A,B ;ACTUAL DATA TO LITES
        LXI H,ENTER ;START OVER
        JMP LITES

;
; PHASE II ERROR
ERR2: XCHG ;SAVE BAD ADDRESS
      ADD D
      MOV B,A
      MOV C,D
      MVI A,OF2H ;PHASE II ERROR TO LITES
      LXI H,COMERR ;RETURN
      JMP LITES

;
; PHASE III ERROR
ERR3: XCHG ;SAVE BAD ADDRESS
      ADD E
      MOV B,A
      MOV C,E
      MVI A,OF3H ;PHASE II ERRO TO LITES
      LXI H,COMERR ;RETURN
      JMP LITES

;
; LITES ROUTINE ENTER WITH RETURN IN REG H&L
; DATA FOR LITES IN A
LITES: CMA
        OUT OFFH ;OUTPUT LITES
        SPHL ;SAVE RETURN IN SP
        IN OFFH ;READ SENSE SWITCHES
        MOV H,A ;SAVE IN H
LP7: IN OFFH ;READ SWITCHES
      XRA H ;SEE IF THEY CHANGED
      JZ LP7
      LXI H,0FC18H ;DELAY LOOP
LP8: INX H
      XRA A
      ORA H
      JNZ LP8
      LXI H,0
      DAD SP ;ZERO H
      PCHL ;MOVE RETURN BACK TO H & L
          ;RETURN

```

```

;8K MINI MEMORY TEST
;
;
; BRIAN DONLAN
; PROM VERSION
;
; ORG 0D600H
ENTER: DI
      MVI A,0FEH
      OUT OFFH
      LXI H,000H
LP2:  XRA A
LP1:  MOV M,A
      MOV B,M
      CMP B
      JNZ ERR1
      INR A
      JNZ LP1
      INX H
      LXI D,0E000H
      XCHG
      DAD D
      XCHG
      JNC LP2
;
; PHASE II
      MVI A,0FDH
      OUT OFFH
      LXI H,000H
LP3:  MOV M,H
      INX H
      LXI D,0E000H
      XCHG
      DAD D
      XCHG
      JNC LP3
;
; READ MEMORY
LP4:  LXI H,000H
      MOV A,M
      SUB H
      JNZ ERR2
      INX H
      LXI D,0E000H
      XCHG
      DAD D
      XCHG
      JNC LP4
;
; PHASE III
;
      MVI A,0FCH
      OUT OFFH
      LXI H,000H
LP5:  MOV M,L
      INX H
      LXI D,0E000H
      XCHG
      DAD D
      XCHG
      JNC LP5
;READ MEM
LP6:  LXI H,000H
      MOV A,M
      SUB L
      JNZ ERR3
      INX H
      LXI D,0E000H
      XCHG
      DAD D
      XCHG
      JNC LP6
;
; OUTPUT PHASE I LITES
; START ADDRESS
; ZERO ACC
; STORE TEST PATTERN IN MEM.
; READ BACK TO B
; COMPARE FOR OK
; JUMP IF ERROR
; NEW TEST PATTERN
;
; STOP ADDRESS
; ADD TWO'S COMPLIMENT
;
; PHASE II LITES
; LOW ADDRESS TO MEM
; STOP ADDRESS
;
; READ MEMORY
; COMPARE
; JUMP IF ERROR
;
; PHASE THREE LITES
; STORE HIGH ADDRESS IN ALL MEM
;
; READ MEMORY
; COMPARE

```

```

; ALL PHASE COMPLETE
;   MVI    A,OFFH
;   LXI    H,ENTER
;   JMP    LITES                ;GO TO LITES PROG
;
; PHASE I ERROR
ERR1:  XCHG
;       MOV    C,A                ;SAVE BAD DATA
;       LXI    H,COMERR           ;RETURN
;       MVI    A,OF1H            ;PHASE I ERROR LITES
;       JMP    LITES
;
; COMMON ERROR OUTPUT ROUTINE
COMERR: MOV    A,D                ;HIGH ADDRESS
;       LXI    H,LOADD           ;RETURN
;       JMP    LITES
LOADD:  MOV    A,E                ;LOW ADDRESS TO LITES
;       LXI    H,TPAT           ;RETURN
;       JMP    LITES
TPAT:   MOV    A,C                ;TEST PATTERN TO LITES
;       LXI    H,ACTDAT         ;RETURN
;       JMP    LITES
ACTDAT: MOV    A,B                ;ACTUAL DATA TO LITES
;       LXI    H,ENTER         ;START OVER
;       JMP    LITES
;
; PHASE II ERROR
ERR2:   XCHG                    ;SAVE BAD ADDRESS
;       ADD    D
;       MOV    B,A
;       MOV    C,D
;       MVI    A,OF2H           ;PHASE II ERROR TO LITES
;       LXI    H,COMERR         ;RETURN
;       JMP    LITES
;
; PHASE III ERROR
ERR3:   XCHG                    ;SAVE BAD ADDRESS
;       ADD    E
;       MOV    B,A
;       MOV    C,E
;       MVI    A,OF3H           ;PHASE II ERRO TO LITES
;       LXI    H,COMERR         ;RETURN
;       JMP    LITES
;
; LITES ROUTINE      ENTER WITH RETURN IN REG H&L
;                   DATA FOR LITES IN A
LITES:  CMA
;       OUT    OFFH              ;OUTPUT LITES
;       SPHL              ;SAVE RETURN IN SP
;       IN     OFFH              ;READ SENSE SWITCHES
;       MOV    H,A              ;SAVE IN H
LP7:    IN     OFFH              ;READ SWITCHES
;       XRA    H                ;SEE IF THEY CHANGED
;       JZ     LP7
LP8:    LXI    H,0FC18H          ;DELAY LOOP
;       INX    H
;       XRA    A
;       ORA    H
;       JNZ    LP8
;       LXI    H,0
;       DAD    SP
;       PCHL
;       ;ZERO H
;       ;MOVE RETURN BACK TO H & L
;       ;RETURN

```

```

; 24K MINI-MEMORY TEST
; PROM VERSION
;
; BRIAN DONLAN
ORG      0D700H
ENTER2:  DI
        MVI      A, OFEH
        OUT      OFFH
        LXI      H, 000H
LP22:    XRA      A
LP12:    MOV      M, A
        MOV      B, M
        CMP      B
        JNZ      ERR12
        INR      A
        JNZ      LP12
        INX      H
        LXI      D, 0A000H
        XCHG
        DAD      D
        XCHG
        JNC      LP22
;
; PHASE II
        MVI      A, OFDH
        OUT      OFFH
        LXI      H, 000H
LP32:    MOV      M, H
        INX      H
        LXI      D, 0A000H
        XCHG
        DAD      D
        XCHG
        JNC      LP32
; READ MEMORY
        LXI      H, 000H
LP42:    MOV      A, M
        SUB      H
        JNZ      ERR22
        INX      H
        LXI      D, 0A000H
        XCHG
        DAD      D
        XCHG
        JNC      LP42
;
; PHASE III
        MVI      A, OFCH
        OUT      OFFH
        LXI      H, 000H
LP52:    MOV      M, L
        INX      H
        LXI      D, 0A000H
        XCHG
        DAD      D
        XCHG
        JNC      LP52
; READ MEM
        LXI      H, 000H
LP62:    MOV      A, M
        SUB      L
        JNZ      ERR32
        INX      H
        LXI      D, 0A000H
        XCHG
        DAD      D
        XCHG
        JNC      LP62
; OUTPUT PHASE I LITES
; START ADDRESS
; ZERO ACC
; STORE TEST PATTERN IN MEM.
; READ BACK TO B
; COMPARE FOR OK
; JUMP IF ERROR
; NEW TEST PATTERN
; STOP ADDRESS
; ADD TWO'S COMPLIMENT
; PHASE II LITES
; LOW ADDRESS TO MEM
; STOP ADDRESS
; READ MEMORY
; COMPARE
; JUMP IF ERROR
; PHASE THREE LITES
; STORE HIGH ADDRESS IN ALL MEM
; READ MEMORY
; COMPARE

```

```

;
; ALL PHASE COMPLETE
;   MVI    A,OFFH
;   LXI    H,ENTER2
;   JMP    LITES2                ;GO TO LITES PROG
;
; PHASE I ERROR
ERR12: XCHG
;   MOV     C,A
;   LXI     H,COMER2            ;SAVE BAD DATA
;   MVI     A,OF1H              ;RETURN
;   JMP     LITES2              ;PHASE I ERROR LITES
;
; COMMON ERROR OUTPUT ROUTINE
COMER2: MOV     A,D                ;HIGH ADDRESS
;   LXI     H,LOADD2            ;RETURN
;   JMP     LITES2
LOADD2: MOV     A,E                ;LOW ADDRESS TO LITES
;   LXI     H,TPAT2            ;RETURN
;   JMP     LITES2
TPAT2:  MOV     A,C                ;TEST PATTERN TO LITES
;   LXI     H,ACTDA2
;   JMP     LITES2
ACTDA2: MOV     A,B                ;ACTUAL DATA TO LITES
;   LXI     H,ENTER2            ;START OVER
;   JMP     LITES2
;
; PHASE II ERROR
ERR22:  XCHG
;   ADD     D                    ;SAVE BAD ADDRESS
;   MOV     B,A
;   MOV     C,D
;   MVI     A,OF2H              ;PHASE II ERROR TO LITES
;   LXI     H,COMER2            ;RETURN
;   JMP     LITES2
;
; PHASE III ERROR
ERR32:  XCHG
;   ADD     E                    ;SAVE BAD ADDRESS
;   MOV     B,A
;   MOV     C,E
;   MVI     A,OF3H              ;PHASE II ERRO TO LITES
;   LXI     H,COMER2            ;RETURN
;   JMP     LITES2
;
; LITES ROUTINE
; ENTER WITH RETURN IN REG H&L
; DATA FOR LITES IN A
LITES2: CMA
;   OUT     OFFH                ;OUTPUT LITES
;   SPHL
;   IN      OFFH                ;SAVE RETURN IN SP
;   MOV     H,A                ;READ SENSE SWITCHES
LP72:   IN      OFFH            ;SAVE IN H
;   XRA     H                  ;READ SWITCHES
;   JZ      LP72               ;SEE IF THEY CHANGED
;   LXI     H,0FC18H
LP82:   INX
;   XRA     H                  ;DELAY LOOP
;   ORA     H
;   JNZ     LP82
;   LXI     H,0
;   DAD     SP
;   PCHL
;   ;ZERO H
;   ;MOVE RETURN BACK TO H & L
;   ;RETURN

```

Section 6

Comprehensive Memory Test

6.1 General Description

This memory test is a comprehensive memory diagnostic. The program will test the read, write, data hold and addressing capabilities of a block of memory between any two given locations.

6.2 Program Details

The comprehensive memory test is based on an implementation by Bock W. Lee [1] of the K and H memory test algorithm. The K and H algorithm, named after its creators, J. Knaizuk and C. Hartmann [2] uses modulo three addressing to help addressing problems which might normally be hidden.

The program uses two test patterns which are compliments of each other, the major and minor pattern. First one and then the other pattern are written in every third location and then read back as the program cycles through all memory locations. After all memory locations are tested using the major and minor patterns, a pass is complete. The patterns are then rotated and another pass through memory begins with the new test patterns.

6.3 Operation

The user is required to input the starting and stopping addresses. The test will continue to cycle through the test using the cyclic patterns until it is interrupted

by typing any console character.

The test has error messages which indicate the failing location and the test pattern which fails. End of pass messages are also given after each complete pass with one pair of test patterns.

The memory test is stored in programmable read-only memory on the diagnostic memory board. The program can be run under the diagnostic operating system test controller, or it can be started from the IMSAI 8080 front panel at a starting address of 'C000'.

The memory test is also stored on floppy disk and can be invoked by the CPM operating system under file name 'MEMTST.COM'

Note: The memory test requires the first 100 hex address for scratch pad so these addresses should not be tested using this test. The first 100 addresses should be tested using the mini-memory test.

```

; MEMORY TEST
; DISC VERSION 24 MAY 79 B. DONLAN
;
; ORG 100H
;
; WO EQU 00 ;TEST BYTE
;
ENTRY1: LXI H,0FOH
        SPHL
ENTRY:  LXI H,ENTRY
        PUSH H
        MVI A,00 ;ZERO ACC
        STA CODE
        CALL CRLF
        LXI H,MSG1
        CALL PMSG
        LXI H,MSG2
        CALL PMSG
        CALL BBIN
        XCHG
        SHLD START
        LXI H,MSG3
        CALL PMSG
        CALL BBIN
        XCHG
        SHLD ENADR
        LXI H,MSG8
        CALL PMSG
        IN CDATA ;RESET IO FLAG
;
;
BEGIN:  MVI C,W0 ;LOAD TEST BYTE
MTEST: MVI A,02 ;LOAD TEST BYTE
        STA PART
MTLOP:  CALL STUFF ;STUFF MAJOR ALL OVER
        MVI A,02 ;SET TWO AS MINOR
        CALL STUFM ;STUFF MINOR
        MVI A,02 ; SET 2 AGAIN
        CALL CHECK ;NOW CHECK ALL LOC
PTCHK:  LDA PART
        DCR A
        STA PART ;STORE NEW PART
        CPI 00 ;FINISH THIS PASS ?
        JZ RECYCLE ;YES
CONT:   MVI A,01 ;NO CONTINUE
        CALL STUFM ;STUFF MINOR SERT
        MOV A,C ;LOAD MAJOR BYTE
        CMA ;COMPLIMENT MAJOR BYTE
        MOV C,A ;SAVE NEW BYTE
        XRA A ;ZERO OTHER TEST BYTE
        CALL CHECK
        JMP MTLOP
;
;
RECYCLE:
        MOV A,C
        STA PART ;SAVE INVERT TB TEMP
        LXI H,MSG4 ;END OF PAS MESSAGE
        CALL PMSG
        LDA CODE ;CHAR CODE
        ORA A ;SET FLAGS
        JNZ ENTRY ;START OVER
        ANA A ;CLEAR CARRY
        LDA PART ;RECOVER TEST BYTE
        ORA A
        JZ BEGIN
        RAL
        CMA
TB:     MOV C,A ;NEW TEST BYTE
        JMP MTEST ;ANOTHER PAS

```

```

;
START: DS      2                ;LOC FOR START ADDR
ENADR: DS      2                ;LOC FOR END ADDRESS
PART:  DS      1
CODE:   DS      1
;
;
STUFF: CALL     STASTO          ;LOAD START AND END ADDR
DOIT:  MOV      M,C             ;STUFF MAJOR ALL OVER
      CALL     HILOX           ;SEE IF ALL MEM DONE
      JMP      DOIT            ;NO KEPP ON STUFFING
;
;
STJFM: CALL     STASTO          ;LOAD ADDR AGAIN
      MOV      B,A             ;MINOR COUNTER
      CPI      00              ;MINOR WORD STUFF
      JNZ      HIL             ;NO
MINOR: MOV      A,C             ;MAJOR TEST BYTE
      CMA                      ;MINOR IS COMPLIMENT OF MAJOR
      MOV      M,A             ;STUFF MINOR BYTE IN MEM
      MVI      B,03            ;START MINOR COUNT AT 3
HIL:   CALL     HILOX           ;INC & CHK IF DONE
      DCR      B               ;DEC MINOR COUNTER
      JNZ      HIL             ;OK TO STUFF NO
      JMP      MINOR           ;YES
;
;
CHECK: CALL     STASTO          ;LOAD START AND END
      MOV      B,A             ;LOAD MINOR COUNT
      CPI      00              ;COUNT ZERO
      JNZ      MAJR            ;NO GO TO MAJOR
MINR:  MOV      A,C             ;LOAD TEST BYTE MAJOR
      CMA                      ;MINOR IS COMPLIMENTYT
      CMP      M               ;READ AND COMPARE MEM LOC
      MVI      B,03            ;MINOR COUNT AT 3
      JMP      CKEND           ;CHECK FOR ERROR OR ABORT
MAJR:  MOV      A,C             ;LOAD MAJOR TEST BYTE
      CMP      M               ;READ AND COMPARE MEM WITH MAJOR
CKEND: PUSH     B               ;SAVE COUNT AND MAJOR
      CNZ      ERR             ;GO TO ERR TO PRNT IF ERROR
      POP      B               ;RESTORE REGS
      IN       CSTAT           ;CHECK KEYBOARD
      ANI      02H
      JZ       FIN
      IN       CDATA           ;READ KEYS
      STA      CODE
FIN:   CALL     HILOX           ;DEC MINOR COUNT
      DCR      B
      JNZ      MAJR
      JMP      MINR            ;COUNT ZERO DO MINOR
;
;
STASTO: LHLD     ENADR           ;LOAD END ADDR
      XCHG
      LHLD     START           ;LOAD START
      RET
;
;
ERR:   PUSH     D               ;SAVE END ADDR
      PUSH     PSW
      CALL     CRLF
      MOV      D,H
      MOV      E,L
      CALL     BINB            ;OUTPUT BAD ADDR
      MVI      D,08            ;SPACE COUNT
      CALL     BLNK            ;SPACE OVER 8
      POP      PSW
      MOV      B,A
      CALL     BITS            ;PRINT TEST BYTE
      MVI      D,0AH
      CALL     BLNK
      MOV      A,M
      CALL     BITS            ;PRINT BAD BYTE
      MOV      A,B
      POP      D               ;MOVE TEST BYTE BACK
      RET                     ;RESTORE END ADDR

```

```

HILOX:  PUSH    A            ;SAVE ACC
        INX     H            ;INC CURRENT ADDR
        MOV     A,H          ;LOAD HIGH ORDER ADDR
        CMP     D            ;COMPARE WITH END
        JNZ     DIFF         ;NO MATCH
        MOV     A,L          ;LOAD LOW ORDER
        CMP     E            ;COMPARE LOW ORDERS
        JNZ     DIFF         ;NO MATCH
        POP     A            ;MATCH END
        INX     SP           ;FAKE RETURN ONE LEVEL OUT
        INX     SP
        RET

DIFF:    POP     A            ;CONTINUE STUFFING
        RET

PROB:    MVI     C,3FH        ; ?
        CALL    CONOT        ;PRINT ?
        JMP     ENTRY

;
;
MSG1     DB      0DH,0AH,'MEMORY TEST',0
MSG2     DB      0DH,0AH,'ENTER START ADDRESS',0
MSG3     DB      0DH,0AH,'ENTER STOP ADDRESS',0
MSG4     DB      0DH,0AH,'END OF PASS',0AH,0
; diagnostic input output routines
; for brian donlan 26 feb 79

CSTAT    EQU     3            ;CONSOLE STATUS PORT.
CCOM      EQU     3            ;CONSOLE COMMAND PORT.
CDATA     EQU     2            ;CONSOLE DATA PORT.
CKBR      EQU     00000010B    ;KEYBOARD READY BIT.
CPTR      EQU     00000001B    ;PRINT READY BIT.
CNULL     EQU     1            ;CONSOLE NULL COUNT.

; CHECK CONSOLE INPUT STATUS.
;
CONST:    IN      CSTAT        ;READ CONSOLE STATUS.
        ANI     CKBR          ;LOOK AT KB READY BIT.
        MVI     A,0           ;SET A=0 FOR RETURN.
        RZ              ;NOT READY WHEN ZERO.
        CMA              ;IF READY A=FF.
        RET              ;RETURN FROM CONST.

;
; READ A CHARACTER FROM CONSOLE.
;
CONIN:    IN      CSTAT        ;READ CONSOLE STATUS.
        ANI     CKBR          ;IF NOT READY,
        JZ      CONIN         ;READY WHEN HIGH.
        IN      CDATA         ;READ A CHARACTER.
        OUT     CDATA
        ANI     7FH           ;MAKE MOST SIG. BIT = 0.
        RET

;
; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CONOT:    MVI     A,0DH        ;IF IT'S A CR,
        CMP     C            ;THEN HOP OUT
        JZ      CONUL        ;TO NULL ROUTINE.
CONOT1:   IN      CSTAT        ;READ CONSOLE STATUS.
        ANI     CPTR          ;IF NOT READY,
        JZ      CONOT1        ;READY WHEN HIGH.
        MOV     A,C           ;GET CHARACTER.
        OUT     CDATA         ;PRINT IT.
        RET                  ;RETURN.
CONUL:    PUSH    B            ;SAVE B&C.
        MVI     B,CNULL       ;GET NULL COUNT.
CONUL1:   CALL    CONOT1        ;PRINT CR.
        MVI     C,0           ;GET NULL CHAR.
        DCR     B             ;DECREMENT COUNTER.
        JNZ     CONUL1        ;DO NEXT NULL.
        POP     B             ;RESTORE B&C.
        MOV     A,C           ;RESTORE A.
        RET                  ;RETURN.

```

```

;
; PRINT MESSAGE UNTIL ZERO
; MESSAGE ADDRESS REG H * L
;
;
;
PMSG:  MOV     A,M      ;GET CHAR
      ORA     A        ;IS IT A ZERO
      RZ
      MOV     C,A      ;OTHERWISE PRINT
      CALL    CONOT
      INC     H        ;INC ADDRESS
      JMP     PMSG
;
;
MSG8   DB      0DH,0AH,0AH,'LOC.      TEST BYTE      MEMORY BYTE',0
;
; PRINT 8 BIT WORD IN BINARY FORMAT
; INPUT:  DATA IN REG A
;
;
BITS:  MOV     B,A      ; DATA
      MVI     A,80H     ; MASK
OVER:  MVI     C,30H
      MOV     E,A      ; STORE MASK
      ANA     B        ; AND WITH MASK
      JZ      PRNT     ; JUMP IF ZERO
      MVI     C,31H
PRNT:  CALL    CONOT
      ANA     B        ; ZERO CARRY
      MOV     A,E      ; LOAD MASK
      RAR
      JNC     OVER
      RET
;
;
BLNK:  MVI     C,20H     ;PRINT BLANKS, # IN REG. D
LP1:   CALL    CONOT1
      DCR     D
      JNZ     LP1
      RET
;
BINHA: MOV     A,D
      RAR
      RAR
      RAR
      RAR
      CALL    BIN1
      MOV     C,A
      CALL    CONOT
      MOV     A,D
      CALL    BIN1
      MOV     C,A
      CALL    CONOT
      RET
;
;
; OUTPUTS FOUR HEX DIGITS IN ASCII
; ENTER WITH DATA IN REG PAIR E AND D
;
;
;
BINB:  CALL    BINHA
      MOV     D,E
      CALL    BINHA
      RET

```

```

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; CONVERTS HEX TO ASCII
; INPUT: 4 BITS HEX REG A
; OUTPUT: 8 BIT ASCII REG A
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;
BIN1:  ANI      0FH
        ADI      30H
        CPI      3AH
        RC
        ADI      07H
        RET
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; INPUTS 4 DIGITS FROM CONSOLE
; RETURN: 4 HEX DIGITS IN REG E-D
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;
BBIN:   CALL     CONIN
        CALL     AHS1
        RAL
        RAL
        RAL
        RAL
        ANI      0F0H
        MOV      D,A
        CALL     CONIN
        CALL     AHS1
        ANI      0FH
        ORA      D
        MOV      D,A
        CALL     CONIN
        CALL     AHS1
        RAL
        RAL
        RAL
        RAL
        ANI      0F0H
        MOV      E,A
        CALL     CONIN
        CALL     AHS1
        ANI      0FH
        ORA      E
        MOV      E,A
        RET
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
; CONVERT ASCII TO HEX
; INPUT: 8 BIT ASCII REG A
; OUTPUT: 4 BIT HEX REG A
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;
AHS1:   NOP
        SUI      30H
        CPI      0AH
        RC
        SUI      07H
        RET

```

```

;
; INITIATE SIO PORTS
;
;
;
INITA: MVI    A,0AAH           ;GET DUMMY MODE WORD
        OUT    CSTAT          ;OUTPUT IT
        MVI    A,40H           ;GET RESET BIT
        OUT    CSTAT          ;RESET SIO BOARD
        MVI    A,0CEH          ;GET REAL MODE WORD
        OUT    CSTAT          ;SET THE MODE FOR REAL
        MVI    A,37H           ;GET THE COMMAND
        OUT    CSTAT          ;OUTPUT IT
        RET

;
;
;
CRLF:  MVI    C,13             ;CR
        CALL   CONOT
LF:    MVI    C,10             ;LF
        CALL   CONOT1
        MVI    C,7FH
        CALL   CONOT1
        CALL   CONOT
        RET

```

Section 7

Formatted Disk Test

7.1 General Description

The formatted disk test is designed to test the operation of the Pertec floppy disk drive and the Tarbell Disk controller. The ability of the disk system to read, write and seek tracks is tested in the normal formatted mode.

7.2 Program Details

The formatted disk test is the largest and most complex of all the diagnostics in this package. The test is completely self-contained and requires no external I/O subroutines.

The ability of the disk system to read, write and seek tracks is tested by writing a known test pattern and then repositioning the read/write head before performing a verification read. In order to test the disk drive for track positioning and skew error the head is moved between each read and write. The test sequence is as follows:

1. write inner track
2. seek outer track
3. write outer track
4. seek inner track
5. read and verify inner track
6. seek outer track
7. read and verify outer track
8. increment inner and outer track counters

The inner track starts at one and the outer track starts at 38. This continues until all 26 sectors on 77 tracks are tested. Extensive error checking is performed on both the read/write data and the disk status. A large number number of error messages are provided to aid in error analysis. All seek error and read/write data error messages include the sector and track number in question.

7.3 Operation

This test will request that a formatted scratch disk for reading and writing be mounted. The mounting of the disk must then be confirmed by the operator typing a 'Y' on the console device.

The test requires no further interaction unless an error is encountered. After an error is reported, the operator must instruct the program whether to repeat(R) the last sector test or to continue(C) on to the next sector. Raising sense switch '0' will direct the test to automatically continue after an error.

The test can be stopped at any time by typing a 'control B' on the console device.

Each sector contains 128 data bytes. When a read data verification error is encountered, the faulty track and sector are reported and the number of incorrect bytes in the sector is counted. Only the last errant data byte is listed.

The disk test is stored in programmable memory on the diagnostic memory board. The disk test can be run under the diagnostic operating system test controller, or it can be started from the IMSAI 8080 front panel at a starting address of 'C800'.

The disk test is also stored on floppy disk and it can be invoked by the CPM operating system under file name 'DSKTST.COM'.

```

; DISK TEST FOR TARBELL DISK CONTROLLER
; BRIAN J. DONLAN
; 18 MAR 79
; DISC VERSION
;
;
ENTRY:  ORG      0100H
        LXI      H,MSG1          ;OPENING MESSAGE
        CALL     PMSG
        LXI      H,MSG1A
        CALL     PMSG
        CALL     CONIN           ;CHECK KEYBOARD
        CPI      'Y'            ;CHECK IF Y
        JNZ      ENTRY          ; ?? START QVER
        CALL     CRLF
LOOP6:   CALL     CRLF
        XRA      A              ;ZERO ACC
        STA      ERRFLG         ;ZERO EERROR FLAG
        STA      LPCNT          ;ZERO LOOP COUNT
        CALL     HOME           ;HOME DRIVE TO TRK 0
LOOP4:   XRA      A
        STA      INNER         ;ZERO INNER TRK
        MVI      A,38           ;OUTER TRK
        STA      OUTER
        CALL     PAT            ;GET PATTERN
        CALL     INWRT
        MVI      A,34
        CALL     SEEK
        CALL     INRD           ;
        MVI      A,01           ;MOVE BACK AND CHECK TRK 00
        STA      INNER         ;SET UP TO DO PAIRS
                                   ;START PAIRS WITH TRK01
; TEST FOR CONSOLE INTERRUPT
LOOP8:   IN       CSTAT
        ANI      02H           ;KEYBAORD READY
        JZ       LOOP3         ;NO
        IN       CDATA         ;READ KEYS
        CPI      03H           ;CONTROL C
        CPI      02H           ;CONTROLS
        JZ       ENTRY         ;START OVER AGAIN
        ;
LOOP3:   CALL     INWRT          ;WRITE INNER TRK
        CALL     OUTWRT
        CALL     INRD           ;READ INNER TRK
        CALL     OUTRD
        LDA      INNER
        INR      A
        STA      INNER
        ADI      38
        STA      OUTER         ;FIND NEXT OUTER TRK
        CPI      77             ;STORE OUTER TRK
        JNZ      LOOP8         ;TRK 77 YET ?
        LDA      LPCNT         ;NOT DONE YET
        INR      A              ;LOOP COUNTER
        STA      LPCNT
        JMP      LOOP4
;
;
; PAT:   PATTERN ROUTINE EXPANDABLE
        LDA      LPCNT          ;LOAD LOOP COUNTER
        JZ       IST
        CPI      01             ;SECOND PASS
        JZ       SECD
        CPI      02
        JZ       THIRD
        LXI      H,MSG2
        CALL     PMSG           ;END OF PASS
        IN       CSTAT         ;CHECK KEYBOARD
        ANI      02H
        JZ       LOOP6         ;CONTINUE TEST UNTIL INTERRUPTED
        HLT
        JMP      ENTRY
IST:     MVI      A,0FFH        ;ALL ONES PATERN
        STA      PATEN         ;STORE PATTERN
        RET
SECD:    MVI      A,00H         ;ALL ZERO PATTERN
        STA      PATEN
        RET
THIRD:   MVI      A,55H         ;ALTER PATTERN
        STA      PATEN
        RET

```

```

;
; WRITE INNER TRK
INWRT: LDA    INNER
      STA    TRK
BOTH:  CALL   SEEK          ;MOVE HEAD TO TRK
      MVI    A,01          ;FIRST SECTOR
      STA    SECT
;
LOOP1: XRA    A              ;ZERO ACC
      STA    REPETE         ;ZERO REPEAT FLAG
      CALL   WRITE          ;WRITE ONE SECTOR
      LDA    REPETE         ;LOAD REPEAT FLAG
      ORA    A              ;SET FLAGS
      JNZ    LOOP1          ;REPEAT SECTOR
      LDA    SECT
      INR    A              ;INC SECTOR
      STA    SECT
      CPI    27             ;ALL SECTOR DONE ?
      JNZ    LOOP1         ;NO
      RET
;
; WRITE OUTER TRK
OUTWRT: LDA   OUTER
      STA    TRK
      JMP    BOTH          ;COMMON WRITE ROUTINE
;
; READ INNER TRK
INRD:  LDA    INNER
      STA    TRK
BOTH2: CALL   SEEK          ;MOVE HEAD TO TRK
      MVI    A,01          ;FIRST SECTOR
      STA    SECT         ;ZERO SECTOR
;
LOOP5: XRA    A              ;ZERO ERROR COUNT
      STA    ERRFLG
      STA    REPETE
      CALL   READ           ;READ ONE SECTOR
      XRA    A              ;ZERO ACC
      STA    ERRFLG
      LDA    REPETE
      ORA    A              ;REPEAT FLAG
      JNZ    LOOP5          ;SET FLAGS
      LDA    SECT
      INR    A
      STA    SECT          ;NEXT SECTOR
      CPI    27             ;ALL SECTORS DONE ?
      JNZ    LOOP5         ;NO
      RET
;
; READ OUTER TRK
OUTRD: LDA    OUTER
      STA    TRK
      JMP    BOTH2
;
ERRPNT: LXI    H,MSG3        ;ERROR MESSAGE
      CALL   PMSG           ;ERROR COUNT
      LDA    ERRFLG
      MOV    D,A
      CALL   BINHA          ;PRINT ERROR COUNT
      LXI    H,MSG4        ;HEADINGS
      CALL   PMSG
      MVI    D,03          ;SPACE OVER
      CALL   BLNK
      LDA    TRK           ;TRACK NO.
      MOV    D,A
      CALL   BINHA          ;PRINT TRACK NO.
      MVI    D,16          ;SPACE OVER
      CALL   BLNK
      LDA    SECT         ;SECTOR NO.
      MOV    D,A

```

```

CALL BINHA ;PRINT SECTOR NO.
MVI D,13 ;SPACE OVER
CALL BLNK
LDA PATEN
CALL BITS ;PRINT TEST PATTERN
MVI D,12 ;SPACE OVE
CALL BLNK
LDA BADBT ;LAST BAD BYTE
CALL BITS ;PRINT LAST BAD BYTE
RET

;
LPCNT: DS 1 ;SPACE FOR LOOP COUNTER
INNER: DS 1 ;SPACE FOR INNER TRK NO.
OUTER: DS 1 ;SPACE FOR OUTER TRK NO.
PATEN DS 1 ;SPACE FOR TEST PATTERN
ERRFLG: DS 1 ;SPACE FOR ERROR COUNT
BADBT: DS 1 ;SPACE FOR BAD BYTE
BDTRK: DS 1 ;SPACE FOR DISK READ TRK WHEN ERR
REPETE: DS 1 ;REPETE FLAG
MSG1: DB ODH,0AH,'DISK TEST NO. 1 FORMATTED TEST ',0
MSG1A: DB ODH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY',0
MSG2 DB ODH,0AH,' END OF PASS ',0
MSG3: DB ODH,0AH,'DATA ERROR ON DISK CHECK ERROR COUNT IN HEX ',0
MSG4: DB ODH,0AH,' TRACK NO. SECTOR NO. TEST BYTE LAST ERROR'
MSG5: DB ODH,0AH,'HEAD POSITION ',0
MSG6: DB ODH,0AH,'DISK TRACK CONTROLLER TRACK SECTOR ',0DH,0AH,0
MSG7: DB ODH,0AH,0DH,0AH,' !! EXECUTION STOPPED !! ',0
MSG8: DB ODH,0AH,'TYPE R TO RETRY, C TO CONTINUE, ANYTHING ELSE STOP ',0

;
CSTAT EQU 3 ;CONSOLE STATUS PORT.
CCOM EQU 3 ;CONSOLE COMMAND PORT.
CDATA EQU 2 ;CONSOLE DATA PORT.
CKBR EQU 00000010B ;KEYBOARD READY BIT.
CPTR EQU 00000001B ;PRINT READY BIT.
CNULL EQU 1 ;CONSOLE NULL COUNT.
DISK EQU 0F8H ;DISK BASE ADDRESS.
DCOM EQU DISK ;DISK COMMAND PORT.
DSTAT EQU DISK ;DISK STATUS PORT.
TRACK EQU DISK+1 ;DISK TRACK PORT.
SECTP EQU DISK+2 ;DISK SECTOR PORT.
DDATA EQU DISK+3 ;DISK DATA PORT.
WAIT EQU DISK+4 ;DISK WAIT PORT.
DCONT EQU DISK+4 ;DISK CONTROL PORT.

TRK: DS 1 ;ADDRESS FOR TRACK
SECT: DS 1 ;ADDRESS FOR SECTOR

;
; READ A CHARACTER FROM CONSOLE.
;
CONIN: IN CSTAT ;READ CONSOLE STATUS.
ANI CKBR ;IF NOT READY,

JZ CONIN ;READY WHEN HIGH.
IN CDATA ;READ A CHARACTER.
OUT CDATA
ANI 7FH ;MAKE MOST SIG. BIT = 0.
RET

;
; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CONOT: MVI A,0DH ;IF IT'S A CR,
CMP C ;THEN HOP OUT
JZ CONUL ;TO NULL ROUTINE.
CONOT1: IN CSTAT ;READ CONSOLE STATUS.
ANI CPTR ;IF NOT READY,
JZ CONOT1 ;READY WHEN HIGH.
MOV A,C ;GET CHARACTER.
OUT CDATA ;PRINT IT.
RET ;RETURN.
CONUL: PUSH B ;SAVE B&C.
MVI B,CNULL ;GET NULL COUNT.
CONUL1: CALL CONOT1 ;PRINT CR.
MVI C,0 ;GET NULL CHAR.
DCR B ;DECREMENT COUNTER.
JNZ CONUL1 ;DO NEXT NULL.
POP B ;RESTORE B&C.
MOV A,C ;RESTORE A.
RET ;RETURN.

```

```

; MOVE DISK TO TRACK ZERO.
;
HOME: MVI A,0DOH ;CLEAR ANY PENDING COMMAND.
      OUT DCOM
      XRA A ;ZERO ACC
      STA TRK ;STORE TRACK
HOME1: IN DSTAT ;READ DISK STATUS.
      RRC ;LOOK AT LSB.
      JC HOME1 ;WAIT FOR NOT BUSY.
      MVI A,3 ;20 MS STEP RATE.
      OUT DCOM ;ISSUE HOME COMMAND.
      IN WAIT ;WAIT FOR INTRQ.
      ORA A ;SET FLAGS.
      JM HERR ;ERROR IF DRQ.
      IN DSTAT ;READ DISK STATUS.
      MOV D,A ;SAVE IN REGISTER D.
      ANI 4 ;LOOK AT BIT 2.
      JZ HERR ;ERROR IF NOT TRK 0.
      MOV A,D ;GET STATUS BACK.
      ANI 91H ;MASK NON-ERROR BITS.
      RZ ;RETURN IF NO ERROR.
HERR: LXI H,HEMSG ;PRINT "HOME ".
      MOV A,D ;MASK NON-ERROR BITS.
      ANI 91H
      MOV D,A
      JMP ERMSG ;DO COMMON ERROR MSGS.

;
; SELECT DISK NUMBER.
;
INTDSK: MVI A,02 ;DRIVE NO. 1
DSK1: OUT DCON1 ;SET THE LATCH WITH CODE.
      RET ;RETURN FROM SELDSK.

;
; READ THE SECTOR AT SECT, FROM THE PRESENT TRACK.
; SECTOR IN SECT
; HEAD LOAD FIRST
;
READ: LXI H,080H ;READ BUFFER
      LDA SECT
READ1: OUT SECTP ;SET SECTOR INTO 1771.
      MVI A,8CH ;CODE FOR READ W/O HD LD.
READ: OUT DCOM ;SEND COMMAND TO 1771.
RLOOP: IN WAIT ;WAIT FOR DRQ OR INTRQ.
      ORA A ;SET FLAGS.
      JP RDDONE ;DONE IF INTRQ.
      IN DDATA ;READ A DATA BYTE FROM DISK.

;
      MOV M,A ;STORE IN BUFFER
      INX H ;INC BUFF POINTER
;
      JMP RLOOP

;
; COMPARE DATA WITH TEST BYTE;
RDDONE: LXI H,080H ;HEAD OF BUFFER
      LDA PATEN ;TEST PATTERN
      MOV B,A ;PATTERN TO B
      MVI D,080H ;COUNTER FOR BYTES
COMPLP: MOV A,M ;GET DATA
      CMP B ;COMPARE WITH TB
      JNZ DATERR ;ERROR
ERRET: INX H
      DCR D ;DEC BYTE COUNT
      JNZ COMPLP ;DO 128 TIMES
      IN DSTAT ;READ DISK STATUS.
      ANI 9DH ;LOOK AT ERROR BITS.
      MOV D,A ;SAVE ERROR BITS
      LDA ERRFLG ;READ ERROR FLAG
      ORA D ;SET FLAGS ON COMBO
      RZ ;RETURN IF NONE.
      LXI H,RDMSG ;PRINT "READ ".
ERMSG: CALL PMSG ;PRINT ORIGIN MESSAGE.

```

```

;      COMMON ERROR PRINT OUT
;
ERMSG1: MOV  A,D      ;GET ERROR BITS.
        ANI  80H      ;IF BIT 7 HIGH,
        LXI  H,NRMSG  ;"NOT READY".
        CNZ  PMSG
        MOV  A,D      ;GET ERROR BITS.
        ANI  10H      ;IF BIT 4 IS HIGH,
        LXI  H,RNMSG  ;PRINT "RECORD NOT FOUND"
        CNZ  PMSG
        MOV  A,D      ;GET ERROR BITS.
        ANI  8H       ;IF BIT 3 IS HIGH,
        LXI  H,CRCMSG ;PRINT "CRC ERROR".
        CNZ  PMSG
        MOV  A,D      ;GET ERROR BITS.
        ANI  4H       ;IF BIT 2 IS HIGH,
        LXI  H,LMSG   ;PRINT "LOST DATA".
        CNZ  PMSG
        MOV  A,D      ;GET ERROR BITS.
        ANI  1        ;IF BIT 1 IS HIGH,
        LXI  H,BSYMSG ;PRINT "BUSY".
        CNZ  PMSG
PERMSG: LXI  H,ERRMSG ;PRINT "ERROR."
        CALL PMSG
        MOV  A,D      ;MOVE FLAGS TO ACC
        ANI  18H      ;CRC OR RECORD NOT FOUND
        JZ   RETRY
TRKCHK: MVI  A,0C4H
        OUT  DCOM      ;READ ADDRESS
        IN   WAIT      ;TRACK ADDRESS
        IN   DDATA
        STA  BDIRK
CHKS2:  IN   WAIT      ;DUMP REST OF DATA
        JM   CHKS2
        LXI  H,MSG5    ;HEAD ERROR MESSAGE
        CALL PMSG
        LXI  H,MSG6    ;HEADINGS
        CALL PMSG
        MVI  D,05H
        CALL BLNK      ;SPACE OVER
        LDA  BDIRK     ;DISK TRK
        MOV  D,A
        CALL BINHA     ;PRINT TRK
        MVI  D,15H
        CALL BLNK      ;SPACE OVER
        IN   TRACK
        MOV  D,A
        CALL BINHA     ;PRINT TRK
        MVI  D,13H
        CALL BLNK
        LDA  SECT      ;SECTOR
        MOV  D,A
        CALL BINHA     ;PRINT SECTO NO.
        LDA  ERRFLG
RETRY:  ORA  A          ;SET FLAGS
        CNZ  ERRPNT    ;GO TO READ CHECK ERROR PRINT
        IN   CDATA     ;CLEAR KEYBOARD
        IN   OFFH      ;READ SENSE SWITCHES
        ANI  01H       ;SWITCH 0
        JNZ  CONT
        LXI  H,MSG8
        CALL PMSG      ;REQUEST INPUT
        CALL CONIN     ;READ KEYS
        CPI  'R'       ;CHECK FOR R
        JZ   FIX
        CPI  'C'       ;CHECK FOR C
        JZ   CONT
        HLT
FIX:    MVI  A,01      ;SET REPETE FLAG
        STA  REPETE
        CALL CRLF

```

```

        CALL    CRLF
        RET
CONT:   CALL    CRLF
        CALL    CRLF
        RET
;
DATERR: STA     BADBT           ;SAVE BAD BYTE
        LDA     ERRFLG         ;LOAD ERROR COUNT
        INR     A
        STA     ERRFLG         ;NEW COUNT
        JMP     ERRET          ;RETURN
;
; WRITE THE SECTOR AT SECT, ON THE PRESENT TRACK.
; USE STARTING ADDRESS AT DMAADD.
; LOAD HEAD FIRST
;
WRITE:  LDA     PATEN
        MOV     B,A            ;TEST PATTERN IN B
        LDA     SECT          ;LOAD SECTOR
WRITE1: OUT     SECTP           ;SET THE SECTOR INTO 1771.
        MVI     A,OACH         ;SET UP 1771 FOR WRITE.
        OUT     DCOM
WLOOP:  IN      WAIT            ;WAIT FOR READY.
        ORA     A              ;SET FLAGS.
        JP      WDONE          ;HOP OUT WHEN DONE.
;
; INSERT PATTERN HERE
        MOV     A,B            ;LOAD TEST PATTERN
;
;
        OUT     DDATA          ;WRITE ONTO DISK.
        INX     H              ;INCREMENT MEM PTR.
        JMP     WLOOP          ;KEEP WRITING.
WDONE:  IN      DSTAT           ;READ DISK STATUS.
        ANI     OFDH           ;LOOK AT THESE BITS.
        MOV     D,A            ;SAVE STATUS BITS
PROCER: RZ                     ;RETURN IF NO ERR.
WERRO:  LXI     H,WMSG          ;PRINT "WRITE ".
        CALL    PMSG
        MOV     A,D            ;GET ERROR BITS.
        ANI     40H            ;LOOK AT BIT 6.
        LXI     H,WPMMSG        ;PRINT "PROTECT ".
        CNZ     PMSG
        MOV     A,D            ;GET ERROR BITS.
        ANI     20H            ;LOOK AT BIT 5.
        LXI     H,WFMSG         ;PRINT "FAULT ".
        CNZ     PMSG
        JMP     ERMSG1          ;DO COMMON MESSAGES.
;
; MOVE THE HEAD TO THE TRACK IN REGISTER A.
;
SEEK:   OUT     DDATA          ;TRACK TO DATA REGISTER.
BUSY:   IN      DSTAT           ;READ DISK STATUS.
        RRC                     ;LOOK AT BIT 0.
        JC      BUSY           ;WAIT TILL NOT BUSY.
        MVI     A,12H          ;SET FOR 10 MS STEP.
        ORI     4              ;VERIFY ON LAST TRACK.
        OUT     DCOM           ;ISSUE SEEK COMMAND.
        IN      WAIT           ;WAIT FOR INTRQ.
        IN      DSTAT          ;READ STATUS.
        ANI     91H            ;LOOK AT BITS.
        MOV     D,A            ;SAVE STATUS
        RZ                     ;RETURN IF NO ERROR
        LXI     H,SKMSG         ;PRINT "SEEK ".
        JMP     ERMSG          ;DO COMMON ERR MESSAGES.
;
; PRINT THE MESSAGE AT H&L UNTIL A ZERO.
;
PMSG:   MOV     A,H            ;GET A CHARACTER.
        ORA     A              ;IF IT'S ZERO,
        RZ                     ;RETURN.
        MOV     C,A            ;OTHERWISE,
        CALL    CONOT          ;PRINT IT.
        INX     H              ;INCREMENT H&L,
        JMP     PMSG           ;AND GET ANOTHER.

```

```

; CBIOS MESSAGES
RENT EQU 0000H ;MONITOR ENTRY
;
;
NRMSG: DB 'NOT READY ',0
RNMSG: DB 'RECORD NOT FOUND ',0
CRCMSG: DB 'CRC ',0
LDMSG: DB 'LOST DATA ',0
BSYMSG: DB 'BUSY ',0
WPMSG: DB 'PROTECT ',0
WFMSG: DB 'FAULT ',0
ERRMSG: DB 'ERROR.',0
RDMSG: DB ODH,0AH,'READ ',0
WTMSG: DB ODH,0AH,'WRITE ',0
SKMSG: DB ODH,0AH,'SEEK ',0
HEMSG: DB ODH,0AH,'HOME ',0
MNTMSG: DB ODH,0AH,'MOUNT ',0

;
;
; PRINT 8 BIT WORD IN BINARY FORMAT
; INPUT: DATA IN REG A
;
;
BITS: MOV B,A ; DATA
MVI A,80H ; MASK
OVER: MVI C,30H
MOV E,A ; STORE MASK
ANA B ; AND WITH MASK
JZ PRNT ; JUMP IF ZERO
MVI C,31H
PRNT: CALL CONOT
ANA B ; ZERO CARRY
MOV A,E ; LOAD MASK
RAR
JNC OVER
RET

;
;
BLNK: MVI C,20H ;PRINT BLANKS, # IN REG. D
LP1: CALL CONOT1
DCR D
JNZ LP1
RET

;
BINHA: MOV A,D
RAR
RAR
RAR
RAR
CALL BIN1
MOV C,A
CALL CONOT
MOV A,D
CALL BIN1
MOV C,A
CALL CONOT
RET

;
;
; OUTPUTS FOUR HEX DIGITS IN ASCII
; ENTER WITH DATA IN REG PAIR E AND D
;
;
BINB: CALL BINHA
MOV D,E
CALL BINHA
RET

```


Section 8

Track Write Routine

8.1 General Description

The track write routine is not a diagnostic test, but rather a programmed aid to be used during disk drive maintenance and alignment. When executed, an entire track, as selected in the sense switches, is repeatedly written with an all ones pattern.

8.2 Program Details

The disk track write routine does not test the data written, but it does test and report on the ability of the disk drive to load the read/write head and to move it to a selected track.

The track write routine reads the IMSAI 8080 front panel sense switches for the desired track. Error checking is performed to test if the selected track is greater than 76, the last track. If the selected track number is greater than 76, the front panel sense lights oscillate and no writing is performed.

If the selected track number is valid, a seek to that track is performed. The read/write head is then loaded and full track writing can begin at the next index marker. The entire track is written with all ones until the index mark is reached again. The track is repeatedly written until a new track is selected by the operator.

This continuous writing can be very valuable when troubleshooting or aligning the disk drive heads and read/write circuitry as outlined in the Pertec disk drive manual.

8.3 Operation

Upon entering the program, the routine will request that a scratch disk be mounted. The scratch disk does not need to be formatted for this routine. After the scratch disk is loaded, the desired track should be set in the sense switches and then type a 'Y' on the console to begin. The drive will then perform a seek to that track and begin writing.

The track number may be changed at any time and the sequence will begin on the new track. A convenient way to stop the writing at any time is to raise the left most sense switch which halts the write operations by forcing a large track number.

Some drive error messages ask the operator whether to abort or continue after the error.

The track write routine is stored in programmable memory on the diagnostic memory board. The write routine can be run under the diagnostic operating system test controller, or it can be started from the IMSAI 8080 front panel at a starting address of 'CD80' Hex.

The track write routine is also stored on floppy disk and it can be invoked by the CPM operating system.

under file name 'WRTRK.COM'.

```

; DISC TEST: FULL TRACK WRITE
; SELECT TRACK IN SENSE SWITCHES
; DISC VERSION
; REFIN CONLAN
; JUNE 79
;
; ORG 0100H
;
ENTRYA: JMP STARTA
ENTRYB: LXI H,080H
        SPHL
        DI
        CALL INITA
STARTA: LXI H,MSG1B
        CALL PMSG
        CALL CONIN
        CPI 'Y'
        LXI H,MSG2A
        JNZ READT
        CALL HOME
STARTB: ORA A
        JNZ STARTA
        LXI H,STARTB
        PUSH H
STARTC: IN OFFH
        CPI 77
        JNC ERRA
        CALL SEEK
        MVI B,OFFH
        MVI A,0F4H
        OUT DCOM
WRTLP: IN WAIT
        ORA A
        JP WDONE
        MOV A,B
        OUT DDATA
        JMP WRTLP
;
;
ERRA: MVI B,0F0H
ERRLP: MOV A,B
        CMA
        OUT OFFH
        LXI D,01H
        LXI H,000H
ERRLPB: DAD D
        JNC ERRLPB
        MOV B,A
        IN OFFH
        CPI 77
        JNC ERRLP
        LXI H,0
        LXI D,01H
        DAD D
        JNC DELP
        LXI H,0
        LXI D,01H
        DAD D
        JNC DELPA
        JMP STARTC
;
;
;SET STACK
;RESET SIO
;READ KEYBOARD
;SET FLAGS
;ERROR START OVER
;SUBROUTINE RETURN
;READ SENSE SWITCHES
;PREVENT TRACK OVER-DRIVE
;MOVE HEAD TO TRACK
;TEST PATTERN
;WRITE TRK COMMAND
;DELAY LOOP
;SEE IF SWITCHES FIXED

```

```

MSG1B: DB      0DH,0AH,0AH,'DISK TRACK WRITE ROUTINE'
DB      0DH,0AH,'LOAD DISPATCH DISK TYPE Y WHEN READY ',0
MSG2A: DB      0DH,0AH,'?? ',0
;
;
LPENT: DS      1          ;SPACE FOR LOOP COUNTER
BADBT: DS      1          ;SPACE FOR BAD BYTE
BDRK: DS      1          ;SPACE FOR DISK READ TRK WHEN ERR
MSG1: DB      0DH,0AH,'DISK TEST NO. 1 FORMATTED TEST ',0
MSG1A: DB      0DH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY',0
MSG2: DB      0DH,0AH,'END OF PASS ',0
MSG3: DB      0DH,0AH,'DATA ERROR ON DISK CHECK      ERROR COUNT IN HEX ',0
MSG4: DB      0DH,0AH,' TRACK NO.          SECTOR NO.          TEST BYTE          LAST ERROR',0DH
MSG5: DB      0DH,0AH,'HEAD POSITION ',0
MSG6: DB      0DH,0AH,'DISK TRACK          CONTROLLER TRACK          SECTOR ',0DH,0AH,0
MSG7: DB      0DH,0AH,0DH,0AH,' !! EXECUTION STOPPED !! ',0
MSG8: DB      0DH,0AH,'TYPE N TO RETRY, C TO CONTINUE, ANYTHING ELSE STOP ',0
;
CSTAT EQU 3          ;CONSOLE STATUS PORT.
CCOM EQU 3          ;CONSOLE COMMAND PORT.
CDATA EQU 2          ;CONSOLE DATA PORT.
CKBR EQU 00000010B   ;KEYBOARD READY BIT.
CPTR EQU 00000001B   ;PRINT READY BIT.
CNUL EQU 1          ;CONSOLE NULL COUNT.
DISK EQU 0FBH        ;DISK BASE ADDRESS.
DCOM EQU DISK        ;DISK COMMAND PORT.
DSTAT EQU DISK       ;DISK STATUS PORT.
TRACK EQU DISK+1     ;DISK TRACK PORT.
SECTP EQU DISK+2     ;DISK SECTOR PORT.
DDATA EQU DISK+3     ;DISK DATA PORT.
WAIT EQU DISK+4      ;DISK WAIT PORT.
DCONT EQU DISK+4     ;DISK CONTROL PORT.

TRK: DS      1          ;ADDRESS FOR TRACK
SECT: DS     1          ;ADDRESS FOR SECTOR

; READ A CHARACTER FROM CONSOLE.
;
CONIN: IN      CSTAT          ;READ CONSOLE STATUS.
      ANI CKBR              ;IF NOT READY,
      JZ      CONIN          ;READY WHEN HIGH.
      IN      CDATA          ;READ A CHARACTER.
      OUT     CDATA
      ANI 7FH              ;MAKE MOST SIG. BIT = 0.
      RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CONCT: MVI A,0DH          ;IF IT'S A CR,
      CMP C              ;THEN HOP OUT
      JZ      CONUL        ;TO NULL ROUTINE.
CONOT1: IN      CSTAT          ;READ CONSOLE STATUS.
      ANI CPTR            ;IF NOT READY,
      JZ      CONOT1        ;READY WHEN HIGH.
      MOV A,C              ;GET CHARACTER.
      OUT     CDATA          ;PRINT IT.
      RET                ;RETURN.
CONUL: PUSH B              ;SAVE B&C.
      MVI B,CNUL          ;GET NULL COUNT.
CONUL1: CALL CONOT1          ;PRINT CR.
      MVI C,0              ;GET NULL CHAR.
      DCR B              ;DECREMENT COUNTER.
      JNZ CONUL1          ;DO NEXT NULL.
      POP B              ;RESTORE B&C.
      MOV A,C              ;RESTORE A.
      RET                ;RETURN.

```

```

; MOVE DISK TO TRACK ZERO.
HOME: MVI A,000H ;CLEAR ANY PENDING COMMAND.
      OUT DCOM
      XRA A ;ZERO ACC
      STA TRK ;STORE TRACK
HOME1: IN DSTAT ;READ DISK STATUS.
      RRC ;LOOK AT LSB.
      JC HOME1 ;WAIT FOR NOT BUSY.
      MVI A,3 ;20 MS STEP RATE.
      OUT DCOM ;ISSUE HOME COMMAND.
      IN WAIT ;WAIT FOR INTRQ.
      CMA A ;SET FLAGS.
      JM HERR ;ERROR IF DRQ.
      IN DSTAT ;READ DISK STATUS.
      MOV D,A ;SAVE IN REGISTER D.
      ANI 4 ;LOOK AT BIT 2.
      JZ HERR ;ERROR IF NOT TRK 0.
      MOV A,D ;GET STATUS BACK.
      ANI 91H ;MASK NON-ERROR BITS.
      RZ ;RETURN IF NO ERROR.
HERR: LXI H,HEMSG ;PRINT "HOME ".
      MOV A,D ;MASK NON-ERROR BITS.
      ANI 91H
      MOV D,A
      JMP ERMSG ;DO COMMON ERROR MSGS.

;
ERMSG: CALL PMSG ;PRINT ORIGIN MESSAGE.
;
;
COMMON ERROR PRINT OUT
ERMSG1: MOV A,D ;GET ERROR BITS.
      ANI 80H ;IF BIT 7 HIGH,
      LXI H,NRMSG ;"NOT READY".
      CNZ PMSG
      MOV A,D ;GET ERROR BITS.
      ANI 10H ;IF BIT 4 IS HIGH,
      LXI H,RRMSG ;PRINT "RECORD NOT FOUND"
      CNZ PMSG
      MOV A,D ;GET ERROR BITS.
      ANI 8H ;IF BIT 3 IS HIGH,
      LXI H,CRCMSG ;PRINT "CRC ERROR".
      CNZ PMSG
      MOV A,D ;GET ERROR BITS.
      ANI 4H ;IF BIT 2 IS HIGH,
      LXI H,LDMSG ;PRINT "LOST DATA".
      CNZ PMSG
      MOV A,D ;GET ERROR BITS.
      ANI 1 ;IF BIT 1 IS HIGH,
      LXI H,BSYMSG ;PRINT "BUSY".
      CNZ PMSG
PERMSG: LXI H,ERRMSG ;PRINT "ERROR."
      CALL PMSG
      MOV A,D ;MOVE FLAGS TO ACC
      ANI 18H ;CRC OR RECORD NOT FOUND
      JZ RETRY
TRKCHK: MVI A,0C4H
      OUT DCOM ;READ ADDRESS
      IN WAIT ;TRACK ADDRESS
      IN DDATA
      STA BDIRK
CHKS2: IN WAIT ;DUMP REST OF DATA
      JM CHKS2
      LXI H,MSG5 ;HEAD ERROR MESSAGE
      CALL PMSG
      LXI H,MSG6 ;HEADINGS
      CALL PMSG
      MVI D,05H
      CALL BLNK ;SPACE OVER

```

```

        LDA      BDTRK          ;DISK TRK
        MOV      D,A
        CALL     BINHA          ;PRINT TRK
        MVI      D,15H
        CALL     BLNK           ;SPACE OVER
        IN       TRACK
        MOV      D,A
        CALL     BINHA          ;PRINT TRK
        MVI      D,13H
        CALL     BLNK
        LDA      SECT           ;SECTOR
        MOV      D,A
        CALL     BINHA          ;PRINT SECTO NO.
RETRY:  IN       CDATA          ;CLEAR KEYBOARD
        IN       OFFH           ;READ SENSE SWITCHES
        ANI      080H           ;SWITCH 0
        JNZ      CONT
        LXI      H,MSG8
        CALL     PMSG           ;REQUEST INPUT
        CALL     CONIN          ;READ KEYS
        CPI      'R'           ;CHECK FOR R
        JZ       FIX
        CPI      'C'           ;CHECK FOR C
        JZ       CONT
        HLT
FIX:    MVI      A,01           ;SET REPETE FLAG
        CALL     CRLF
        CALL     CRLF
        RET
CONT:   CALL     CRLF
        CALL     CRLF
        RET
;
;
;
WDONE:  IN       DSTAT          ;READ DISK STATUS.
        ANI      0FDH           ;LOOK AT THESE BITS.
        MOV      D,A           ;SAVE STATUS BITS
PROGCR: RZ
WERRO:  LXI      H,WMSG         ;RETURN IF NO ERR.
        CALL     PMSG          ;PRINT "WRITE ".
        MOV      A,D           ;GET ERROR BITS.
        ANI      40H           ;LOOK AT BIT 6.
        LXI      H,WMSG         ;PRINT "PROTECT ".
        CNZ      PMSG
        MOV      A,D           ;GET ERROR BITS.
        ANI      20H           ;LOOK AT BIT 5.
        LXI      H,WMSG         ;PRINT "FAULT ".
        CNZ      PMSG
        JMP      ERMSG          ;DO COMMON MESSAGES.
;
; MOVE THE HEAD TO THE TRACK IN REGISTER A.
;
SEEK:   OUT      DDATA          ;TRACK TO DATA REGISTER.
BUSY:   IN       DSTAT          ;READ DISK STATUS.
        RRC                   ;LOOK AT BIT 0.
        JC       BUSY          ;WAIT TILL NOT BUSY.
        MVI      A,12H         ;SET FOR 10 MS STEP.
        OUT      DCOM          ;ISSUE SEEK COMMAND.
        IN       WAIT          ;WAIT FOR INTRQ.
        IN       DSTAT          ;READ STATUS.
        ANI      91H           ;LOOK AT BITS.
        MOV      D,A           ;SAVE STATUS
        RZ                   ;RETURN IF NO ERROR
        LXI      H,SKMSG        ;PRINT "SEEK ".
        JMP      ERMSG          ;DO COMMON ERR MESSAGES.
;
; PRINT THE MESSAGE AT H&L UNTIL A ZERO.
;
PMSG:   MOV      A,M           ;GET A CHARACTER.
        ORA      A             ;IF IT'S ZERO,
        RZ                   ;RETURN.
        MOV      C,A           ;OTHERWISE,
        CALL     CONOT          ;PRINT IT.
        INX      H             ;INCREMENT H&L,
        JMP      PMSG          ;AND GET ANOTHER.

```



```

;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      INITIATE SIO PORTS
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;
INITA: MVI      A,0AAH          ;GET DUMMY MODE WORD
        OUT     CSTAT          ;OUTPUT IT
        MVI     A,40H          ;GET RESET BIT
        OUT     CSTAT          ;RESET SIO BOARD
        MVI     A,0CEH         ;GET REAL MODE WORD
        OUT     CSTAT          ;SET THE MODE FOR REAL
        MVI     A,37H          ;GET THE COMMAND
        OUT     CSTAT          ;OUTPUT IT
        RET

;
;
;
CRLF:  MVI      C,13           ;CR
        CALL    CONOT
LF:     MVI      C,10           ;LF
        CALL    CONOT1
        MVI     C,7FH
        CALL    CONOT1
        CALL    CONOT
        RET

```

Section 9

Track Read Routine

9.1 General Description

The Track Read Routine is not a diagnostic test, but rather a programmed aid to be used during disk drive maintenance and alignment. When executed, an entire track as selected on the sense switches, is repeatedly read.

9.2 Program Details

The disk drive read routine does not save or test the data being read, but it does test and report on the ability of the disk drive to load the read/write head and to move it to a selected track.

The track read routine reads the IMSAI 8080 front panel sense switches for the desired track. Error checking is performed to test if the selected track number is greater than 76, the last track. If the selected number is greater than 76, the front panel sense lights oscillate and no reading is performed.

If the track number is valid, a seek to that track is performed. The read/write head is then loaded so full track reading can begin on the next index marker. The entire track is read, but the data is not saved. The track is repeatedly read until a new track is selected by the operator.

This continuous reading can be very valuable when

trouble shooting or aligning the disk head or read/write circuitry as outlined in the Pertec disk drive manual.

9.3 Operation

Upon entering the program, the routine will request that a scratch disk be mounted. The scratch disk does not need to be formatted for this routine. After a scratch disk is loaded, the desired track should be set in the sense switches a. then type a 'Y' on the console to begin. The drive will then perform a seek to that track and begin reading.

The track number may be changed at anytime and the sequence will begin on the new track. A convenient way to stop the reading at any time is to raise the left most sense switch which halts the read operations by forcing a large track number.

Some drive error asks the operator whether to abort or continue after the error.

The track read routine is stored in programmable memory on the diagnostic board. The read routine can be run under the diagnostic operating system test controller, or it can be started from the IMSAI 8080 front panel at a starting address of 'CE40'.

The read routine is also stored on floppy disk and it can be invoked by the CPM operating system under file name 'RDTRK.COM'.

```

;
; DISC TEST FULL TRACK READ
; SELECT TRACK IN SENSE SWITCHES
; DISC VERSION
; BRIAN DONLAN
; JUNE 79
;
; ORG 0100H
;
ENTRYA: JMP STARTA
ENTRYB: LXI H,080H
        SPHL
        DI
        CALL INITA
STARTA: LXI H,MSG1B
READT: CALL PMSG
        CALL CONIN
        CPI 'Y'
        LXI H,MSG2A
        JNZ READT
        CALL HOME
STARTB: ORA A
        JNZ STARTA
        LXI H,STARTB
        PUSH H
STARTC: IN OFFH
        CPI 77
        JNC ERRA
SEEKA: CALL SEEK
        MVI A,0ESH
        OUT DCOM
RDLP: IN WAIT
        ORA A
        JP RDONE
        IN DDATA
        JMP RDLP
;
;
ERRA: MVI B,0FOH
ERRLP: MOV A,B
        CMA
        OUT OFFH
        LXI D,01H
        LXI H,000H
ERRLPB: DAD D
        JNC ERRLPB
        MOV B,A
        IN OFFH
        CPI 77
        JNC ERRLP
DELAY: LXI H,0
        LXI D,01H
DELP: DAD D
        JNC DELP
DELAYA: LXI H,0
        LXI D,01H
DELPA: DAD D
        JNC DELPA
        JMP STARTC

```

```

;
RDONE: IN      DSTAT      ;READ STATUS
      ANI      9DH
      MOV      D,A
      RZ
      LXI      H,RDMSG    ;PRINT 'READ'
      JMP      ERMSG
;
;;
MSG1B: DB      0DH,0AH,0AH,'DISK TRACK READ ROUTINE'
      DB      0DH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY ',0
MSG2A: DB      0DH,0AH,'?? ',0
;
;
LPCNT: DS      1          ;SPACE FOR LOOP COUNTER
BADBT: DS      1          ;SPACE FOR BAD BYTE
BDIRK: DS      1          ;SPACE FOR DISK READ TRK WHEN ERR
MSG1:  DB      0DH,0AH,'DISK TEST NO. 1 FORMATTED TEST ',0
MSG1A: DB      0DH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY',0
MSG2:  DB      0DH,0AH,' END OF PASS ',0
MSG3:  DB      0DH,0AH,'DATA ERROR ON DISK CHECK          ERROR COUNT IN HEX ',0
MSG4:  DB      0DH,0AH,' TRACK NO.          SECTOR NO.          TEST BYTE          LAST ERROR',0DH
MSG5:  DB      0DH,0AH,'HEAD POSITION ',0
MSG6:  DB      0DH,0AH,'DISK TRACK          CONTROLLER TRACK          SECTOR ',0DH,0AH,0
MSG7:  DB      0DH,0AH,0DH,0AH,' !! EXECUTION STOPPED !! ',0
MSG8:  DB      0DH,0AH,'TYPE R TO RETRY, C TO CONTINUE, ANYTHING ELSE STOP ',0
;
CSTAT EQU      3          ;CONSOLE STATUS PORT.
CCOM  EQU      3          ;CONSOLE COMMAND PORT.
CDATA EQU      2          ;CONSOLE DATA PORT.
CKBR  EQU      00000010B  ;KEYBOARD READY BIT.
CPTR  EQU      00000001B  ;PRINT READY BIT.
CNULL EQU      1          ;CONSOLE NULL COUNT.
DISK  EQU      0F8H       ;DISK BASE ADDRESS.
DCOM  EQU      DISK       ;DISK COMMAND PORT.
DSTAT EQU      DISK       ;DISK STATUS PORT.
TRACK EQU      DISK+1     ;DISK TRACK PORT.
SECTP EQU      DISK+2     ;DISK SECTOR PORT.
DDATA EQU      DISK+3     ;DISK DATA PORT.
WAIT  EQU      DISK+4     ;DISK WAIT PORT.
DCONT EQU      DISK+4     ;DISK CONTROL PORT.
;
TRK:  DS      1          ;ADDRESS FOR TRACK
SECT: DS      1          ;ADDRESS FOR SECTOR
;
; READ A CHARACTER FROM CONSOLE.
;
CONIN: IN      CSTAT      ;READ CONSOLE STATUS.
      ANI      CKBR      ;IF NOT READY,
;
      JZ      CONIN      ;READY WHEN HIGH.
      IN      CDATA      ;READ A CHARACTER.
      OUT     CDATA
      ANI      7FH       ;MAKE MOST SIG. BIT = 0.
      RET
;
; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CONOT: MVI      A,0DH     ;IF IT'S A CR,
      CMP      C          ;THEN HOP OUT
      JZ      CONUL      ;TO NULL ROUTINE.
CONOT1: IN      CSTAT     ;READ CONSOLE STATUS.
      ANI      CPTR      ;IF NOT READY,
      JZ      CONOT1     ;READY WHEN HIGH.
      MOV      A,C        ;GET CHARACTER.
      OUT     CDATA      ;PRINT IT.
      RET               ;RETURN.
CONUL: PUSH     B         ;SAVE B&C.
      MVI      B,CNULL    ;GET NULL COUNT.
CONUL1: CALL    CONOT1    ;PRINT CR.
      MVI      C,0        ;GET NULL CHAR.
      DCR      B         ;DECREMENT COUNTER.
      JNZ     CONUL1     ;DO NEXT NULL.
      POP      B         ;RESTORE B&C.
      MOV      A,C        ;RESTORE A.
      RET               ;RETURN.

```

```

; MOVE DISK TO TRACK ZERO.
HOME: MVI A,000H ;CLEAR ANY PENDING COMMAND.
      OUT DCOM
      XRA A ;ZERO ACC
      STA TRK ;STORE TRACK
HOME1: IN DSTAT ;READ DISK STATUS.
      RRC ;LOOK AT LSB.
      JC HOME1 ;WAIT FOR NOT BUSY.
      MVI A,3 ;20 MS STEP RATE.
      OUT DCOM ;ISSUE HOME COMMAND.
      IN WAIT ;WAIT FOR INTRQ.
      ORA A ;SET FLAGS.
      JM HERR ;ERROR IF DRQ.
      IN DSTAT ;READ DISK STATUS.
      MOV D,A ;SAVE IN REGISTER D.
      ANI 4 ;LOOK AT BIT 2.
      JZ HERR ;ERROR IF NOT TRK 0.
      MOV A,D ;GET STATUS BACK.
      ANI 91H ;MASK NON-ERROR BITS.
      RZ ;RETURN IF NO ERROR.
HERR: LXI H,HEMSG ;PRINT "HOME ".
      MOV A,D ;MASK NON-ERROR BITS.
      ANI 91H
      MOV D,A
      JMP ERMMSG ;DO COMMON ERROR MSGS.
;
;
ERMMSG: CALL PMSG ;PRINT ORIGIN MESSAGE.
;
;
COMMON ERROR PRINT OUT
;
ERMMSG1: MOV A,D ;GET ERROR BITS.
        ANI 80H ;IF BIT 7 HIGH,
        LXI H,NRMSG ;"NOT READY".
        CNZ PMSG
        MOV A,D ;GET ERROR BITS.
        ANI 10H ;IF BIT 4 IS HIGH,
        LXI H,RNMSG ;PRINT "RECORD NOT FOUND"
        CNZ PMSG
        MOV A,D ;GET ERROR BITS.
        ANI 8H ;IF BIT 3 IS HIGH,
        LXI H,CRMSG ;PRINT "CRC ERROR".
        CNZ PMSG
        MOV A,D ;GET ERROR BITS.
        ANI 4H ;IF BIT 2 IS HIGH,
        LXI H,LDMSG ;PRINT "LOST DATA".
        CNZ PMSG
        MOV A,D ;GET ERROR BITS.
        ANI 1 ;IF BIT 1 IS HIGH,
        LXI H,BSYMSG ;PRINT "BUSY".
        CNZ PMSG
PERMSG: LXI H,ERRMSG ;PRINT "ERROR."
        CALL PMSG
        MOV A,D ;MOVE FLAGS TO ACC
        ANI 18H ;CRC OR RECORD NOT FOUND
        JZ RETRY
TRKCHK: MVI A,004H
        OUT DCOM ;READ ADDRESS
        IN WAIT
        IN DDATA ;TRACK ADDRESS
        STA BDTRK
CHKS2: IN WAIT ;DUMP REST OF DATA
      JM CHKS2
      LXI H,MSG5 ;HEAD ERROR MESSAGE
      CALL PMSG
      LXI H,MSG6 ;HEADINGS
      CALL PMSG
      MVI D,05H
      CALL BLNK
      LDA BDTRK ;SPACE OVER
               ;DISK TRK

```

```

MOV     D,A
CALL    BINHA           ;PRINT TRK
MVI     D,15H
CALL    BLNK           ;SPACE OVER
IN      TRACK
MOV     D,A
CALL    BINHA           ;PRINT TRK
MVI     D,13H
CALL    BLNK
LDA     SECT           ;SECTOR
MOV     D,A
RETRY:  CALL    BINHA           ;PRINT SECTO NO.
        IN      CDATA           ;CLEAR KEYBOARD
        IN      OFFH           ;READ SENSE SWITCHES
        ANI     080H           ;SWITCH 0
        JNZ     CONT
        LXI     H,MSG8
        CALL    PMSG           ;REQUEST INPUT
        CALL    CONIN          ;READ KEYS
        CPI     'R'           ;CHECK FOR R
        JZ      FIX
        CPI     'C'           ;CHECK FOR C
        JZ      CONT
        HLT
FIX:    MVI     A,01           ;SET REPETE FLAG
        CALL    CRLF
        CALL    CRLF
        RET
CONT:   CALL    CRLF
        CALL    CRLF
        RET
;
;
;
WDONE:  IN      DSTAT          ;READ DISK STATUS.
        ANI     0FDH           ;LOOK AT THESE BITS.
        MOV     D,A           ;SAVE STATUS BITS
PROCR:  RZ
WERRO:  LXI     H,WMSG         ;RETURN IF NO ERR.
        CALL    PMSG          ;PRINT "WRITE ".
        MOV     A,D           ;GET ERROR BITS.
        ANI     40H           ;LOOK AT BIT 6.
        LXI     H,WPMMSG      ;PRINT "PROTECT ".
        CNZ     PMSG
        MOV     A,D           ;GET ERROR BITS.
        ANI     20H           ;LOOK AT BIT 5.
        LXI     H,WFMSG       ;PRINT "FAULT ".
        CNZ     PMSG
        JMP     ERMSG1        ;DO COMMON MESSAGES.
;
; MOVE THE HEAD TO THE TRACK IN REGISTER A.
;
SEEK:   OUT     DDATA          ;TRACK TO DATA REGISTER.
BUSY:   IN      DSTAT          ;READ DISK STATUS.
        RRC                 ;LOOK AT BIT 0.
        JC      BUSY          ;WAIT TILL NOT BUSY.
        MVI     A,12H         ;SET FOR 10 MS STEP.
        OUT     DCOM          ;ISSUE SEEK COMMAND.
        IN      WAIT          ;WAIT FOR INTREQ.
        IN      DSTAT          ;READ STATUS.
        ANI     91H           ;LOOK AT BITS.
        MOV     D,A           ;SAVE STATUS
        RZ                  ;RETURN IF NO ERROR
        LXI     H,SKMSG        ;PRINT "SEEK ".
        JMP     ERMSG         ;DO COMMON ERR MESSAGES.
;
; PRINT THE MESSAGE AT H&L UNTIL A ZERO.
;
PMSG:   MOV     A,H           ;GET A CHARACTER.
        ORA     A             ;IF IT'S ZERO,
        RZ                  ;RETURN.
        MOV     C,A           ;OTHERWISE,
        CALL    CONOT         ;PRINT IT.
        INX     H             ;INCREMENT H&L,
        JMP     PMSG          ;AND GET ANOTHER.

```

68

Section 10

Unibus Port Test

10.1 General Description

The Unibus Port Test checks the S-100 to Unibus adapter and the Unibus data and address lines.

10.2 Program Details

The Unibus is a wired-or bus and any device on the bus can pull a data or address line low (logical 1). When not in use all data and address lines should be high (logical 0). The port test continuously reads the Unibus and in a cyclic fashion, sets a line low and then verifies that this and only this line is low (logical 1). The test then proceeds to the next line and tests it until all data and address lines are tested. The port test uses the Unibus adapter to read and write to the Unibus lines, verifying their operation. If an error is detected, the Unibus adapter port number and the actual and expected bit patterns are printed on the console. Figure 10.1 shows the relationship between the IMSAI adapter port numbers and Unibus data and address lines.

This test has proven particularly helpful in locating problems arising due to misaligned cards in the Unibus card cage where address and data lines become shorted together.

10.3 Operation

The Unibus Port Test is self-contained and requires no operator responses once running. The test checks all the data lines, reports any errors, and returns to the monitor automatically. There are no console or sense switch inputs.

The Unibus Port test is stored in programmable memory on the diagnostic memory board. The program can be run under the diagnostic operating system test controller, or it can be started from the IMSAI 8080 front panel at a starting address of 'D000' Hex.

The port test is also stored on floppy disk and it can be invoked by the CPM operating system under file name 'UBPORT.COM'.

Unibus Signal		Port No.								
HI Address A 16:09	<table><tr><td>A16</td><td>A15</td><td>A14</td><td>A13</td><td>A12</td><td>A11</td><td>A10</td><td>A01</td></tr></table>	A16	A15	A14	A13	A12	A11	A10	A01	(10) ₁
A16	A15	A14	A13	A12	A11	A10	A01			
LO Address A 08:01	<table><tr><td>A08</td><td>A07</td><td>A06</td><td>A05</td><td>A04</td><td>A03</td><td>A02</td><td>A01</td></tr></table>	A08	A07	A06	A05	A04	A03	A02	A01	(11) ₁
A08	A07	A06	A05	A04	A03	A02	A01			
HI Data D 15:08	<table><tr><td>D15</td><td>D14</td><td>D13</td><td>D12</td><td>D11</td><td>D10</td><td>D09</td><td>D08</td></tr></table>	D15	D14	D13	D12	D11	D10	D09	D08	(12) ₁₆
D15	D14	D13	D12	D11	D10	D09	D08			
LO Data D 07:00	<table><tr><td>D07</td><td>D06</td><td>D05</td><td>D04</td><td>D03</td><td>D02</td><td>D01</td><td>D00</td></tr></table>	D07	D06	D05	D04	D03	D02	D01	D00	(13) ₁₆
D07	D06	D05	D04	D03	D02	D01	D00			
Control 1	<table><tr><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>Ssn</td><td>C1</td><td>Msn</td></tr></table>	---	---	---	---	---	Ssn	C1	Msn	(14) ₁₆
---	---	---	---	---	Ssn	C1	Msn			
Control 2	<table><tr><td>Int</td><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>---</td><td>Grt Req</td></tr></table>	Int	---	---	---	---	---	---	Grt Req	(15) ₁₆
Int	---	---	---	---	---	---	Grt Req			

Ssn = Slave Sync (input)
 C1 = Read/ write (output)
 Msn = Master sync (output)
 Int = Initiate (output)
 Grt = Grant (input)
 Req = Request unibus (output)

Fig 10.1 Unibus Adapter Ports

```

      ORG      100H
      UNIBUS PORT TEST
; DISC VERSION 24 MAY 79  B. DONLAN
;
;
ENTRY: LXI      H,ENTRY
      PUSH     H
      DI
      LXI      H,MSG1      ;OPENING MESSAGE
      CALL     PMSG

; BEGINING OF TEST
      MVI      A,01H
      MVI      C,10H      ;PORT UNDER TEST
PORT10: OUT     10H
      MOV      B,A        ;SAVE TEST PATTERN
      IN       10H        ;READ BUSS
      CMP      B          ;COMPARE
      CNZ      ERR        ;CALL IF IN ERROR
      RLC
      JNC      PORT10     ;TEST FOR A COMPLETE CICLE
;
      MVI      A,01H
      MVI      C,11H      ;PORT 11
PORT11: OUT     11H
      MOV      B,A        ;SAVE PATTERN
      IN       11H        ;READ BUSS
      CMP      B          ;C OMPARE
      CNZ      ERR        ;CALL IF ERROR
      RLC
      JNC      PORT11
;
;
      MVI      A,01H
      MVI      C,12H      ;PORT12
PORT12: OUT     12H
      MOV      B,A        ;SAVE TEST PATERN
      IN       12H
      CMP      B
      CNZ      ERR
      RLC
      JNC      PORT12
;
;
      MVI      A,01H
      MVI      C,13H
PORT13: OUT     13H
      MOV      B,A
      IN       13H
      CMP      B
      CNZ      ERR
      RLC
      JNC      PORT13
;
;
;
      LXI      H,MSG4      ;FINISHED MESSAGE
      CALL     PMSG
      JMP      RENT        ;RETURN TO MONITOR

```

```

ERR:  PUSH    B                ;SAVE ERROR PATTERN
      PUSH    A                ;SAVE TEST PATTERN
      MOV     D,C
      LXI     H,MSG2
      CALL    PMSG
      CALL    BINHA            ;PRINT 2 DIGITS
      LXI     H,MSG0            ;ERROR MESSAGE
      CALL    PMSG
      MOV     A,B                ;LOAD TEST PATTERN
      CALL    BITS            ;PRINT TEST PATTERN
      LXI     H,MSG3            ;MORE TEXT
      CALL    PMSG
      POP     A
      CALL    BITS            ;PRINT ERROR PATTERN
      POP     B                ;RESTORE B AND C
      MOV     A,B                ;MOVE TEST PATTERN TO A
      STC
      CMC
      RET

```

```

;
;
MSG1: DB    0AH,0AH,0DH,'UNIBUS PORT TEST',0
MSG2: DB    0AH,0AH,0DH,'ERROR PORT NO.',0
MSG0: DB    0AH,0DH,'TEST PATTERN',0
MSG3: DB    0AH,0DH,'ACTUAL PATTERN',0
MSG4: DB    0AH,0AH,0DH,'END OF TEST',0
RENT  EQU    0000H            ;MONITOR ENTRY
;
;
;

```

```

; diagnostic input output routines
; for Brian Donlan 26 Feb 79

```

```

CSTAT EQU 3                ;CONSOLE STATUS PORT.
CCOM EQU 3                ;CONSOLE COMMAND PORT.
CDATA EQU 2                ;CONSOLE DATA PORT.
CKBR EQU 00000010B        ;KEYBOARD READY BIT.
CPR EQU 00000001B        ;PRINT READY BIT.
CNULL EQU 1                ;CONSOLE NULL COUNT.

```

```

;
; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;

```

```

CONOT: MVI    A,0DH        ;IF IT'S A CR,
      CMP     C            ;THEN HOP OUT
      JZ      CONUL        ;TO NULL ROUTINE.
CONOT1: IN     CSTAT        ;READ CONSOLE STATUS.
      ANI     CPR          ;IF NOT READY,
      JZ      CONOT1       ;READY WHEN HIGH.
      MOV     A,C          ;GET CHARACTER.
      OUT     CDATA        ;PRINT IT.
      RET
CONUL:  PUSH   B            ;RETURN.
      ;SAVE B&C.
      MVI     B,CNULL      ;GET NULL COUNT.
CONUL1: CALL    CONOT1      ;PRINT CR.
      MVI     C,0          ;GET NULL CHAR.
      DCR     B            ;DECREMENT COUNTER.
      JNZ     CONUL1       ;DO NEXT NULL.
      POP     B            ;RESTORE B&C.
      MOV     A,C          ;RESTORE A.
      RET                ;RETURN.

```


Section 11

Unibus Communication Test

11.1 General Description

The Unibus Communication Test is one of the most versatile tests written for this system. This test allows an operator to transfer a data word either to or from the console and any device on the Unibus. Error checking is accomplished to monitor the transfer and report any bus errors.

11.2 Program Details

The Unibus Communication Test is completely interactive with the operator responding to the computer requests for data. The test program first requests the transfer mode. There are three valid responses to this request:

I- Input, transfer from Unibus device to console

Control C-exit

O-Output, transfer users data word to Unibus device.

If the output mode is selected, the user is next prompted for the 4 Hex digit data word to be transferred. Error checking is performed on the bus status and operation, but not on the transferred data. A timer is incorporated in the transfer program which will time-out and report on errors if the selected device has made no response after approximately 10 milliseconds. Errors such as bus busy and bus mastership conflicts are also reported

in error messages.

11.3 Operation

Since the Unibus Communication Test is interactive, the operator need only respond to the computer's request. All data and address words are 4 hex digits long with no carriage return used.

The communication test is stored in the programmable memory on the diagnostic memory board. The program can be run under the diagnostic operating system controller, or it can be started directly from the IMSAI 8080 front panel at a starting address of 'D100' Hex.

The communication test is also stored on floppy disk and it can be invoked by the CPM operating system under file name 'UBCOMM.COM'.

Unibus Address	Device
FE00	AP-120B Formatter
FE20	AP-120B Word Count
FE21	AP-120B Host Memory Address
FE22	AP-120B DMA Control
FE23	AP-120B AP Memory Address
FE24	AP-120B Panel Switches
FE25	AP-120B Panel Functions
FE26	AP-120B Panel Lites
FE27	AP-120B Reset
FFE8	Filter Control Register
F800	Front Panel
F801	Front Panel
F802	Front Panel
0003	Data Acquisition Module
2102-21FF	Display Memory

Table 11.1
Unibus Addresses

```

; UNIBUS COMMUNICATION TEST
; DISC VERSION 24 MAY 79 B. DONLAN
;
; RENT: EQU 0
;
; ORG 100H
ENTRY3: LXI H, ENTRY3
; PUSH H
; LXI H, MSG5 ; OPENNING MESSAGE
; CALL PMSG
; CALL BBIN ; GET HEX CHAR
; PUSH D ; SAVE ADDRESS
; LXI H, MSG6 ; REQUEST MODE
; CALL PMSG
TRYGN: CALL CONIN
; CPI 'I'
; JZ PUTIN ; JUMP IF INPUT MODE
; CPI 03H ; TEST IF CONTROL C
; JZ RENT ; RETURN TO MONITOR
; CPI 'O'
; JNZ QUEST
;
; OUTPUT MODE
;
; PUTOUT: LXI H, MSG11 ; OUTPUT MESSAGE
; CALL PMSG
; CALL BBIN ; GET DIGITS TO OUTPUT
; POP B ; RESTORE ADDRESS TO REG B & C
; CALL DATA0 ; UNIBUSS DRIVER
; JMP DONE
;
; PUTIN: LXI H, MSG9 ; INPUT MESSAGE
; CALL PMSG
; POP B ; RESTORE ADDRESS TO B & C
; CALL DATA1 ; UNIBUS INPUT ROUTINE
; CALL BINB ; PRINT DATA FROM BUSS
; JMP DONE
;
;
; DONE: LXI H, MSG10 ; PRINT END OF TEST
; CALL PMSG
; JMP ENTRY3
;
;
; QUEST: LXI H, MSG7 ; ??
; CALL PMSG
; JMP TRYGN
;
;
; MSG5: DB 0AH, 0AH, 0DH, 'UNIBUS COMMUNICATION TEST'
; DB 0AH, 0DH, 'ENTER UNIBUS ADDRESS ', 0
; MSG6: DB 0AH, 0DH, 'INPUT (I), OUTPUT (O), EXIT (CONTROL C) ? ', 0
; MSG7: DB 0AH, 0DH, ' ', 0
; MSG11: DB 0AH, 0DH, 'ENTER DATA TO OUTPUT IN 4 HEX DIGITS ', 0
; MSG9: DB 0AH, 0DH, ' DATA FROM BUS ', 0
; MSG10: DB 0AH, 0DH, 'TRANSFER COMPLETE', 0

```

```

; diagnostic input output routines
; for Brian Donlan 25 Feb 79

CSTAT EQU 3 ;CONSOLE STATUS PORT.
CCOM EQU 3 ;CONSOLE COMMAND PORT.
CDATA EQU 2 ;CONSOLE DATA PORT.
CKBR EQU 00000010B ;KEYBOARD READY BIT.
CPR EQU 00000001B ;PRINT READY BIT.
CNULL EQU 1 ;CONSOLE NULL COUNT.

; CHECK CONSOLE INPUT STATUS.
;
CONST: IN CSTAT ;READ CONSOLE STATUS.
      ANI CKBR ;LOOK AT KB READY BIT.
      MVI A,0 ;SET A=0 FOR RETURN.
      RZ ;NOT READY WHEN ZERO.
      CMA ;IF READY A=FF.
      RET ;RETURN FROM CONST.

; READ A CHARACTER FROM CONSOLE.
;
CONIN: IN CSTAT ;READ CONSOLE STATUS.
      ANI CKBR ;IF NOT READY,

      JZ CONIN ;READY WHEN HIGH.
      IN CDATA ;READ A CHARACTER.
      OUT CDATA
      ANI 7FH ;MAKE MOST SIG. BIT = 0.
      RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CONOT: MVI A,0DH ;IF IT'S A CR,
      CMP C ;THEN HOP OUT
      JZ CONUL ;TO NULL ROUTINE.
CONOT1: IN CSTAT ;READ CONSOLE STATUS.
      ANI CPR ;IF NOT READY,
      JZ CONOT1 ;READY WHEN HIGH.
      MOV A,C ;GET CHARACTER.
      OUT CDATA ;PRINT IT.
      RET ;RETURN.
CONUL: PUSH B ;SAVE B&C.
      MVI B,CNULL ;GET NULL COUNT.
CONUL1: CALL CONOT1 ;PRINT CR.
      MVI C,0 ;GET NULL CHAR.
      DCR B ;DECREMENT COUNTER.
      JNZ CONUL1 ;DO NEXT NULL.
      POP B ;RESTORE B&C.
      MOV A,C ;RESTORE A.
      RET ;RETURN.

;
; PRINT MESSAGE UNTIL ZERO
; MESSAGE ADDRESS REG H & L
;
;
PMSG: MOV A,M ;GET CHAR
      ORA A ;IS IT A ZERO
      RZ
      MOV C,A ;OTHERWISE PRINT
      CALL CONOT
      INX H ;INC ADDRESS
      JMP PMSG

```

```

.....
PRINT 8 BIT WORD IN BINARY FORMAT
INPUT: DATA IN REG A
.....

```

```

BITS:  MOV      B,A      ; DATA
OVER:  MVI      A,80H    ; MASK
      MOV      C,30H
      ANA      E,A      ; STORE MASK
      JZ       PRNT      ; AND WITH MASK
      MVI      C,31H    ; JUMP IF ZERO
PRNT:  CALL     CONOT
      ANA      B
      MOV      A,E      ; ZERO CARRY
      RAR      ; LOAD MASK
      JNC      OVER
      RET

```

```

;;
BLNK:  MVI      C,20H
LP1:   CALL     CONOT1    ;PRINT BLANKS, # IN REG. D
      DCR      D
      JNZ      LP1
      RET

```

```

.....
OUTPUTS 2 HEX DIGITS IN ASCII
FROM REG D
.....

```

```

BINHA: MOV      A,D
      RAR
      RAR
      RAR
      RAR
      CALL     BIN1
      MOV      C,A
      CALL     CONOT
      MOV      A,D
      CALL     BIN1
      MOV      C,A
      CALL     CONOT
      RET

```

```

.....
OUTPUTS FOUR HEX DIGITS IN ASCII
ENTER WITH DATA IN REG PAIR E AND D
.....

```

```

BINB:  CALL     BINHA
      MOV      D,E
      CALL     BINHA
      RET

```

```

.....
CONVERTS HEX TO ASCII
INPUT: 4 BITS HEX REG A
OUTPUT: 8 BIT ASCII REG A
.....

```

```

BIN1:  ANI      0FH
      ADI      30H
      CPI      3AH
      RC
      ADI      07H
      RET

```



```

; ROUTINE TO OUTPUT A 16 BIT WORD ON THE UNIBUS
; REG B = 15H, 14H, 13H, 12H, 11H, 10H, 9H, 8H, 7H, 6H, 5H, 4H, 3H, 2H, 1H, 0H
; REG D = 0H, 1H, 2H, 3H, 4H, 5H, 6H, 7H, 8H, 9H, 10H, 11H, 12H, 13H, 14H, 15H
;
DATO:  MVI    A,OFFH
      STA    BIZCNT
BIZLP2: IN     14H
      ANI    04H
      JNZ    BBUSY2
;
      MOV    A,B           ;OUTPUT HIGH ADDRESS
      OUT    10H
      MOV    A,C           ;OUTPUT LOW ADDRESS
      OUT    11H
      MOV    A,D           ;OUTPUT HIGH DATA
      OUT    12H
      MOV    A,E           ;OUTPUT LOW DATA
      OUT    13H
      MVI    A,02H        ;OUTPUT C1=1
      OUT    14H
;
      MVI    A,03H        ;OUTPUT MSYN=1
      OUT    14H
;
SYNLP2: MVI    A,OFFH
      STA    SYNCNT
DCLoop: IN     14H         ;CHECKS FOR SSYN
      OUT    OFFH
      ANI    04H         ;TO GET ASSERTED
      JZ     NOSYN2
;
      SUB    A
      OUT    14H         ;CLEARS MSYN AND C1
      OUT    10H
      OUT    11H         ;CLEARS EVERYTHING
      OUT    12H         ;OUTPUT TO THE BUS
      OUT    13H
      IN     OFFH        ;READ SENSE SWITCH
      ANI    080H
      JNZ    DATO        ;LOOP IF UP
      RET
;
;
;
;
GETBUS: MVI    A,OFFH
      STA    GETCNT
      MVI    A,01H
      OUT    15H
LOOP:   IN     15H
      ANI    01H
      JZ     NOGET
      RET
;
; ON-LINE UNIBUS DIAGNOSTICS
; BY BRIAN DONLAN 24 APR 79
;
BBUSY1: LDA    BIZCNT      ;LOOP COUNT
      DCR    A
      STA    BIZCNT      ;NEW COUNT
      JNZ    BIZLP1      ;JUMP IF STILL COUNT
      LXI    H,ERMSG2
      CALL   PMSG        ;DISPLAY ERROR MESSAGE
      JMP    ENTRY3
;
;
BBUSY2: LDA    BIZCNT
      DCR    A
      STA    BIZCNT
      JNZ    BIZLP2
      LXI    H,ERMSG2
      CALL   PMSG
      JMP    ENTRY3

```

[illegible]

Section 12

Unibus Snapshot

12.1 General Description

The Unibus Snapshot program is not a test, but a routine which presents on the console device the status of the Unibus data, address and some status lines at the time the routine was executed.

12.2 Program Details

The Unibus Snapshot routine can either be run as a stand-alone program, or by using an alternate entry point as a subroutine which can be called by any user program. The Snapshot routine presents in Hex, the status of the Unibus address and data lines. The routine also presents in binary, the Unibus slave sync and bus grant lines. The routine also has the capability to present the master sync line, but this requires some minor hardware changes which are not available at this time.

12.3 Operation

Snapshot requires no operator responses or input to run and upon completion of the listing, execution is returned to the monitor or program from which it came.

Snapshot is stored in programmable memory on the diagnostic memory board. The program can be run under the diagnostic operating system test controller, or it can be run from the IMSAI 8080 front panel at a starting address.

AD-A092 441

AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
DIAGNOSTIC SOFTWARE DEVELOPMENT FOR A REAL-TIME SPECTRAL ANALYS--ETC(U)
APR 80 B J DONLAN
AFIT-CI-80-9T

F/6 14/2

UNCLASSIFIED

NL

2 of 2

NO
000041



END

DATE

FILED

1-81

DTIC

of 'D500' Hex.

Snapshot is also stored on floppy disk and it can be invoked by the CPM operating system under file name 'SNAPST.COM'.

Snapshot is also available as a subroutine which can be called from user program. The PROM version subroutine entry point is at 'D508' Hex.

```

; UNIBUS SNAP SHOT ROUTINE
ORG      100H
ENTRY4: LXI      H,FINIS
        PUSH     H
        DI
        CALL     INITA          ;RESET I/O
;
;SUBROUTINE ENTRY POINT
ENTRY5: LXI      H,MSG12
        CALL     PMSG
        IN       10H           ;HIGH ADDRESS
        MOV      D,A           ;SAVE IN D
        IN       11H           ;LOW ADDRESS
        MOV      E,A
        CALL     BINB          ;PRINT UNIBUS ADDRESS
;
        MVI      D,8H
        CALL     BLNK          ;SPACE OVER
        IN       12H           ;HIGH DATA
        MOV      D,A
        IN       13H           ;LOW DATA BITS
        MOV      E,A
        CALL     BINB          ;PRINT UNIBUS DATA BITS
        CALL     BLNK          ;SPACE OVER
;
        IN       14H           ;STATUS PORT
        ANI      04H
        JZ       NOSIS         ;FIND SLAVE SYN
        MVI      C,'1'
        JMP      OUTSIS
NOSIS:  MVI      C,'0'
OUTSIS: CALL     CONOT          ;PRINT SLAVE SYN
        MVI      D,09H
        CALL     BLNK          ;SPACE OVER
;
        IN       15H           ;STATUS PORT
        ANI      01H
        JZ       NOBUS         ;BUS GRANT
        MVI      C,'1'
        JMP      OUTBUS
NOBUS:  MVI      C,'0'
OUTBUS: CALL     CONOT          ;PRINT BUS GRANT
        MVI      D,08H
        CALL     BLNK          ;SPACE OVER
;
        IN       14H
        ANI      00H
        JZ       NOMSYN
        MVI      C,'1'
        JMP      OUTMSN
NOMSYN: MVI      C,'0'
OUTMSN: CALL     CONOT          ;PRINT MSYN
        RET
FINIS:  JMP      FINIS
;
MSG12:  DB       0AH,0AH,0DH,'UNIBUS SNAP-SHOT '
        DB       0AH,0DH,'ADDRESS DATA SSYN GRANT MSYN'
        DB       0AH,0DH,' ',0
;
; for brian donlan 26 feb 79

CSTAT EQU 3 ;CONSOLE STATUS PORT.
CCOM EQU 3 ;CONSOLE COMMAND PORT.
CDATA EQU 2 ;CONSOLE DATA PORT.
CKBR EQU 00000010B ;KEYBOARD READY BIT.
CPTR EQU 00000001B ;PRINT READY BIT.
CNULL EQU 1 ;CONSOLE NULL COUNT.

```

```

; CHECK CONSOLE INPUT STATUS.
;
CONST: IN  CSTAT      ;READ CONSOLE STATUS.
      ANI  CKBR      ;LOOK AT KB READY BIT.
      MVI  A,0        ;SET A=0 FOR RETURN.
      RZ             ;NOT READY WHEN ZERO.
      CMA             ;IF READY A=FF.
      RET             ;RETURN FROM CONST.

; READ A CHARACTER FROM CONSOLE.
;
CONIN: IN  CSTAT      ;READ CONSOLE STATUS.
      ANI  CKBR      ;IF NOT READY,
      JZ   CONIN      ;READY WHEN HIGH.
      IN  CDATA      ;READ A CHARACTER.
      OUT CDATA
      ANI  7FH        ;MAKE MOST SIG. BIT = 0.
      RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CONOT: MVI  A,0DH      ;IF IT'S A CR,
      CMP  C           ;THEN HOP OUT
      JZ   CONUL       ;TO NULL ROUTINE.
CONOT1: IN  CSTAT      ;READ CONSOLE STATUS.
      ANI  CPTIR      ;IF NOT READY,
      JZ   CONOT1      ;READY WHEN HIGH.
      MOV  A,C         ;GET CHARACTER.
      OUT  CDATA      ;PRINT IT.
      RET             ;RETURN.
CONUL: PUSH B          ;SAVE B&C.
      MVI  B,CNULL     ;GET NULL COUNT.
CONUL1: CALL CONOT1    ;PRINT CR.
      MVI  C,0         ;GET NULL CHAR.
      DCR  B           ;DECREMENT COUNTER.
      JNZ  CONUL1      ;DO NEXT NULL.
      POP  B           ;RESTORE B&C.
      MOV  A,C         ;RESTORE A.
      RET             ;RETURN.

;
; PRINT MESSAGE UNTIL ZERO
; MESSAGE ADDRESS REG H & L
;
;
PMSG:  MOV  A,M        ;GET CHAR
      ORA  A           ;IS IT A ZERO
      RZ
      MOV  C,A         ;OTHERWISE PRINT
      CALL CONOT
      INX  H           ;INC ADDRESS
      JMP  PMSG

;
; PRINT 8 BIT WORD IN BINARY FORMAT
; INPUT: DATA IN REG A
;
;
BITS:  MOV  B,A        ; DATA
      MVI  A,80H      ; MASK
OVER:  MVI  C,30H
      MOV  E,A        ; STORE MASK
      ANA  B          ; AND WITH MASK
      JZ   PRNT       ; JUMP IF ZERO
      MVI  C,31H

```

```

PRNT:  CALL    CONOT
      ANA      B      ; ZERO CARRY
      MOV      A,E    ; LOAD MASK
      RAR
      JNC      OVER
      RET

;;
BLNK:  MVI      C,20H      ;PRINT BLANKS, # IN REG. D
LP31:  CALL     CONOT1
      DCR      D
      JNZ      LP31
      RET

;
;
;   OUTPUTS 2 HEX DIGITS IN ASCII
;   FROM REG D
;
;
;
BINHA: MOV      A,D
      RAR
      RAR
      RAR
      RAR
      CALL     BIN1
      MOV      C,A
      CALL     CONOT
      MOV      A,D
      CALL     BIN1
      MOV      C,A
      CALL     CONOT
      RET

;
;
;   OUTPUTS FOUR HEX DIGITS IN ASCII
;   ENTER WITH DATA IN REG PAIR E AND D
;
;
;
BINB:  CALL     BINHA
      MOV      D,E
      CALL     BINHA
      RET

;
;
;   CONVERTS HEX TO ASCII
;   INPUT: 4 BITS HEX REG A
;   OUTPUT: 8 BIT ASCII REG A
;
;
;
BIN1:  ANI      0FH
      ADI      30H
      CPI      3AH
      RC
      ADI      07H
      RET

;
;
;   INPUTS 4 DIGITS FROM CONSOLE
;   RETURN: 4 HEX DIGITS IN REG E-D
;
;
;
BBIN:  CALL     CONIN
      CALL     ANS1
      RAL
      RAL

```


Section 13

Diagnostic Memory Board

13.1 General Description

In order to have all diagnostic programs available for execution even in the event of a disk failure, all diagnostic programs were placed in programmable read-only memories. A small operating system was also included in the prom memory. This enabled all programs to be resident and not require any disk loading before execution. A 16K byte read-only memory board was added to the IMSAI 8080 computer system to hold these programs.

13.2 Detailed Description

The memory board decided upon was in a kit manufactured by SSM Microcomputer products. This board has a capacity of 16K bytes of memory stored in Intel 2708 memory chips. The memory board can be assigned to any 16K memory block area but for this application it was placed at the very top of the memory address range. The board was set to occupy from 'C000' to 'FFFF' hex. Presently, only 10K of the possible 16K is used for diagnostics, leaving the remainder for future expansion.

The memory board is S-100 Bus compatible and receives all its power from the Bus.

'C000'	Memory Test
'C290'	Mini Memory test 0 to 100 Hex version
'C800'	Formatted Disc test
'CD80'	Disc Full track write routine
'CE40'	Disc Full track write routine
'D000'	Unibus Port test
'D100'	Unibus Communication test
'D500'	Unibus Snap Shot routine
'D600'	Mini-Memory test 8K version
'D700'	Mini-Memory test 24K version
'F000'	Operating system
'F800'	Help program
'FD00'	test controller
'E000'	Color graphic test (future options)

table 13.1 Memory map

Section 14

Conclusion

This report has described the design and development of a diagnostic system for a real-time spectral analysis system.

The developed software has been verified and tested. In fact, many of the tests were used by the author and other members of the project team to keep the system operational, enabling further development to proceed. The diagnostic system also proved very helpful in testing new hardware designs and components. For example, hardware modifications were performed on the color graphics display, giving it added capabilities. During the development of a display test, the newly written programs pointed out unknown degradations of the original capabilities.

Since the research personnel working on and with the spectral analysis system is always changing, every effort was made to make the tests easy to run and error notifications self-explanatory. This is unfortunate since the tests contain a wealth of information about the system state and a more experienced user can interpret this information.

The structure of the diagnostic system is such that many of the functions are performed using subroutines. These subroutines can be used by any program

and could be of great use in the future for I/O functions on on-line continuous verification.

Unfortunately, as of this writing, no further funding or development effort is programmed of the system. With the exception of the memory diagnostics, all the tests are very specific and will be of little value for use in a general purpose microcomputer system.

The diagnostic system was designed with expansion in mind. The programmable memory board contains room for many more tests and the test directory can easily be expanded.

Section 15

Literature

1. Lee, Bock w., Two Additions to the Z-80 RAM Test, Dr. Dobb's Journal, Jan 1979.
2. Knaizuk, John, Hartmann, C.P.R., An Algorithm for Testing Random Access Memories, IEEE Trans. on Computers, April 1977.
3. Intel Corp., Intel 8080 Microcomputer User's Manual, Intel Corp., Santa Clara, CA., 1975.
4. IMSAI Assco., IMSAI 8080 Microcomputer User's Manual Set, IMSAI Manf. Corp., San leandro, CA., 1976.
5. Ng, Howard, Disk System for a Microcomputer Controlled Spectra Analysis System, Master's Project, RPI, 1978
6. Tarbell Electronics, The Tarbell Floppy Disk Interface, Tarbell Electronics, Carson, CA., 1977
7. Dykstra, K.U., Input Module and Front Panel Design for a Real Time Spectral Analysis System, Master's Project, RPI, 1978.
8. Digital Research, CP/M Technical Manual Set, Digital Research, Pacific Grove, CA., 1976, 1977.
9. Sparrell, D.K., Software Development for a Real-Time Spectral Analysis System for Nonstationary Signals, Master's Project, RPI, 1978.
10. Scordo, Dominick, A Real-Time Signal Processing and Display System for Non-Stationary Signals, Master's Project, RPI, 1978.

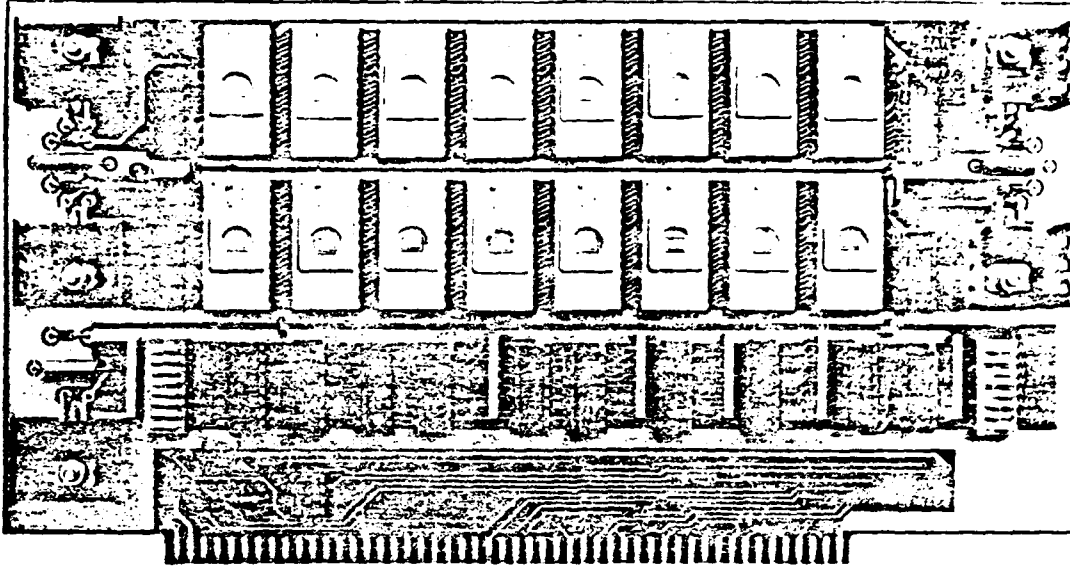
APPENDIX A

MEMORY BOARD DIAGRAMS



Microelectronics Products, 2115 West Myrtle, Santa Clara, CA 95050, (415) 251-1107

MB8A 1K TO 16K EPROM BOARD



FEATURES:

SYSTEM COMPATIBILITY

- S-100 bus computer systems.

MEMORY

- Up to 16K bytes of 2708 EPROMs (not included)
- Any unused EPROM socket will automatically disable the board for that 1K increment. For example, with 8 EPROMs it acts as an 8K board, taking up only 8K of memory address space.

ADDRESSING

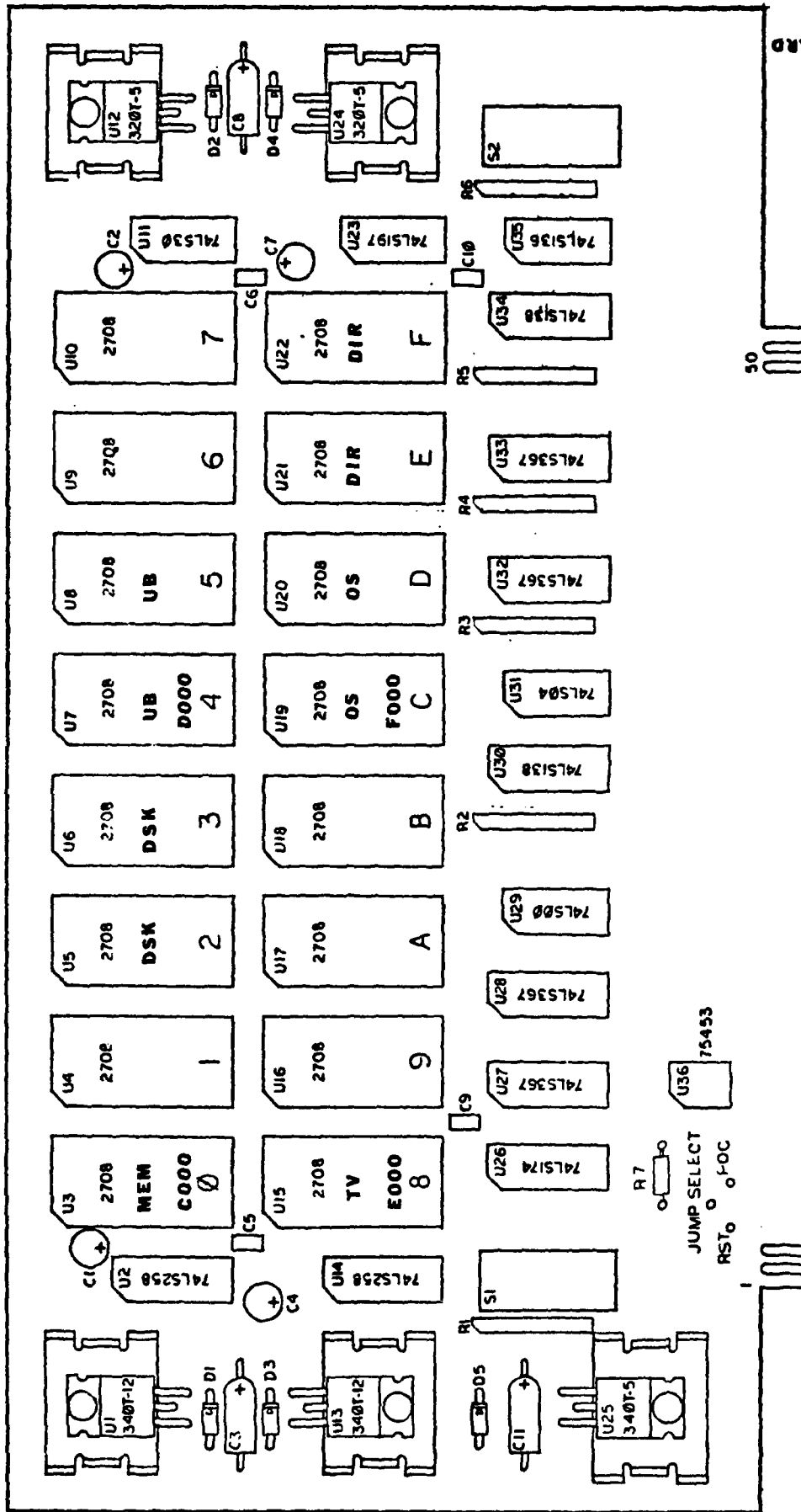
- DIP switch selection of memory address assignment in 16K byte increments.
- Magic Mapping™ allows any byte within ROM to be mixed with any similarly addressed RAM board equipped with Phantom Disable.

VECTOR JUMP

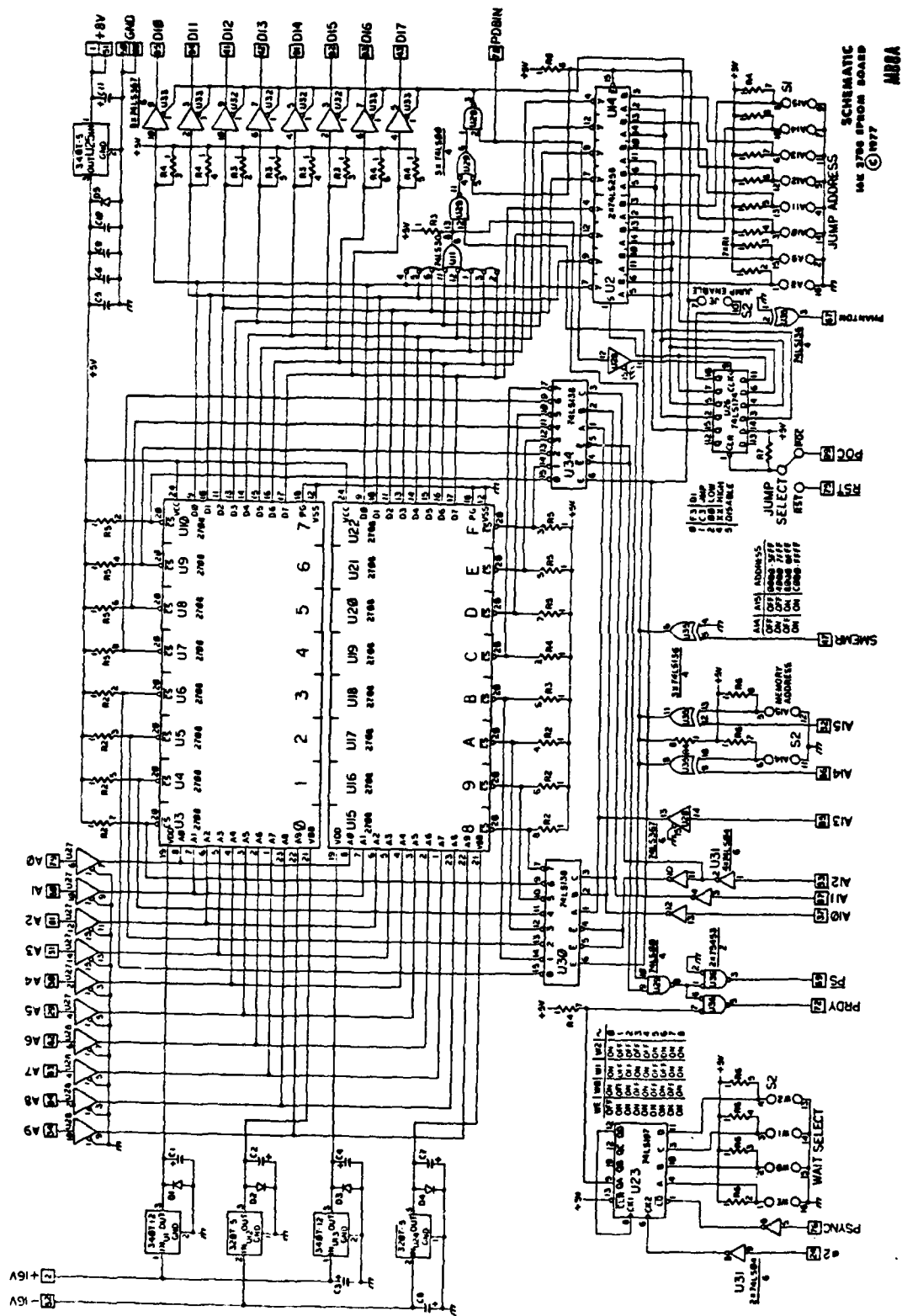
- Power-on/reset vector jump to any 256 byte increment; DIP switch addressable.
- Vector jump can be disabled.
- Vector jump requires other memory boards to be equipped with Phantom Disable.

OTHER FEATURES

- DIP switch selection of 0 to 8 wait state clock cycles, so fast or slow EPROMs can be used.
- All lines buffered, Reverse voltage protection.
- High grade glass epoxy PC board with gold plated edge connector contacts.
- Low profile sockets provided for all ICs.
- Power requirements (less EPROMs) -- +8V @ 160mA, +16V @ 10mA, -16V @ 10mA typical.



ASSEMBLY
16K 2708 EPROM BOARD
MB8A
© 1977



APPENDIX B
COMMONLY USED SUBROUTINES

Commonly Used Subroutine

The following is a listing of some subroutines which are commonly used in the diagnostic programs:

CONST 'C177'

Checks console status. Returns with zero in reg A if not ready. Returns with 'FF' in reg A if ready.

CONIN 'C180'

Reads a character from the console. The input character is returned in reg A. The input character is echoed on the console.

CONOT 'C18E'

Writes a character on the console device. The character is output from reg C.

PMSG 'C1AE'

Prints a character string on the console device. The address of the beginning of the string must be placed in reg H and L. The string is printed until a null (00) is encountered.

BITS 'C1E5'

Prints an 8 bit byte in binary format in the console device. The data word is taken from reg A.

BLNK 'C1FB'

Prints the number of blanks found in reg D.

BINB 'C21A'

Outputs four hex digits in ASCII on the console.

Enter with the data in reg E and D.

BINHA 'C205'

Outputs two hex digits in ASCII on the console.

Enter with the data in reg D.

BIN1 'C222'

Converts hex to ASCII. Input with 4 bits in reg A

Outputs with 8 bit ASCII character in reg A.

BBIN 'C22C'

Inputs 4 hex digits from the console. Converts

the ASCII characters to hex. Returns with the 4 hex digits in reg E and D.

AHS1 'C25B'

Converts ASCII to Hex. Inputs with a 8bit ASCII

character in reg A. Returns with a 4 bit hex digit in reg A.

INITA 'C264'

Initiates the SIO port. No inputs or outputs.

CRLF 'C275'

Sends one Carriage return and one line feed to the console.

LF 'C27A'

Sends one line feed to the console.

DATI 'D312'

Inputs a 16 bit word from the Unibus. Reg B and C needs the Unibus address and the data will be returned in reg D and E.

DATO 'D351'

Outputs a 16 bit word to the Unibus. Reg B and C needs the Unibus address and reg D and E needs the data.

GETBUS 'D391'

This routine gets the IMSAI master-ship of the unibus.

APPENDIX C
PROM PROGRAM LISTINGS

```

;
; MEMORY TEST
; PROM VERSION 24 MAY 79 B. DONLAN
;
C000      ORG      0C000H

0000 =    WO      EQU      00      ;TEST BYTE

C000 218000 ENTRY1: LXI      H,080H
C003 F9      SPHL
C004 2104C0 ENTRY:  LXI      H,ENTRY
C007 E5      PUSH     H
C008 3E00     MVI      A,00      ;ZERO ACC
C00A 320A00   STA      CODE
C00D CD64C2   CALL     INITA     ;RESET I/O PORT
C010 CD75C2   CALL     CRLF
C013 2120C1   LXI      H,MSG1
C016 CDAEC1   CALL     PMSG
C019 212EC1   LXI      H,MSG2
C01C CDAEC1   CALL     PMSG
C01F CD2CC2   CALL     BBIN
C022 EB      XCHG
C023 220400   SHLD     START
C026 2149C1   LXI      H,MSG3
C029 CDAEC1   CALL     PMSG
C02C CD2CC2   CALL     BBIN
C02F EB      XCHG
C030 220600   SHLD     ENADR
C033 21B9C1   LXI      H,MSG8
C036 CDAEC1   CALL     PMSG
C039 DB02     IN       CDATA     ;RESET IO FLAG

;
; BEGIN: MVI      C,W0      ;LOAD TEST BYTE
C03B 0E00     MTEST: MVI      A,02      ;LOAD TEST BYTE
C03D 3E02     STA      PART
C03F 320800   MTL0P: CALL     STUFF     ;STUFF MAJOR ALL OVER
C042 CD8EC0   MVI      A,02      ;SET TWO AS MINOR
C045 3E02     CALL     STUFM     ;STUFF MINOR
C047 CD98C0   MVI      A,02      ; SET 2 AGAIN
C04A 3E02     CALL     CHECK     ;NOW CHECK ALL LOC
C04C CDB0C0   PTCHK: LDA      PART
C04F 3A0800   DCR      A
C052 3D      STA      PART      ;STORE NEW PART
C053 320800   CPI      00      ;FINISH THIS PASS ?
C056 FE00     JZ       RECYCLE   ;YES
C058 CA6AC0   CONT: MVI      A,01      ;NO CONTINUE
C05B 3E01     CALL     STUFM     ;STUFF MINOR SERT
C05D CD98C0   MOV      A,C      ;LOAD MAJOR BYTE
C060 79      CMA      A          ;COMPLIMENT MAJOR BYTE
C061 2F      MOV      C,A      ;SAVE NEW BYTE
C062 4F      XRA      A          ;ZERO OTHER TEST BYTE
C063 AF      CALL     CHECK
C064 CDB0C0   JMP      MTL0P
C067 C342C0

;
; RECYCLE:
C06A 79      MOV      A,C
C06B 320800   STA      PART      ;SAVE INVERT TB TEMP
C06E 2162C1   LXI      H,MSG4     ;END OF PAS MESSAGE
C071 CDAEC1   CALL     PMSG
C074 3A0A00   LDA      CODE      ;CHAR CODE
C077 FE03     CPI      03H      ;CONTROL C
C079 CA21FD   JZ       RENT      ;RETURN TO MONITRO
C07C B7      ORA      A          ;SET FLAGS
C07D C204C0   JNZ      ENTRY     ;STANT OVER
C080 A7      ANA      A          ;CLEAR CARRY
C081 3A0800   LDA      PART      ; RECOVER TEST BYTE
C084 B7      ORA      A
C085 CA3BC0   JZ       BEGIN
C088 17      RAL
C089 2F      CMA
C08A 4F      MOV      C,A      ;NEW TEST BYTE
C08B C33DC0   MTEST: MTEST     ; ANOTHER PAS

```

```

;
F021 =      RENT: EQU      0F021H ;MONITOR ENTRY
0004 =      START: EQU      04H ;LOC OF START ADDR
0006 =      ENADR: EQU      06H ;LOC OF END ADDR
0008 =      PART: EQU      08H ;LOC FOR PART
00CA =      CODE: EQU      0AH ;LOC FOR CODE
;
;
C08E CDDECO STUFF: CALL STASTO ;LOAD START AND END ADDR
C091 71 DOIT: MOV M,C ;STUFF MAJOR ALL OVER
C092 CD06C1 CALL HILOX ;SEE IF ALL MEM DONE
C095 C391C0 JMP DOIT ;NO KEPP ON STUFFING
;
;
C098 CDDECO STJFM: CALL STASTO ;LOAD ADDR AGAIN
C099 47 MOV B,A ;MINOR COUNTER
C09C FE00 CPI 00 ;MINOR WORD STUFF
C09E C2A6C0 JNZ HIL ;NO
C0A1 79 MINOR: MOV A,C ;MAJOR TEST BYTE
C0A2 2F CMA ;MINOR IS COMPLIMENT OF MAJOR
C0A3 77 MOV M,A ;STUFF MINOR BYTE IN MEM
C0A4 0603 MVI B,03 ;START MINOR COUNT AT 3
C0A6 CD06C1 HIL: CALL HILOX ;INC & CHK IF DONE
C0A9 05 DCR B ;DEC MINOR COUNTER
C0AA C2A6C0 JNZ HIL ;OK TO STUFF NO
C0AD C3A1C0 JMP MINOR ;YES
;
;
C0B0 CDDECO CHECK: CALL STASTO ;LOAD START AND END
C0B1 47 MOV B,A ;LOAD MINOR COUNT
C0B4 FE00 CPI 00 ;COUNT ZERO
C0B6 C2C1C0 JNZ MAJR ;NO GO TO MAJOR
C0B9 79 MINR: MOV A,C ;LOAD TEST BYTE MAJOR
C0BA 2F CMA ;MINOR IS COMPLIMENT
C0BB BE CMP M ;READ AND COMPARE MEM LOC
C0BC 0603 MVI R,03 ;MINOR COUNT AT 3
C0BE C3C3C0 JMP CKEND ;CHECK FOR ERROR OR ABORT
C0C1 79 MAJR: MOV A,C ;LOAD MAJOR TEST BYTE
C0C2 BE CMP M ;READ AND COMPARE MEM WITH MAJOR
C0C3 C5 CKEND: PUSH B ;SAVE COUNT AND MAJOR
C0C4 C4E6C0 CNZ ERR ;GO TO ERR TO PRNT IF ERROR
C0C7 C1 POP B ;RESTORE REGS
C0C8 DB03 IN CSTAT ;CHECK KEYBOARD
C0CA E602 ANI 02H
C0CC CAD4C0 JZ FIN
C0CF DB02 IN CDATA ;READ KEYS
C0D1 320A00 STA CODE
C0D4 CD06C1 FIN: CALL HILOX
C0D7 05 DCR B ;DEC MINOR COUNT
C0D8 C2C1C0 JNZ MAJR
C0DB C3B9C0 JMP MINR ;COUNT ZERO DO MINOR
;
;
CODE 2A0600 STASTO: LHLD ENADR ;LOAD END ADDR
COE1 EB XCHG ;MOVE END TO C&D
COE2 2A0400 LHLD START ;LOAD START
COE5 C9 RET

```

```

C0E6 D5      ; ERR:  PUSH  D      ;SAVE END ADDR
C0E7 F5      PUSH  PSW
C0E8 CD75C2  CALL  CRLF
C0EB 54      MOV   D,H
C0EC 5D      MOV   E,L
C0ED CD1AC2  CALL  BINB      ;OUTPUT BAD ADDR
C0F0 1608    MVI   D,08      ;SPACE COUNT
C0F2 CDFBC1  CALL  BLNK      ;SPACE OVER 8
C0F5 F1      POP   PSW
C0F6 47      MOV   B,A
C0F7 CDE5C1  CALL  BITS      ;PRINT TEST BYTE
C0FA 160A    MVI   D,0AH
C0FC CDFBC1  CALL  BLNK
C0FF 7E      MOV   A,M
C100 CDE5C1  CALL  BITS      ;PRINT BAD BYTE
C103 78      MOV   A,B      ;MOVE TEST BYTE BACK
C104 D1      POP   D      ;RESTORE END ADDR
C105 C9      RET

;
;
;
C106 F5      HILOX:  PUSH  A      ;SAVE ACC
C107 23      INX   H      ;INC CURRENT ADDR
C108 7C      MOV   A,H      ;LOAD HIGH ORDER ADDR
C109 BA      CMP   D      ;COMPARE WITH END
C10A C216C1  JNZ   DIFF      ;NO MATCH
C10D 7D      MOV   A,L      ;LOAD LOW ORDER
C10E BB      CMP   E      ;COMPARE LOW ORDERS
C10F C216C1  JNZ   DIFF      ;NO MATCH
C112 F1      POP   A      ;MATCH END
C113 33      INX   SP      ;FAKE RETURN ONE LEVEL OUT
C114 33      INX   SP
C115 C9      RET
C116 F1      DIFF:  POP   A      ;CONTINUE STUFFING
C117 C9      RET
C118 0E3F    PROB:  MVI   C,3FH ; ?
C11A CD8EC1  CALL  CONOT ;PRINT ?
C11D C304C0  JMP   ENTRY

;
;
C120 0D0A4D454DMSG1 DB 0DH,0AH,'MEMORY TEST',0
C12E 0D0A454E54MSG2 DB 0DH,0AH,'ENTER START ADDRESS',0
C149 0D0A454E54MSG3 DB 0DH,0AH,'ENTER STOP ADDRESS',0
C162 0D0A454E44MSG4 DB 0DH,0AH,'END OF PASS',0AH,0
; diagnostic input output routines
; for brian donlan 26 feb 79

0003 *      CSTAT EQU 3      ;CONSOLE STATUS PORT.
0003 *      CCOM EQU 3      ;CONSOLE COMMAND PORT.
0002 *      CDATA EQU 2      ;CONSOLE DATA PORT.
0002 *      CKBR EQU 00000010B ;KEYBOARD READY BIT.
0001 *      CPTR EQU 00000001B ;PRINT READY BIT.
0001 *      CNULL EQU 1      ;CONSOLE NULL COUNT.

; CHECK CONSOLE INPUT STATUS.
;
C177 DB03    CONST:  IN   CSTAT      ;READ CONSOLE STATUS.
C179 E602    ANI   CKBR      ;LOOK AT KB READY BIT.
C17B 3E00    MVI   A,0      ;SET A=0 FOR RETURN.
C17D C8      RZ           ;NOT READY WHEN ZERO.
C17E 2F      CMA          ;IF READY A=FF.
C17F C9      RET          ;RETURN FROM CONST.

```

```

; READ A CHARACTER FROM CONSOLE.
;
C180 DB03      CONIN:  IF  CSTAT      ;READ CONSOLE STATUS.
C182 E602      A.I  CKBR      ;IF NOT READY,

C184 CA80C1    JZ  CONIN      ;READY WHEN HIGH.
C187 DB02      IN  CDATA      ;READ A CHARACTER.
C189 D302      OUT CDATA
C18B E67F      ANI  7FH      ;MAKE MOST SIG. BIT = 0.
C18D C9        RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
C18E 3E0D      CONOT:  MVI  A,ODH      ;IF IT'S A CR,
C190 B9        CMP  C          ;THEN HOP OUT
C191 CA9FC1    JZ  CONJL      ;TO NULL ROUTINE.
C194 DB03      CONOT1: IN  CSTAT      ;READ CONSOLE STATUS.
C196 E601      ANI  CPTR      ;IF NOT READY,
C198 CA94C1    JZ  CONOT1     ;READY WHEN HIGH.
C19B 79        MOV  A,C        ;GET CHARACTER.
C19C D302      OUT CDATA      ;PRINT IT.
C19E C9        RET           ;RETURN.
C19F C5        CONJL:  PUSH B      ;SAVE B&C.
C1A0 0601      MVI  B,CNULL     ;GET NULL COUNT.
C1A2 CD94C1    CONJL1: CALL CONOT1 ;PRINT CR.
C1A5 0E00      MVI  C,0        ;GET NULL CHAR.
C1A7 05        DCR  B          ;DECREMENT COUNTER.
C1A8 C2A2C1    JNZ CONJL1     ;DO NEXT NULL.
C1AB C1        POP  B          ;RESTORE B&C.
C1AC 79        MOV  A,C        ;RESTORE A.
C1AD C9        RET           ;RETURN.

;
;
; PRINT MESSAGE UNTIL ZERO
; MESSAGE ADDRESS REG H & L
;
;
C1AE 7E        PMSG:  MOV  A,M      ;GET CHAR
C1AF B7        ORA  A          ;IS IT A ZERO
C1B0 C8        RZ
C1B1 4F        MOV  C,A        ;OTHERWISE PRINT
C1B2 CD8EC1    CALL CONOT
C1B5 23        INX  H          ;INC ADDRESS
C1B6 C3AEC1    JMP  PMSG

;
;
C1B9 0D0A0A4C4FMSG8 DB  ODH,0AH,0AH,'LOC.      TEST BYTE      MEMORY BYTE',0
;
; PRINT 8 BIT WORD IN BINARY FORMAT
; INPUT: DATA IN REG A
;
;
C1E5 47        BITS:  MOV  B,A      ; DATA
C1E6 3F80      MVI  A,80H      ; MASK
C1E8 0E30      OVER:  MVI  C,30H
C1EA 5F        MOV  E,A        ; STORE MASK
C1EB A0        ANA  B          ; AND WITH MASK
C1EC CAF1C1    JZ  PRNT        ; JUMP IF ZERO
C1EF 0E31      MVI  C,31H
C1F1 CD8EC1    PRNT:  CALL CONOT
C1F4 A0        ANA  B          ; ZERO CARRY
C1F5 7B        MOV  A,E        ; LOAD MASK
C1F6 1F        RAR
C1F7 D2E8C1    JNC OVER
C1FA C9        RET

```

110


```

;
; MINI-MEMORY TEST
; PROM VERSION FOR J TO 100H
;
; BRIAN J. DONLAN
;
C290
C290 F3      ENTER:  ORG      0C290H
C291 3EFE      MVI      A,0FEH
C293 D3FF      OUT      OFFH      ;OUTPUT PHASE I LITES
C295 210000    LXI      H,000H    ;START ADDRESS
C298 AF        LP2:     XRA      A      ;ZERO ACC
C299 77        LP1:     MOV      M,A      ;STORE TEST PATTERN IN MEM.
C29A 46        MOV      B,M      ;READ BACK TO B
C29B 88        CMP      B      ;COMPARE FOR OK
C29C C2FDC2    JNZ      ERR1      ;JUMP IF ERROR
C29F 3C        INR      A      ;NEW TEST PATTERN
C2A0 C299C2    JNZ      LP1
C2A3 23        INX      H
C2A4 1100FF    LXI      D,OFF00H      ;STOP ADDRESS
C2A7 EB        XCHG
C2A8 19        DAD      D      ;ADD TWO'S COMPLIMENT
C2A9 EB        XCHG
C2AA D298C2    JNC      LP2

;
; PHASE II
C2AD 3EFD      MVI      A,OFDH      ;PHASE II LITES
C2AF D3FF      OUT      OFFH
C2B1 210000    LXI      H,000H
C2B4 74        LP3:     MOV      M,H      ;LOW ADDRESS TO MEM
C2B5 23        INX      H
C2B6 1100FF    LXI      D,OFF00H      ;STOP ADDRESS
C2B9 EB        XCHG
C2BA 19        DAD      D
C2BB EB        XCHG
C2BC D2B4C2    JNC      LP3
;
; READ MEMORY
C2BF 210000    LP4:     LXI      H,000H
C2C2 7E        MOV      A,M      ;READ MEMORY
C2C3 94        SUB      H      ;COMPARE
C2C4 C223C3    JNZ      ERR2      ;JUMP IF ERROR
C2C7 23        INX      H
C2C8 1100FF    LXI      D,OFF00H
C2CB EB        XCHG
C2CC 19        DAD      D
C2CD EB        XCHG
C2CE D2C2C2    JNC      LP4

;
; PHASE III
C2D1 3EFC      MVI      A,OFCH      ;PHASE THREE LITES
C2D3 D3FF      LXI      H,000H
C2D5 210000    LP5:     MOV      M,L      ;STORE HIGH ADDRESS IN ALL MEM
C2D8 75        INX      H
C2D9 23        LXI      D,OFF00H
C2DA 1100FF    XCHG
C2DD EB        DAD      D
C2DE 19        XCHG
C2DF EB        DAD      D
C2E0 D2D8C2    JNC      LP5
; READ MEM
C2E3 210000    LP6:     LXI      H,000H
C2E6 7E        MOV      A,M      ;READ MEMORY
C2E7 95        SUB      L      ;COMPARE
C2E8 C22FC3    JNZ      ERR3
C2EB 23        INX      H
C2EC 1100FF    LXI      D,OFF00H
C2EF EB        XCHG
C2F0 19        DAD      D
C2F1 EB        XCHG
C2F2 D2E6C2    JNC      LP6

```

```

;
; ALL PHASE COMPLETE
C2F5 3EFF      MVI    A,OFFH
C2F7 2190C2    LXI    H,ENTER
C2FA C33BC3    JMP     LITES      ;GO TO LITES PROG
;
; PHASE I ERROR
ERR1: XCHG
C2FD EB        MOV     C,A      ;SAVE BAD DATA
C2FE 4F        LXI     H,COMERR  ;RETURN
C2FF 2107C3    MVI     A,0F1H   ;PHASE I ERROR LITES
C302 3EF1      JMP     LITES
;
; COMMON ERROR OUTPUT ROUTINE
COMERR: MOV     A,D      ;HIGH ADDRESS
C308 210EC3    LXI     H,LOADD   ;RETURN
C30B C33BC3    JMP     LITES
C30E 7B        LOADD: MOV    A,E      ;LOW ADDRESS TO LITES
C30F 2115C3    LXI     H,TPAT    ;RETURN
C312 C33BC3    JMP     LITES
C315 79        TPAT:  MOV    A,C      ;TEST PATTERN TO LITES
C316 211CC3    LXI     H,ACTDAT   ;RETURN
C319 C33BC3    JMP     LITES
C31C 78        ACTDAT: MOV   A,B      ;ACTUAL DATA TO LITES
C31D 2190C2    LXI     H,ENTER    ;START OVER
C320 C33BC3    JMP     LITES
;
; PHASE II ERROR
ERR2: XCHG      ;SAVE BAD ADDRESS
C323 EB        ADD     D
C324 82        MOV     B,A
C325 47        MOV     C,D
C326 4A        MVI     A,0F2H     ;PHASE II ERROR TO LITES
C327 3EF2      LXI     H,COMERR   ;RETURN
C329 2107C3    JMP     LITES
C32C C33BC3
;
; PHASE III ERROR
ERR3: XCHG      ;SAVE BAD ADDRESS
C32F EB        ADD     E
C330 83        MOV     B,A
C331 47        MOV     C,E
C332 4B        MVI     A,0F3H     ;PHASE II ERRO TO LITES
C333 3EF3      LXI     H,COMERR   ;RETURN
C335 2107C3    JMP     LITES
C338 C33BC3
;
; LITES ROUTINE      ENTER WITH RETURN IN REG H&L
;                      DATA FOR LITES IN A
LITES: CMA
C33B 2F        OUT     OFFH      ;OUTPUT LITES
C33C D3FF      SPHL
C33E F9        IN      OFFH     ;SAVE RETURN IN SP
C33F DBFF      MOV     H,A      ;READ SENSE SWITCHES
C341 67        IN      OFFH     ;SAVE IN H
C342 DBFF      IN      OFFH     ;READ SWITCHES
C344 AC        XRA     H        ;SEE IF THEY CHANGED
C345 CA42C3    JZ      LP7
C348 2118FC    LXI     H,0FC18H  ;DELAY LOOP
C34B 23        INX     H
C34C AF        XRA     A
C34D B4        ORA     H
C34E C24BC3    JNZ     LP8
C351 210000    LXI     H,0
C354 39        DAD     SP
C355 E9        PCHL
;ZERO H
;MOVE RETURN BACK TO H & L
;RETURN

```

```

;
; DISK TEST FOR TARBELL DISK CONTROLLER
; BRIAN J. DONLAN
; 18 MAR 79
; FROM VERSION
;
C800          ORG      0C800H
C803 217000  ENTRY1: LXI      H,070H          ;SET STACK POINTER
C804 F9      SPHL
C804 F3      DI
C805 CD52CD  ENTRY:  CALL     INITA          ;INITI ITY
C808 2162C9  LXI      H,MSG1          ;OPENING MESSAGE
C808 CD8CCC  CALL     PMSG
C80E 2185C9  LXI      H,MSG1A
C811 CD8CCC  CALL     PMSG
C814 CDDACA  CALL     CONIN
C817 FE59    CPI      'Y'          ;CHECK KEYBOARD
C819 C205C8  JNZ      ENTRY          ;CHECK IF Y
C81C CD63CD  CALL     CRLF          ; ?? START OVER
C81F CD63CD  LOOP6:  CALL     CRLF
C822 AF      XRA      A          ;ZERO ACC
C823 320800  STA      ERRFLG        ;ZERO EERROR FLAG
C826 320400  STA      LPCNT        ;ZERO LOOP COUNT
C829 CD08CB  CALL     HOME          ;HOME DRIVE TO TRK 0
C82C AF      LOOP4:  XRA      A
C82D 320500  STA      INNER        ;ZERO INNER TRK
C830 3E26    MVI      A,38         ;OUTER TRK
C832 320600  STA      OUTER
C835 CD82C8  CALL     PAT          ;GET PATTERN
C838 CDB5C8  CALL     INWRT
C83B 3E22    MVI      A,34
C83D CD72CC  CALL     SEEK
C840 CDE7C8  CALL     INRD
C843 3E01    MVI      A,01
C845 320500  STA      INNER        ;MOVE BACK AND CHECK TRK 00
;
; TEST FOR CONSOLE INTERRUPT
;
C848 DB03    LOOP8:  IN      CSTAT
C84A E602    ANI      02H          ;KEYBAORD READY
C84C CA5BC8  JZ       LOOP3        ;NO
C84F DB02    IN      CDATA
C851 FE03    CPI      03H          ;READ KEYS
C853 CA00F0  JZ       RENT         ;CONTROL C
C856 FE02    CPI      02H          ;RETURN TO MONITOR
C858 CA05C8  JZ       ENTRY        ;CONTROLB
;
; START OVER AGAIN
;
C85B CDB5C8  LOOP3:  CALL     INWRT        ;WRITE INNER TRK
C85E CDEEC8  CALL     OUTWRT
C861 CDE7C8  CALL     INRD          ;READ INNER TRK
C864 CD17C9  CALL     OUTRD
C867 3A0500  LDA      INNER
C86A 3C      INR      A
C86B 320500  STA      INNER
C86E C626    ADI      38
C870 320600  STA      OUTER        ;FIND NEXT OUTER TRK
C873 FE4D    CPI      77          ;STORE OUTER TRK
C875 C248C8  JNZ      LOOP8        ;TRK ?? YET ?
C878 3A0400  LDA      LPCNT        ;NOT DONE YET
C87B 3C      INR      A          ;LOOP COUNTER
C87C 320400  STA      LPCNT
C87F C32CC8  JMP      LOOP4

```

```

;
; PATTERN ROUTINE EXPANDABLE
C882 3A0400 PAT: LDA LPCNT ;LOAD LOOP COUNTER
C885 CAA3C8 JZ IST
C888 FE01 CPI 01 ;SECOND PASS
C88A CAA9C8 JZ SECD
C88D FE02 CPI 02
C88F CAAFC8 JZ THIRD
C892 21ACC9 LXI H,MSG2 ;END OF PASS
C895 CD8CCC CALL PMSG
C898 DB03 IN CSTAT ;CHECK KEYBOARD
C89A E602 ANI 02H
C89C CA1FC8 JZ LOOP6 ;CONTINUE TEST UNTIL INTERRUPTED
C89F 76 HLT
C8A0 C305C8 JMP ENTRY
C8A3 3EFF IST: MVI A,OFFH ;ALL ONES PATTERN
C8A5 320700 STA PATEN ;STORE PATTERN
C8A8 C9 RET
C8A9 3E00 SECD: MVI A,00H ;ALL ZERO PATTERN
C8AB 320700 STA PATEN
C8AE C9 RET
C8AF 3E55 THIRD: MVI A,55H ;ALTER PATTERN
C8B1 320700 STA PATEN
C8B4 C9 RET

;
; WRITE INNER TRK
C8B5 3A0500 INWRT: LDA INNER
C8B8 320800 STA TRK
C8BB CD72CC BOTH: CALL SEEK ;MOVE HEAD TO TRK
C8BE 3E01 MVI A,01 ;FIRST SECTOR
C8C0 320C00 STA SECT

;
; LOOP1:
C8C3 AF XRA A ;ZERO ACC
C8C4 320D00 STA REPETE ;ZERO REPEAT FLAG
C8C7 CD37CC CALL WRITE ;WRITE ONE SECTOR
C8CA 3A0D00 LDA REPETE ;LOAD REPEAT FLAG
C8CD B7 ORA A ;SET FLAGS
C8CE C2C3C8 JNZ LOOP1 ;REPEAT SECTOR
C8D1 3A0C00 LDA SECT
C8D4 3C INR A ;INC SECTOR
C8D5 320C00 STA SECT
C8D8 FE1B CPI 27 ;ALL SECTOR DONE ?
C8DA C2C3C8 JNZ LOOP1 ;NO
C8DD C9 RET

;
; WRITE OUTER TRK
C8DE 3A0600 OUTWRT: LDA OUTER ;LOAD OUTER TRK
C8E1 320800 STA TRK
C8E4 C3BBC8 JMP BOTH ;COMMON WRITE ROUTINE

;
; READ INNER TRK
C8E7 3A0500 INRD: LDA INNER
C8EA 320800 STA TRK
C8ED CD72CC BOTH2: CALL SEEK ;MOVE HEAD TO TRK
C8F0 3E01 MVI A,01 ;FIRST SECTOR
C8F2 320C00 STA SECT ;ZERO SECTOR

;
; LOOP5:
C8F5 AF XRA A ;ZERO ERROR COUNT
C8F6 320800 STA ERRFLG
C8F9 320D00 STA REPETE ;READ ONE SECTOR
C8FC CD3BCB CALL READ ;ZERO ACC
C8FF AF XRA A
C900 320800 STA ERRFLG
C903 3A0D00 LDA REPETE ;REPEAT FLAG
C906 B7 ORA A ;SET FLAGS
C907 C2F5C8 JNZ LOOP5
C90A 3A0C00 LDA SECT
C90D 3C INR A
C90E 320C00 STA SECT ;NEXT SECTOR
C911 FE1B CPI 27 ;ALL SECTORS DONE ?
C913 C2F5C8 JNZ LOOP5 ;NO
C916 C9 RET

```

```

;
; READ OUTER TRK
C917 3A0600 OUTRD: LDA OUTER ;OUTER TRK NO.
C91A 320B00 STA TRK
C91D C3EDC8 JMP BOTH2

;
ERRPNT: LXI H,MSG3 ;ERROR MESSAGE
C920 21BCC9 CALL PMSG
C923 CD8CCC LDA ERRFLG ;ERROR COUNT
C926 3A0800 MOV D,A
C929 57 CALL BINHA ;PRINT ERROR COUNT
C92A CD2BCD LXI H,MSG4 ;HEADINGS
C92D 21F3C9 CALL PMSG
C930 CD8CCC MVI D,03 ;SPACE OVER
C933 1603 CALL BLNK
C935 CD21CD LDA TRK ;TRACK NO.
C938 3A0B00 MOV D,A
C93B 57 CALL BINHA ;PRINT TRACK NO.
C93C CD2BCD MVI D,16 ;SPACE OVER
C93F 1610 CALL BLNK
C941 CD21CD LDA SECT ;SECTOR NO.
C944 3A0C00 MOV D,A
C947 57 CALL BINHA ;PRINT SECTOR NO.
C948 CD2BCD MVI D,13 ;SPACE OVER
C94B 1603 CALL BLNK
C94D CD21CD LDA PATEN
C950 3A0700 CALL BITS ;PRINT TEST PATTERN
C953 CD0BCD MVI D,12 ;SPACE OVE
C956 160C CALL BLNK
C958 CD21CD LDA BADBT ;LAST BAD BYTE
C95B 3A0900 CALL BITS ;PRINT LAST BAD BYTE
C95E CD0BCD RET
C961 C9

;
0004 = LPCNT: EQU 4 ;SPACE FOR LOOP COUNTER
0005 = INNER: EQU 5 ;SPACE FOR INNER TRK NO.
0006 = OUTER: EQU 6 ;SPACE FOR OUTER TRK NO.
0007 = PATEN: EQU 7 ;SPACE FOR TEST PATTERN
0008 = ERRFLG: EQU 8 ;SPACE FOR ERROR COUNT
0009 = BADBT: EQU 9 ;SPACE FOR BAD BYTE
000A = BDTRK: EQU 0AH ;SPACE FOR DISK READ TRK WHEN ERR
000D = REPETE: EQU 0DH ;REPETE FLAG
C962 0D0A444953MSG1: DB 0DH,0AH,'DISK TEST NO. 1 FORMATTED TEST ',0
C985 0D0A4C4F41MSG1A: DB 0DH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY',0
C9AC 0D0A20454EMSG2: DB C3H,0AH,'END OF PASS ',0
C9BC 0D0A444154MSG3: DB 0DH,0AH,'DATA ERROR ON DISK CHECK ERROR COUNT IN HEX ',0
C9F3 0D0A205452MSG4: DB 0DH,0AH,' TRACK NO. SECTOR NO. TEST BYTE LAST ERROR'
CA37 0D0A484541MSG5: DB 0DH,0AH,'HEAD POSITION ',0
CA48 0D0A444953MSG6: DB 0DH,0AH,'DISK TRACK CONTROLLER TRACK SECTOR ',0DH,0AH,
CA83 0D0A0D0A20MSG7: DB 0DH,0AH,0DH,0AH,' !! EXECUTION STOPPED !! ',0
CAA2 0D0A545950MSG8: DB 0DH,0AH,'TYPE R TO RETRY, C TO CONTINUE, ANYTHING ELSE STOP ',0

;
0003 = CSTAT: EQU 3 ;CONSOLE STATUS PORT.
0003 = CCOM: EQU 3 ;CONSOLE COMMAND PORT.
0002 = CDATA: EQU 2 ;CONSOLE DATA PORT.
0002 = CKBR: EQU 00000010B ;KEYBOARD READY BIT.
0001 = CPTR: EQU 00000001B ;PRINT READY BIT.
0001 = CNULL: EQU 1 ;CONSOLE NULL COUNT.
00F8 = DISK: EQU 0F8H ;DISK BASE ADDRESS.
00F8 = DCOM: EQU DISK ;DISK COMMAND PORT.
00F8 = DSTAT: EQU DISK ;DISK STATUS PORT.
00F9 = TRACK: EQU DISK+1 ;DISK TRACK PORT.
00FA = SECTP: EQU DISK+2 ;DISK SECTOR PORT.
00FB = DDATA: EQU DISK+3 ;DISK DATA PORT.
00FC = WAIT: EQU DISK+4 ;DISK WAIT PORT.
00FC = DCONT: EQU DISK+4 ;DISK CONTROL PORT.

000B = TRK: EQU 0BH ;ADDRESS FOR TRACK
000C = SECT: EQU 0CH ;ADDRESS FOR SECTOR

```

```

; READ A CHARACTER FROM CONSOLE.
;
GADA DB03
CADC E602
CONIN: IN CSTAT ;READ CONSOLE STATUS.
      ANI CKBR ;IF NOT READY,

CADE CADACA
CAE1 DB02 JZ CONIN ;READY WHEN HIGH.
CAE3 D302 IN CDATA ;READ A CHARACTER.
CAE5 E67F OUT CDATA
CAE7 C9 ANI 7FH ;MAKE MOST SIG. BIT = 0.
      RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
CAE8 3E0D CONOT: MVI A,ODH ;IF IT'S A CR,
CAEA B9 CMP C ;THEN HOP OUT
CAEB CAF9CA JZ CONUL ;TO NULL ROUTINE.
CAEE DB03 CONOT1: IN CSTAT ;READ CONSOLE STATUS.
CAFO E601 ANI CPTR ;IF NOT READY,
CAF2 CAEECA JZ CONOT1 ;READY WHEN HIGH.
CAF5 79 MOV A,C ;GET CHARACTER.
CAF6 D302 OUT CDATA ;PRINT IT.
CAF8 C9 RET ;RETURN.
CAF9 C5 CONUL: PUSH B ;SAVE B&C.
CAFA 0601 MVI B,CNULL ;GET NULL COUNT.
CAFC CDEECA CONUL1: CALL CONOT1 ;PRINT CR.
CAFF 0E00 MVI C,0 ;GET NULL CHAR.
CB01 05 DCR B ;DECREMENT COUNTER.
CB02 C2FCCA JNZ CONUL1 ;DO NEXT NULL.
CB05 C1 POP B ;RESTORE B&C.
CB06 79 MOV A,C ;RESTORE A.
CB07 C9 RET ;RETURN.

; MOVE DISK TO TRACK ZERO.
;
CB08 3E0D HOME: MVI A,0D0H ;CLEAR ANY PENDING COMMAND.
CB0A D3F8 OUT DCOM
CB0C AF XRA A ;ZERO ACC
CB0D 320B00 STA TRK ;STORE TRACK
CB10 DBF8 HOME1: IN DSTAT ;READ DISK STATUS.
CB12 0F RRC ;LOOK AT LSB.
CB13 DA10CB JC HOME1 ;WAIT FOR NOT BUSY.
CB16 3E03 MVI A,3 ;20 MS STEP RATE.
CB18 D3F8 OUT DCOM ;ISSUE HOME COMMAND.
CB1A DBFC IN WAIT ;WAIT FOR INTRQ.
CB1C B7 ORA A ;SET FLAGS.
CB1D FA2CCB JM HERR ;ERROR IF DRQ.
CB20 DBF8 IN DSTAT ;READ DISK STATUS.
CB22 57 MOV D,A ;SAVE IN REGISTER D.
CB23 E604 ANI 4 ;LOOK AT BIT 2.
CB25 CA2CCB JZ HERR ;ERROR IF NOT TRK 0.
CB28 7A MOV A,D ;GET STATUS BACK.
CB29 E691 ANI 91H ;MASK NON-ERROR BITS.
CB2B C8 RZ ;RETURN IF NO ERROR.
CB2C 21FACC HERR: LXI H,HEMSG ;PRINT "HOME ".
CB2F 7A MOV A,D ;MASK NON-ERROR BITS.
CB30 E691 ANI 91H
CB32 57 MOV D,A
CB33 C374CB JMP ERMMSG ;DO COMMON ERROR MSGS.

; SELECT DISK NUMBER.
;
CB36 3E02 INTDSK: MVI A,02 ;DRIVE NO. 1
CB38 D3FC DSK1: OUT DCONT ;SET THE LATCH WITH CODE.
CB3A C9 RET ;RETURN FROM SELDSK.

```

```

; READ THE SECTOR AT SECT, FROM THE PRESENT TRACK.
; SECTOR IN SECT
; HEAD LOAD FIRST
;
CB3B 218000 READ: LXI H,080H ;READ BUFFER
CB3E 3A0C00 LDA SECT
CB41 D3FA READ1: OUT SECTP ;SET SECTOR INTO 1771.
CB43 3E8C MVI A,8CH ;CODE FOR READ W/O HD LD.
CB45 D3F8 READE: OUT DCOM ;SEND COMMAND TO 1771.
CB47 DBFC RLOOP: IN WAIT ;WAIT FOR DRQ OR INTRQ.
CB49 B7 ORA A ;SET FLAGS.
CB4A F254CB JP RDDONE ;DONE IF INTRQ.
CB4D DBFB IN DDATA ;READ A DATA BYTE FROM DISK.
;
CB4F 77 ; MOV M,A ;STORE IN BUFFER
CB50 23 ; INX H ;INC BUFF POINTER
;
CB51 C347CB ; JMP RLOOP
;
; COMPARE DATA WITH TEST BYTE;
RDDONE: LXI H,080H ;HEAD OF BUFFER
CB54 218000 LDA PATEN ;TEST PATTERN
CB57 3A0700 MOV B,A ;PATTERN TO B
CB5A 47 MVI D,080H ;COUNTER FOR BYTES
CB5B 1680 COMPLP: MOV A,M ;GET DATA
CB5D 7E CMP B ;COMPARE WITH TB
CB5E B8 JNZ DATERR ;ERROR
CB5F C22ACC ERRET: INX H
CB62 23 DCR D ;DEC BYTE COUNT
CB63 15 JNZ COMPLP ;DO 128 TIMES
CB64 C25DCB IN DSTAT ;READ DISK STATUS.
CB67 DBF8 ANI 9DH ;LOOK AT ERROR BITS.
CB69 E69D MOV D,A ;SAVE ERROR BITS
CB6B 57 LDA ERRFLG ;READ ERROR FLAG
CB6C 3A0800 ORA D ;SET FLAGS ON COMBO
CB6F B2 RZ ;RETURN IF NONE.
CB70 C8 LXI H,RDMSG ;PRINT "READ ".
CB71 21E1CC ERMSG: CALL PMSG ;PRINT ORIGIN MESSAGE.
CB74 CD8CCC ;
;
; COMMON ERROR PRINT OUT
;
CB77 7A ERMSG1: MOV A,D ;GET ERROR BITS.
CB78 E680 ANI 80H ;IF BIT 7 HIGH,
CB7A 2197CC LXI H,NRMSG ;"NOT READY".
CB7D C48CCC CNZ PMSG
CB80 7A MOV A,D ;GET ERROR BITS.
CB81 E610 ANI 10H ;IF BIT 4 IS HIGH,
CB83 21A2CC LXI H,RNMSG ;PRINT "RECORD NOT FOUND"
CB86 C48CCC CNZ PMSG
CB89 7A MOV A,D ;GET ERROR BITS.
CB8A E608 ANI 8H ;IF BIT 3 IS HIGH,
CB8C 21B4CC LXI H,CRCMSG ;PRINT "CRC ERROR".
CB8F C48CCC CNZ PMSG
CB92 7A MOV A,D ;GET ERROR BITS.
CB93 E604 ANI 4H ;IF BIT 2 IS HIGH,
CB95 21B9CC LXI H,LDMSG ;PRINT "LOST DATA".
CB98 C48CCC CNZ PMSG
CB9B 7A MOV A,D ;GET ERROR BITS.
CB9C E601 ANI 1 ;IF BIT 1 IS HIGH,
CB9E 21C4CC LXI H,BSYMSG ;PRINT "BUSY".
CBA1 C48CCC CNZ PMSG
CBA4 21DACC PERMSG: LXI H,ERRMSG ;PRINT "ERROR."
CBA7 CD8CCC CALL PMSG
CBAA 7A MOV A,D ;MOVE FLAGS TO ACC
CBAB E618 ANI 18H ;CRC OR RECORD NOT FOUND
CBAD CAEFCB JZ RETRY
CBB0 3EC4 TRCHK: MVI A,0C4H
CBB2 D3F8 OUT DCOM ;READ ADDRESS
CBB4 DBFC IN WAIT

```

```

CBB6 DBFB          IN      DDATA          ;TRACK ADDRESS
CBB8 320A00        STA      BDIRK
CBBB DBFC          CHKS2 IN      WAIT      ;DUMP REST OF DATA
CBBD FABBCB        JM       CHKS2
CBC0 2137CA        LXI      H,MSG5        ;HEAD ERROR MESSAGE
CBC3 CD8CCC        CALL     MSG          ;HEADINGS
CBC6 2148CA        LXI      H,MSG6
CBC9 CD8CCC        CALL     MSG
CBCC 1605          MVI      D,05H
CBCE CD21CD        CALL     BLNK          ;SPACE OVER
CBD1 3A0A00        LDA      BDIRK        ;DISK TRK
CBD4 57            MOV      D,A
CBD5 CD2BCD        CALL     BINHA        ;PRINT TRK
CBD8 1615          MVI      D,15H
CBDA CD21CD        CALL     BLNK          ;SPACE OVER
CBD9 DBF9          IN      TRACK
CBDF 57            MOV      D,A
CBE0 CD2BCD        CALL     BINHA        ;PRINT TRK
CBE3 1613          MVI      D,13H
CBE5 CD21CD        CALL     BLNK
CBE8 3A0C00        LDA      SECT          ;SECTOR
CBEB 57            MOV      D,A
CBEC CD2BCD        CALL     BINHA        ;PRINT SECTO NO.
CBF3 3A0800        RETRY: LDA      ERRFLG
CBF2 B7            ORA      A
CBF3 C420C9        CNZ      ERRPNT        ;SET FLAGS
CBF6 DB02          IN      CDATA          ;GO TO READ CHECK ERROR PRINT
CBF8 DBFF          IN      OFFH          ;CLEAR KEYBOARD
CBFA E601          ANI      01H          ;READ SENSE SWITCHES
CBFC C221CC        JNZ      CONT          ;SWITCH 0
CBFF 21A2CA        LXI      H,MSG8
CC02 CD8CCC        CALL     MSG          ;REQUEST INPUT
CC05 CDDACA        CALL     COMIN        ;READ KEYS
CC08 FE52          CPI      'R'          ;CHECK FOR R
CC0A CA15CC        JZ       FIX           ;CHECK FOR C
CC0D FE43          CPI      'C'
CC0F CA21CC        JZ       CONT
CC12 C300F0        JMP      RENT
CC15 3E01          MVI      A,01          ;SET REPETE FLAG
CC17 320D00        STA      REPETE
CC1A CD63CD        CALL     CRLF
CC1D CD63CD        CALL     CRLF
CC20 C9           RET
CC21 CD63CD        CONT: CALL     CRLF
CC24 CD63CD        CALL     CRLF
CC27 3E01          MVI      A,01
CC29 C9           RET

;
; DATERR: STA      BADBT          ;SAVE BAD BYTE
;          LDA      ERRFLG        ;LOAD ERROR COUNT
;          INR      A
;          STA      ERRFLG        ;NEW COUNT
;          JMP      ERRET         ;RETURN
;
;
; WRITE THE SECTOR AT SECT, ON THE PRESENT TRACK.
; USE STARTING ADDRESS AT DMAADD.
; LOAD HEAD FIRST
;
CC37 3A0700        WRITE: LDA      PATEN
CC3A 47            MOV      B,A          ;TEST PATTERN IN B
CC3B 3A0C00        LDA      SECT        ;LOAD SECTOR
CC3E D3FA          WRITE1: OUT     SECTP   ;SET THE SECTOR INTO 1771.
CC40 3EAC          MVI      A,OACH      ;SET UP 1771 FOR WRITE.
CC42 D3F8          OUT     DCOM
CC44 DBFC          WLOOP: IN      WAIT   ;WAIT FOR READY.
CC46 B7            ORA      A           ;SET FLAGS.
CC47 F251CC        JP       WDONE      ;HOP OUT WHEN DONE.
;
; INSERT PATTERN HERE
;          MOV      A,B          ;LOAD TEST PATTERN
CC4A 78

```

```

CC4B D3FB      OUT  DDATA      ;WRITE ONTO DISK.
CC4D 23        INX  H          ;INCREMENT MEM PTR.
CC4E C344CC    JMP  WLOOP      ;KEEP WRITING.
CC51 DBF8      WDONE: IN  DSTAT  ;READ DISK STATUS.
CC53 E6FD      ANI  OFDH       ;LOOK AT THESE BITS.
CC55 57        MOV  D,A        ;SAVE STATUS BITS.
CC56 C8        PROCER: RZ      ;RETURN IF NO ERR.
CC57 21E9CC    WERRO: LXI  H,WMSG ;PRINT "WRITE ".
CC5A CD8CCC    CALL PMSG
CC5D 7A        MOV  A,D        ;GET ERROR BITS.
CC5E E640      ANI  40H       ;LOOK AT BIT 6.
CC60 21CACC    LXI  H,WMSG     ;PRINT "PROTECT ".
CC63 C48CCC    CNZ  PMSG
CC66 7A        MOV  A,D        ;GET ERROR BITS.
CC67 E620      ANI  20H       ;LOOK AT BIT 5.
CC69 21D3CC    LXI  H,WMSG     ;PRINT "FAULT ".
CC6C C48CCC    CNZ  PMSG
CC6F C377CB    JMP  ERMSG1     ;DO COMMON MESSAGES.

;
; MOVE THE HEAD TO THE TRACK IN REGISTER A.
;
CC72 D3FB      SEEK:  OUT  DDATA  ;TRACK TO DATA REGISTER.
CC74 DBF8      BUSY:  IN   DSTAT  ;READ DISK STATUS.
CC76 0F        RRC          ;LOOK AT BIT 0.
CC77 DA74CC    JC   BUSY      ;WAIT TILL NOT BUSY.
CC7A 3E12      MVI  A,12H     ;SET FOR 10 MS STEP.
CC7C D3FB      OUT  DCOM      ;ISSUE SEEK COMMAND.
CC7E DBFC      IN   WAIT      ;WAIT FOR INTRQ.
CC80 DBF8      IN   DSTAT     ;READ STATUS.
CC82 E691      ANI  91H       ;LOOK AT BITS.
CC84 57        MOV  D,A        ;SAVE STATUS.
CC85 C8        RZ          ;RETURN IF NO ERROR.
CC86 21F2CC    LXI  H,SKMSG    ;PRINT "SEEK ".
CC89 C374CB    JMP  ERMSG      ;DO COMMON ERR MESSAGES.

;
; PRINT THE MESSAGE AT H&L UNTIL A ZERO.
;
CC8C 7E        PMSG:  MOV  A,M    ;GET A CHARACTER.
CC8D B7        ORA  A          ;IF IT'S ZERO,
CC8E C8        RZ          ;RETURN.
CC8F 4F        MOV  C,A        ;OTHERWISE,
CC90 CDEBCA    CALL CONOT      ;PRINT IT.
CC93 23        INX  H          ;INCREMENT H&L,
CC94 C38CCC    JMP  PMSG      ;AND GET ANOTHER.

;
; CBIOS MESSAGES
FOO0 =        RENT  EQU  0F000H ;MONITOR ENTRY
;
;
CC97 4E4F542052NRMSG: DB  'NOT READY ',0
CCA2 5245434F52RNMSG: DB  'RECORD NOT FOUND ',0
CCB4 4352432000CRCMSG: DB  'CRC ',0
CCB9 4C4F535420LDMSG:  DB  'LOST DATA ',0
CCC4 4255535920BSYMSG: DB  'BUSY ',0
CCCA 50524F5445WPMSG:  DB  'PROTECT ',0
CCD3 4641554C54WFMSG:  DB  'FAULT ',0
CCDA 4552524F52ERRMSG: DB  'ERROR.',0
CCE1 0D0A524541RDMSG:  DB  0DH,0AH,'READ ',0
CCE9 0D0A575249WTMSG:  DB  0DH,0AH,'WRITE ',0
CCF2 0D0A534545SKMSG:  DB  0DH,0AH,'SEEK ',0
CCFA 0D0A484F4DHMSG:   DB  0DH,0AH,'HOME ',0
CD02 0D0A4D4F55MNTMSG: DB  0DH,0AH,'MOUNT ',0

```



```

CDC8 06F0      ;
CDCA 78      ERRAL: MVI      B,0F0H
CDCB 2F      ERRLP: MOV      A,B
CDCC D3FF      CMA
CDCE 210000    OUT      OFFH
CDD1 110100    LXI      H,000H      ;DELAY LOOP
CDD4 19      LXI      D,01H
CDD5 D2D4CD    ERRLP: DAD      D
CDD8 47      JNC      ERRLPB
CDD9 DBFF      MOV      B,A
CDDB FE4D      IN      OFFH      ;SEE IF SWITCHES FIXED
CDDD D2CACD    CPI      77
CDE0 210000    JNC      ERRLP
CDE3 110100    DELAY: LXI      H,0H
CDE6 19      LXI      D,01
CDE7 D2E6CD    DELP: DAD      D
CDEA 210000    JNC      DELP
CEDD 110100    DELAY: LXI      H,0H
CDF0 19      LXI      D,01H
CDF1 D2FOCD    DELPA: DAD      D
CDF4 C3ACCD    JNC      DELPA
                JMP      STARTC
                ;
CDF7 0D0A0A4449MSG1B: DB      0DH,0AH,0AH,'DISK TRACK WRITE ROUTINE'
CE12 0D0A4C4F41 DB      0DH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY',0
CE39 0D0A3F3F00MSG2A: DB      0DH,0AH,'??',0
                ;
                ;
                ; DISC TEST FULL TRACK READ
                ; SELECT TRACK IN SENSE SWITCHES
                ; PROM VERSION
                ; BRIAN DONLAN
                ; JUNE 79
                ;
CE40      ;      ORG      0CE40H
                ;
CE40 C34BCE    ENTRYD: JMP      STARTD
CE43 218000    ENTRYE: LXI      H,080H
CE46 F9      SPHL
CE47 F3      DI
CE48 CD52CD    CALL      INITA      ;SET STACK
CE4B 21C0CE    STARTD: LXI      H,MSG1E      ;RESET SIO
CE4E CD8CCC    READU: CALL      PMSG
CE51 3E00      MVI      A,00H
CE53 320800    STA      ERRFLG
CE56 CDDACA    CALL      CONIN      ;READ KEYBOARD
CE59 FE59      CPI      'Y'
CE5B 2139CE    LXI      H,MSG2A
CE5E C24ECE    JNZ      READU
CE61 CD08CB    CALL      HOME
CE64 B7      STARTE: ORA      A      ;SET FLAGS
CE65 C24BCE    JNZ      STARTD      ;ERROR START OVER
CE68 2164CE    LXI      H,STARTE      ;SUBROUTINE RETURN
CE6B E5      PUSH      H
CE6C DBFF      STARTF: IN      OFFH      ;READ SENSE SWITCHES
CE6E FE4D      CPI      77      ;PREVENT TRACK OVER-DRIVE
CE70 D285CE    JNC      ERRD
CE73 CD72CC    SEEKD: CALL      SEEK      ;MOVE HEAD TO TRACK
CE76 3EE5      MVI      A,0E5H      ;READ COMMAND
CE78 D3F8      OUT      DCOM
CE7A DBFC      RDLP : IN      WAIT
CE7C B7      ORA      A
CE7D F2B4CE    JP      RDONE
CE80 DBFB      IN      DDATA
CE82 C37ACE    JMP      RDLP

```

```

CE85 06F0      ;
CE87 78      ERRD: MVI B,0F0H
CE88 2F      ERRLPD: MOV A,B
CE89 D3FF      CMA
CE8B 110100      OUT OFFH
CE8E 210000      LXI D,01H
CE91 19      LXI H,000H      ;DELAY LOOP
CE92 D291CE      ERRLPE: DAD D
CE95 47      JNC ERRLPE
CE96 DBFF      MOV B,A
CE98 FE4D      IN OFFH      ;SEE IF SWITCHES FIXED
CE9A D287CE      CPI 77
CE9D 210000      JNC ERRLPD
CEA0 110100      DELAYD: LXI H,0
CEA3 19      LXI D,01H
CEA4 D2A3CE      DELPD: DAD D
CEA7 210000      JNC DELPD
CEAA 110100      DELAYE: LXI H,0
CEAD 19      LXI D,01H
CEAE D2ADCE      DELPE: DAD D
CEB1 C36CCE      JNC DELPE
CEB4 DBF8      JMP STARTF
CEB6 E69D      ;
CEB8 57      RDONE: IN DSTAT
CEB9 C8      ANI 9DH
CEBA 21E1CC      MOV D,A
CEBD C374CB      RZ
CEC0 0D0A0A4449MSG1E: DB ODH,0AH,0AH,'DISK TRACK READ ROUTINE'
CEDA 0DCA4C4F41      DB ODH,0AH,'LOAD SCRATCH DISK TYPE Y WHEN READY ',0

```

```

D000          ORG      00000H
              ;        UNIBUS PORT TEST
              ;        AND UBIBUS COMM. TEST COMBINED
              ;        AND SNAP-SHOT
              ;        PROM VERSION
              ;
D000 218000   ENTRY1: LXI      H,080H
D003 F9       SPHL          ;SET STACK POINTER
D004 2104D0   ENTRY:  LXI      H,ENTRY
D007 E5       PUSH        H
D008 F3       DI
D009 CDDAD2   CALL        INITA      ;RESET IO
D00C 2184D0   LXI         H,MSG1     ;OPENING MESSAGE
D00F CD50D2   CALL        PMSG
              ;
              ;BEGINING OF TEST
D012 3E01     MVI         A,01H
D014 0E10     MVI         C,10H      ;PORT UNDER TEST
D016 D310     PORT10: OUT    10H
D018 47       MOV        B,A        ;SAVE TEST PATTERN
D019 DB10     IN         10H        ;READ BUSS
D01B B8       CMP        B          ;COMPARE
D01C C45FD0   CNZ         ERR        ;CALL IF IN ERROR
D01F 07       RLC
D020 D216D0   JNC        PORT10     ;TEST FOR A COMPLETE CICLE
              ;
D023 3E01     MVI         A,01H
D025 0E11     MVI         C,11H      ;PORT 11
D027 D311     PORT11: OUT    11H
D029 47       MOV        B,A        ;SAVE PATTERN
D02A DB11     IN         11H        ;READ BUSS
D02C B8       CMP        B          ;C OMPARE
D02D C45FD0   CNZ         ERR        ;CALL IF ERROR
D030 07       RLC
D031 D227D0   JNC        PORT11
              ;
              ;
D034 3E01     MVI         A,01H
D036 0E12     MVI         C,12H      ;PORT12
D038 D312     PORT12: OUT    12H
D03A 47       MOV        B,A        ;SAVE TEST PATERN
D03B DB12     IN         12H
D03D B8       CMP        B
D03E C45FD0   CNZ         ERR
D041 07       RLC
D042 D238D0   JNC        PORT12
              ;
              ;
D045 3E01     MVI         A,01H
D047 0E13     MVI         C,13H
D049 D313     PORT13: OUT    13H
D04B 47       MOV        B,A
D04C DB13     IN         13H
D04E B8       CMP        B
D04F C45FD0   CNZ         ERR
D052 07       RLC
D053 D249D0   JNC        PORT13
              ;
              ;
D056 21D5D0   LXI         H,MSG4     ;FINISHED MESSAGE
D059 CD50D2   CALL        PMSG
D05C C300F0   JMP         RENT      ;RETURN TO MONITOR

```

```

D05F C5      ; ERR:  PUSH  B      ;SAVE ERROR PATTERN
D060 F5      PUSH  A      ;SAVE TEST PATTERN
D061 51      MOV   D,C
D062 2198D0  LXI   H,MSG2
D065 CD50D2  CALL  PMSG
D068 CD7BD2  CALL  BINHA
D06B 21AED0  LXI   H,MSG0
D06E CD50D2  CALL  PMSG
D071 78      MOV   A,B
D072 CD5BD2  CALL  BITS
D075 21C1D0  LXI   H,MSG3
D078 CD50D2  CALL  PMSG
D07B F1      POP   A
D07C CD5BD2  CALL  BITS
D07F C1      POP   B
D080 78      MOV   A,B
D081 37      STC
D082 3F      CMC
D083 C9      RET

;
D084 0A0A0D554MSG1 DB 0AH,0AH,ODH,'UNIBUS PORT TEST',0
D098 0A0A0D4552MSG2: DB 0AH,0AH,ODH,'ERROR PORT NO.',0
D0AE 0A0D544553MSG0: DB 0AH,ODH,'TEST PATTERN',0
D0C1 0A0D414354MSG3 DB 0AH,ODH,'ACTUAL PATTERN',0
D0D5 0A0A0D454MSG4 DB 0AH,0AH,ODH,'END OF TEST',0
FOOO = RENT EQU 0F000H ;MONITOR ENTRY

;
;UNIBUS COMMUNICATION TEST
;
D100      ORG 0D100H
D100 218000 ENTRY2: LXI H,080H ;SET STACK
D103 F9      SPHL
D104 CDDAD2  CALL INITA
D107 2107D1 ENTRY3: LXI H,ENTRY3
D10A E5      PUSH H
D10B F3      DI
D10C 2160D1  LXI H,MSG5 ;OPENNING MEESSAGE
D10F CD50D2  CALL PMSG
D112 CDA2D2  CALL BBIN ;GET HEX CHAR
D115 D5      PUSH D ;SAVE ADDRESS
D116 2196D1  LXI H,MSG6 ;REQUEST MODE
D119 CD50D2  CALL PMSG
D11C CD22D2  TRYGN: CALL CONIN
D11F FE49  CPI 'I'
D121 CA3ED1  JZ PUTIN ;JUMP IF INPUT MAODE
D124 FE03  CPI 03H ;TEST IF CONTROL C
D126 CA00F0  JZ RENT ;RETURN TO MONITOR
D129 FE4F  CPI 'O'
D12B C257D1  JNZ QUEST

;
;OUTPUT MODE
;
D12E 21C7D1  PUTOUT: LXI H,MSG11 ;OUTPUT MESSAGE
D131 CD50D2  CALL PMSG
D134 CDA2D2  CALL BBIN ;GET DIGITS TO OUTPUT
D137 C1      POP B ;RESTORE ADDRESS TO REG B & C
D138 CD08D3  CALL DATA0 ;UNIBUSS DRIVER
D13B C34ED1  JMP DONE

;
D13E 21F1D1  PUTIN: LXI H,MSG9 ;INPUT MESSAGE
D141 CD50D2  CALL PMSG
D144 C1      POP B ;RESTORE ADDRESS TO B & C
D145 CDFED2  CALL DATAI ;UNIBUS INPUT ROUTINE
D148 CD90D2  CALL BINB ;PRINT DATA FROM BUSS
D14B C34ED1  JMP DONE

;
D14E 2105D2  DONE: LXI H,MSG10 ;PRINT END OF TEST
D151 CD50D2  CALL PMSG
D154 C307D1  JMP ENTRY3

```

```

;
D157 21C2D1    QUEST: LXI    H,MSG7
D15A CD50D2    CALL    PMSG      ; ??
D15D C31CD1    JMP     TRYGN

;
D160 0A0A0D554EMSG5: DB    0AH,0AH,0DH,'UNIBUS COMMUNICATION TEST'
D17C 0A0D454E54    DB    0AH,0DH,'ENTER UNIBUS ADDRESS ',0
D196 0A0D494E50MSG6: DB    0AH,0DH,'INPUT (I), OUTPUT (O), EXIT (CONTROL C) ?',0
D1C2 0A0D203F00MSG7: DB    0AH,0DH,' ??',0
D1C7 0A0D454E54MSG11: DB    0AH,0DH,'ENTER DATA TO OUTPUT IN 4 HEX DIGITS ',0
D1F1 0A0D204441MSG9: DB    0AH,0DH,' DATA FROM BUS ',0
D205 0A0D545241MSG10 DB    0AH,0DH,'TRANSFER COMPLETE',0

;
; diagnostic input output routines
; for brian donlan 26 feb 79

0003 = CSTAT EQU 3 ;CONSOLE STATUS PORT.
0003 = CCOM EQU 3 ;CONSOLE COMMAND PORT.
0002 = CDATA EQU 2 ;CONSOLE DATA PORT.
0002 = CKBR EQU 00000010B ;KEYBOARD READY BIT.
0001 = CPTR EQU 00000001B ;PRINT READY BIT.
0001 = CNULL EQU 1 ;CONSOLE NULL COUNT.

; CHECK CONSOLE INPUT STATUS.
;
D219 DB03 CONST: IN CSTAT ;READ CONSOLE STATUS.
D21B E602 ANI CKBR ;LOOK AT KB READY BIT.
D21D 3E00 MVI A,0 ;SET A=0 FOR RETURN.
D21F C8 RZ ;NOT READY WHEN ZERO.
D220 2F CMA ;IF READY A=FF.
D221 C9 RET ;RETURN FROM CONST.

;
; READ A CHARACTER FROM CONSOLE.
;
D222 DB03 CONIN: IN CSTAT ;READ CONSOLE STATUS.
D224 E602 ANI CKBR ;IF NOT READY,

D226 CA22D2 JZ CONIN ;READY WHEN HIGH.
D229 DB02 IN CDATA ;READ A CHARACTER.
D22B D302 OUT CDATA
D22D E67F ANI 7FH ;MAKE MOST SIG. BIT = 0.
D22F C9 RET

;
; WRITE A CHARACTER TO THE CONSOLE DEVICE.
;
D230 3E0D CONOT: MVI A,0DH ;IF IT'S A CR,
D232 B9 CMP C ;THEN HOP OUT
D233 CA41D2 JZ CONUL ;TO NULL ROUTINE.
D236 DB03 CONOT1: IN CSTAT ;READ CONSOLE STATUS.
D238 E601 ANI CPTR ;IF NOT READY,
D23A CA36D2 JZ CONOT1 ;READY WHEN HIGH.
D23D 79 MOV A,C ;GET CHARACTER.
D23E D302 OUT CDATA ;PRINT IT.
D240 C9 RET ;RETURN.
D241 C5 CONUL: PUSH B ;SAVE B&C.
D242 0601 MVI B,CNULL ;GET NULL COUNT.
D244 CD36D2 CONUL1: CALL CONOT1 ;PRINT CR.
D247 0E00 MVI C,0 ;GET NULL CHAR.
D249 05 DCR B ;DECREMENT COUNTER.
D24A C244D2 JNZ CONUL1 ;DO NEXT NULL.
D24D C1 POP B ;RESTORE B&C.
D24E 79 MOV A,C ;RESTORE A.
D24F C9 RET ;RETURN.

```



```

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;      INITIATE SIO PORTS
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;
D2DA 3EAA      INITA: MVI    A,0AAH      ;GET DUMMY MODE WORD
D2DC D303      OUT     CSTAT           ;OUTPUT IT
D2DE 3E40      MVI     A,40H          ;GET RESET BIT
D2E0 D303      OUT     CSTAT           ;RESET SIO BOARD
D2E2 3ECE      MVI     A,0CEH          ;GET REAL MODE WORD
D2E4 D303      OUT     CSTAT           ;SET THE MODE FOR REAL
D2E6 3E37      MVI     A,37H          ;GET THE COMMAND
D2E8 D303      OUT     CSTAT           ;OUTPUT IT
D2EA C9

;
;
D2EB 0E0D      CRLF:  MVI     C,13      ;CR
D2ED CD30D2    CALL    CONOT
D2F0 0E0A      LF:   MVI     C,10      ;LF
D2F2 CD36D2    CALL    CONOT1
D2F5 0E7F      MVI     C,7FH
D2F7 CD36D2    CALL    CONOT1
D2FA CD30D2    CALL    CONOT
D2FD C9

;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
;
;
D2FE CD91D3    DATAI: CALL    GETBUS
D301 CD12D3    CALL    DATI
D304 97        SUB     A
D305 D315      OUT     15H
D307 C9

;
;
D308 CD91D3    DATAO: CALL    GETBUS
D30B CD51D3    CALL    DATO
D30E 97        SUB     A
D30F D315      OUT     15H
D311 C9

;
;
;ROUTINE TO INPUT A 16 BIT WORD FROM UNIBUS
;REG B = A<16:09>, REG C = A<08:01>
;DATA WILL BE CONTAINED IN REG D = D<15:08>, REG C = D<07:00>
;
;
D312 3EFF      DATI:  MVI     A,OFFH      ;SET LOOP COUNT
D314 328200    STA     BIZCNT
D317 DB14      BIZLP1: IN      14H        ;CHECK FOR SYS = 0
D319 E604      ANI     04H        ;FROM LAST TRANSACTION
D31B C2A2D3    JNZ     BBUSY1

;
D31E 78        MOV     A,B          ;OUTPUT HIGH ADDRESS
D31F D310      OUT     10H
D321 79        MOV     A,C          ;OUTPUT LOW ADDRESS
D322 D311      OUT     11H
D324 97        SUB     A            ;OUTPUT C1=0
D325 D314      OUT     14H
D327 F601      ORI     01H          ;OUTPUT MSYN=1
D329 D314      OUT     14H

;
D32B 3EFF      SYNLP1: MVI     A,OFFH      ;LOOP COUNT
D32D 328100    STA     SYNCNT
D330 DB14      DILOOP: IN      14H        ;CHECKS IF SSYN=1
D332 D3FF      OUT     OFFH
D334 E604      ANI     04H
D336 CAC8D3    JZ      NOSYN1

```

131

```

; ON-LINE UNIBUS DIAGNOSTICS
; BY BRIAN DONLAN 24 APR 79
;
D3A2 3A8200 BBUSY1: LDA BIZCNT ;LOOP COUNT
D3A5 3D DCR A
D3A6 328200 STA BIZCNT ;NEW COUNT
D3A9 C217D3 JNZ BIZLP1 ;JUMP IF STILL COUNT
D3AC 213ED4 LXI H,ERMSG2
D3AF CD50D2 CALL PMSG ;DISPLAY ERROR MESSAGE
D3B2 C307D1 JMP ENTRY3
;
D3B5 3A8200 BBUSY2: LDA BIZCNT
D3B8 3D DCR A
D3B9 328200 STA BIZCNT
D3BC C256D3 JNZ BIZLP2
D3BF 213ED4 LXI H,ERMSG2
D3C2 CD50D2 CALL PMSG
D3C5 C307D1 JMP ENTRY3
;
D3C8 3A8100 NOSYN1: LDA SYNCNT
D3CB 3D DCR A
D3CC 328100 STA SYNCNT
D3CF C230D3 JNZ DILOOP
D3D2 DBFF IN OFFH
D3D4 E640 ANI 040H
D3D6 C230D3 JNZ DILOOP
D3D9 50 SYSERR: MOV D,B ;
D3DA 59 MOV E,C ;MOV ADDRESS FOR OUTPUT
D3DB 215BD4 LXI H,ERMSG3
D3DE CD50D2 CALL PMSG
D3E1 CD90D2 CALL BINB ;OUTPUT ADDRESS
D3E4 216BD4 LXI H,ERMSG4
D3E7 CD50D2 CALL PMSG
D3EA AF XRA A ;ZERO A
D3EB D311 OUT 11H
D3ED D312 OUT 12H
D3EF D313 OUT 13H
D3F1 D314 OUT 14H
D3F3 D310 OUT 10H
D3F5 C307D1 JMP ENTRY3
;
D3F8 3A8100 NOSYN2: LDA SYNCNT
D3FB 3D DCR A
D3FC 328100 STA SYNCNT
D3FF C276D3 JNZ DOLOOP
D402 DBFF IN OFFH
D404 E640 ANI 040H
D406 C276D3 JNZ DOLOOP
D409 C3D9D3 JMP SYSERR
;
D40C 3A8000 NOGET: LDA GETCNT
D40F 3D DCR A
D410 328000 STA GETCNT
D413 C29AD3 JNZ LOOP
D416 211FD4 LXI H,ERMSG1
D419 CD50D2 CALL PMSG
D41C C307D1 JMP ENTRY3
;
;
;
0080 = GETCNT: EQU 080H
0081 = SYNCNT: EQU 081H
0082 = BIZCNT: EQU 082H
D41F 0D0D0A2020ERMSG1: DB 0DH,0DH,0AH,' IMSAI CAN NOT GET BUSS ',0
D43E 0D0D0A2020ERMSG2: DB 0DH,0DH,0AH,' ERROR BUSS BUSY ',0
D45B 0D0A202044ERMSG3: DB 0DH,0AH,' DEVICE NO. ',0
D46B 204E4F2052ERMSG4: DB ' NO RESPONSE ',0

```

```

; UNIBUS SNAP SHOT ROUTINE
D500      ; ORG      00500H
D500 2100F0  ENTRY: LXI      H,0F000H
D503 E5      PUSH      H
D504 F3      DI
D505 CDDAD2   CALL      INITA      ;RESET I/O

; SUBROUTINE ENTRY POINT
D508 216BD5  ENTRY5: LXI      H,MSG12
D508 CDS0D2   CALL      PMSG
D50E DB10      IN          10H      ;HIGH ADDRESS
D510 57      MOV          D,A      ;SAVE IN D
D511 DB11      IN          11H      ;LOW ADDRESS
D513 5F      MOV          E,A
D514 CD90D2   CALL      BINB      ;PRINT UNIBUS ADDRESS

;
D517 1608      MVI          D,08H      ;SPACE OVER
D519 CD71D2   CALL      BLNK
D51C DB12      IN          12H      ;HIGH DATA
D51E 57      MOV          D,A
D51F DB13      IN          13H      ;LOW DATA BITS
D521 5F      MOV          E,A
D522 CD90D2   CALL      BINB      ;PRINT UNIBUS DATA BITS
D525 1608      MVI          D,08H      ;SPACE OVER
D527 CD71D2   CALL      BLNK

;
D52A DB14      IN          14H      ;STATUS PORT
D52C E604      ANI          04H      ;FIND SLAVE SYN
D52E CA36D5      JZ          NOSIS
D531 0E31      MVI          C,'1'
D533 C338D5      JMP          OUTSIS
D536 0E30      NOSIS: MVI          C,'0'
D538 CD30D2      OUTSIS: CALL      CONOT      ;PRINT SLAVE SYN
D53B 1609      MVI          D,09H      ;SPACE OVER
D53D CD71D2   CALL      BLNK

;
D540 DB15      IN          15H      ;STATUS PORT
D542 E601      ANI          01H      ;BUS GRANT
D544 CA4CD5      JZ          NOBUS
D547 0E31      MVI          C,'1'
D549 C34ED5      JMP          OUTBUS
D54C 0E30      NOBUS: MVI          C,'0'
D54E CD30D2      OUTBUS: CALL      CONOT      ;PRINT BUS GRANT
D551 160B      MVI          D,0BH      ;SPACE OVER
D553 CD71D2   CALL      BLNK

;
D556 DB14      IN          14H
D558 E600      ANI          00H
D55A CA62D5      JZ          NOMSYN
D55D 0E31      MVI          C,'1'
D55F C364D5      JMP          OUTMSN
D562 0E30      NOMSYN: MVI          C,'0'
D564 CD30D2      OUTMSN: CALL      CONOT      ;PRINT MSYN
D567 C9      RET
D568 C368D5      FINIS: JMP          FINIS

D56B 0A0A0D554EMSG12: DB      0AH,0AH,0DH,'UNIBUS SNAP-SHOT '
D580 0A0D414444      DB      0AH,0DH,'ADDRESS      DATA      SSYN      GRANT      MSYN'
D5B4 0A0D202000      DB      0AH,0DH,' ',0

```

```

; 8K MINI MEMORY TEST
;
; BRIAN DONLAN
; PROM VERSION
D600          ORG      0D600H
D600 F3      ENTER:   DI
D601 3EFE    MVI      A, OFEH
D603 D3FF    OUT      OFFH
D605 210000  LXI      H, 000H
D608 AF      LP2:     XRA      A
D609 77      LP1:     MOV      M, A
D60A 46      MOV      B, M
D60B B8      CMP      B
D60C C26DD6  JNZ      ERR1
D60F 3C      INR      A
D610 C209D6  JNZ      LP1
D613 23      INX      H
D614 1100E0  LXI      D, 0E000H
D617 EB      XCHG
D618 19      DAD      D
D619 EB      XCHG
D61A D208D6  JNC      LP2
;
; PHASE II
D61D 3EFD    MVI      A, OFDH
D61F D3FF    OUT      OFFH
D621 210000  LXI      H, 000H
D624 74      LP3:     MOV      M, H
D625 23      INX      H
D626 1100E0  LXI      D, 0E000H
D629 EB      XCHG
D62A 19      DAD      D
D62B EB      XCHG
D62C D224D6  JNC      LP3
;
; READ MEMORY
D62F 210000  LP4:     LXI      H, 000H
D632 7E      MOV      A, M
D633 94      SUB      H
D634 C293D6  JNZ      ERR2
D637 23      INX      H
D638 1100E0  LXI      D, 0E000H
D63B EB      XCHG
D63C 19      DAD      D
D63D EB      XCHG
D63E D232D6  JNC      LP4
;
; PHASE III
;
D641 3EFC    MVI      A, OFCH
D643 D3FF    OUT      OFFH
D645 210000  LXI      H, 000H
D648 75      LP5:     MOV      M, L
D649 23      INX      H
D64A 1100E0  LXI      D, 0E000H
D64D EB      XCHG
D64E 19      DAD      D
D64F EB      XCHG
D650 D248D6  JNC      LP5
; READ MEM
D653 210000  LP6:     LXI      H, 000H
D656 7E      MOV      A, M
D657 95      SUB      L
D658 C29FD6  JNZ      ERR3
D65B 23      INX      H
D65C 1100E0  LXI      D, 0E000H
D65F EB      XCHG
D660 19      DAD      D
D661 EB      XCHG
D662 D256D6  JNC      LP6
; OUTPUT PHASE I LITES
; START ADDRESS
; ZERO ACC
; STORE TEST PATTERN IN MEM.
; READ BACK TO B
; COMPARE FOR OK
; JUMP IF ERROR
; NEW TEST PATTERN
; STOP ADDRESS
; ADD TWO'S COMPLIMENT
; PHASE II LITES
; LOW ADDRESS TO MEM
; STOP ADDRESS
; READ MEMORY
; COMPARE
; JUMP IF ERROR
; PHASE THREE LITES
; STORE HIGH ADDRESS IN ALL MEM
; READ MEMORY
; COMPARE

```

```

D665 3EFF      ; ALL PHASE COMPLETE
D667 2100D6    MVI    A,OFFH
D66A C3ABD6    LXI    H,ENTER
                JMP    LITES                                ;GO TO LITES PROG
;
; PHASE I ERROR
D66D EB        ERR1: XCHG
D66E 4F        MOV    C,A
D66F 2177D6    LXI    H,COMERR                                ;SAVE BAD DATA
D672 3EF1      MVI    A,OF1H                                ;RETURN
D674 C3ABD6    JMP    LITES                                ;PHASE I ERROR LITES
;
; COMMON ERROR OUTPUT ROUTINE
D677 7A        COMERR: MOV    A,D                                ;HIGH ADDRESS
D678 217ED6    LXI    H,LOADD                                ;RETURN
D67B C3ABD6    JMP    LITES
D67E 7B        LOADD: MOV    A,E                                ;LOW ADDRES TO LITES
D67F 2185D6    LXI    H,TPAT                                ;RETURN
D682 C3ABD6    JMP    LITES
D685 79        IPAT:  MOV    A,C                                ;TEST PATTERN TO LITES
D686 218CD6    LXI    H,ACTDAT                                ;RETURN
D689 C3ABD6    JMP    LITES
D68C 78        ACTDAT: MOV    A,B                                ;ACTUAL DATA TO LITES
D68D 2100D6    LXI    H,ENTER                                ;START OVER
D690 C3ABD6    JMP    LITES
;
; PHASE II ERROR
D693 EB        ERR2: XCHG                                ;SAVE BAD ADDRESS
D694 82        ADD    D
D695 47        MOV    B,A
D696 4A        MOV    C,D
D697 3EF2      MVI    A,OF2H                                ;PHASE II ERROR TO LITES
D699 2177D6    LXI    H,COMERR                                ;RETURN
D69C C3ABD6    JMP    LITES
;
; PHASE III ERROR
D69F EB        ERR3: XCHG                                ;SAVE BAD ADDRESS
D6A0 83        ADD    E
D6A1 47        MOV    B,A
D6A2 4B        MOV    C,E
D6A3 3EF3      MVI    A,OF3H                                ;PHASE II ERRO TO LITES
D6A5 2177D6    LXI    H,COMERR                                ;RETURN
D6A8 C3ABD6    JMP    LITES
;
; LITES ROUTINE
D6AB 2F        LITES: CMA                                ENTER WITH RETURN IN REG H&L
D6AC D3FF      OUT    OFFH                                DATA FOR LITES IN A
D6AE F9        SPHL
D6AF DBFF      IN     OFFH                                ;OUTPUT LITES
D6B1 67        IN     OFFH                                ;SAVE RETURN IN SP
D6B2 DBFF      MOV    H,A                                ;READ SENSE SWITCHES
D6B4 AC        LP7:  IN     OFFH                                ;SAVE IN H
D6B5 CAB2D6    XRA    H                                ;READ SWITCHES
D6B8 2118FC    JZ     LP7                                ;SEE IF THEY CHANGED
D6BB 23        LP8:  LXI    H,0FC18H                                ;DELAY LOOP
D6BC AF        INX    H
D6BD B4        XRA    A
D6BE C2BBD6    ORA    H
D6C1 210000    JNZ    LP8
D6C4 39        LXI    H,0
D6C5 E9        DAD    SP                                ;ZERO H
                PCHL                                ;MOVE RETURN BACK TO H & L
                ;RETURN

```

```

; 24K MINI-MEMORY TEST
; PROM VERSION
;
; BRIAN DONLAN
D700 ORG 0D700H
D700 F3 ENTER2: DI
D701 3EFE MVI A,0FEH
D703 D3FF OUT OFFH ;OUTPUT PHASE I LITES
D705 210000 LXI H,000H ;START ADDRESS
D708 AF LP22: XRA A ;ZERO ACC
D709 77 LP12: MOV M,A ;STORE TEST PATTERN IN MEM.
D70A 46 MOV B,M ;READ BACK TO B
D70B B8 CMP B ;COMPARE FOR OK
D70C C26DD7 JNZ ERR12 ;JUMP IF ERROR
D70F 3C INR A ;NEW TEST PATTERN
D710 C209D7 JNZ LP12
D713 23 INX H
D714 1100A0 LXI D,0A000H ;STOP ADDRESS
D717 EB XCHG
D718 19 DAD D ;ADD TWO'S COMPLIMENT
D719 EB XCHG
D71A D208D7 JNC LP22

; PHASE II
D71D 3EFD MVI A,0FDH ;PHASE II LITES
D71F D3FF OUT OFFH
D721 210000 LXI H,000H
D724 74 LP32: MOV M,H ;LOW ADDRESS TO MEM
D725 23 INX H
D726 1100A0 LXI D,0A000H ;STOP ADDRESS
D729 EB XCHG
D72A 19 DAD D
D72B EB XCHG
D72C D224D7 JNC LP32
; READ MEMORY
D72F 210000 LXI H,000H
D732 7E LP42: MOV A,M ;READ MEMORY
D733 94 SUB H ;COMPARE
D734 C293D7 JNZ ERR22 ;JUMP IF ERROR
D737 23 INX H
D738 1100A0 LXI D,0A000H
D73B EB XCHG
D73C 19 DAD D
D73D EB XCHG
D73E D232D7 JNC LP42

; PHASE III
D741 3EFC MVI A,0FCH ;PHASE THREE LITES
D743 D3FF OUT OFFH
D745 210000 LXI H,000H
D748 75 LP52: MOV M,L ;STORE HIGH ADDRESS IN ALL MEM
D749 23 INX H
D74A 1100A0 LXI D,0A000H
D74D EB XCHG
D74E 19 DAD D
D74F EB XCHG
D750 D248D7 JNC LP52
; READ MEM
D753 210000 LXI H,000H
D756 7E LP62: MOV A,M ;READ MEMORY
D757 95 SUB L ;COMPARE
D758 C29FD7 JNZ ERR32
D75B 23 INX H
D75C 1100A0 LXI D,0A000H
D75F EB XCHG
D760 19 DAD D
D761 EB XCHG
D762 D256D7 JNC LP62

```

```

; ALL PHASE COMPLETE
D765 3EFF      MVI    A,OFFH
D767 2100D7    LXI    H,ENTER2
D76A C3ABD7    JMP     LITES2      ;GO TO LITES PROG

; PHASE I ERROR
D76D EB        ERR12: XCHG
D76E 4F        MOV     C,A
D76F 2177D7    LXI    H,COMER2    ;SAVE BAD DATA
D772 3EF1      MVI    A,OF1H      ;RETURN
D774 C3ABD7    JMP     LITES2      ;PHASE I ERROR LITES

; COMMON ERROR OUTPUT ROUTINE
D777 7A        COMER2: MOV    A,D
D778 217ED7    LXI    H,LOADD2    ;HIGH ADDRESS
D77B C3ABD7    JMP     LITES2      ;RETURN
D77E 7B        LOADD2: MOV    A,E
D77F 2185D7    LXI    H,TPAT2     ;LOW ADDRESS TO LITES
D782 C3ABD7    JMP     LITES2      ;RETURN
D785 79        TPAT2:  MOV    A,C
D786 218CD7    LXI    H,ACTDA2    ;TEST PATTERN TO LITES
D789 C3ABD7    JMP     LITES2
D78C 78        ACTDA2: MOV    A,B
D78D 2100D7    LXI    H,ENTER2    ;ACTUAL DATA TO LITES
D790 C3ABD7    JMP     LITES2      ;START OVER

; PHASE II ERROR
D793 EB        ERR22: XCHG
D794 82        ADD     D           ;SAVE BAD ADDRESS
D795 47        MOV     B,A
D796 4A        MOV     C,D
D797 3EF2      MVI    A,OF2H
D799 2177D7    LXI    H,COMER2    ;PHASE II ERROR TO LITES
D79C C3ABD7    JMP     LITES2      ;RETURN

; PHASE III ERROR
D79F EB        ERR32: XCHG
D7A0 83        ADD     E           ;SAVE BAD ADDRESS
D7A1 47        MOV     B,A
D7A2 4B        MOV     C,E
D7A3 3EF3      MVI    A,OF3H
D7A5 2177D7    LXI    H,COMER2    ;PHASE II ERRO TO LITES
D7A8 C3ABD7    JMP     LITES2      ;RETURN

; LITES ROUTINE ENTER WITH RETURN IN REG H&L
; DATA FOR LITES IN A
D7AB 2F        LITES2: CMA
D7AC D3FF      OUT     OFFH        ;OUTPUT LITES
D7AE F9        SPHL
D7AF DBFF      IN      OFFH        ;SAVE RETURN IN SP
D7B1 67        MOV     H,A         ;READ SENSE SWITCHES
D7B2 DBFF      IN      OFFH        ;SAVE IN H
D7B4 AC        XRA     H           ;READ SWITCHES
D7B5 CAB2D7    JZ      LP72        ;SEE IF THEY CHANGED
D7B8 2118FC    LXI    H,OF18H
D7BB 23        INX     H           ;DELAY LOOP
D7BC AF        XRA     A
D7BD B4        ORA     H
D7BE C2BBD7    JNZ     LP82
D7C1 210000    LXI    H,0
D7C4 39        DAD     SP
D7C5 E9        PCHL
;ZERO H
;MOVE RETURN BACK TO H & L
;RETURN

```

```

;      8080 MONITOR V1.0
;
;      PROGRAMMER: C. E. OHME
;                  (415)657-8326
;
;
;      SYSTEM CONFIGURATION INTERFACE

F600      SCP      EQU      0F600H
F600      IOTAB    EQU      SCP
F630      ADSCS    EQU      SCP+48
F633      ADSCR    EQU      SCP+51
F636      ADIOB    EQU      SCP+54
F639      ADUST    EQU      SCP+57

;
;      ASCII CHARACTERS

000D      CR      EQU      0DH
000A      LF      EQU      0AH

F000      ORG      0F000H

;
;      EXTERNALLY REFERENCED SUBROUTINE
;      JUMP TABLE

F000 C324F0      JMP      BEGIN
F003 C3D6F0      JMP      CI
F006 C3E3F0      JMP      RI
F009 C3D1F0      JMP      CO
F00C C3E5F0      JMP      PO
F00F C3EAF0      JMP      LO
F012 C3D8F0      JMP      CSTS
F015 C30EF1      JMP      IOCHK
F018 C315F1      JMP      IOSET
F01B C3B1F1      JMP      MEMCK
F01E C31EF1      JMP      STRNG
F021 0E00      REENT: MVI      C,0
F023 210003      LXI      H,0
F024              ORG      $-2

F024 0E01      BEGIN: MVI      C,1
F026 11FF00      LXI      D,OFFH
F029 C3EAF6      JMP      INITA
F02C EB          XCHG
F02D 0615      MVI      B,ENDX-EXIT-1
F02F 1176F2      LXI      D,ENDX-1
F032 1B          BGI:  DCX      D
F033 1A          LDAX     D
F034 2B          DCX      H
F035 77          MOV      M,A
F036 05          DCR      B

```

F037 C232F0	JNZ	BG1
F03A F9	SPHL	
F03B CD39F6	CALL	ADUST
F03E E5	PUSH	H
F03F 2600	MVI	H,0
F041 E5	PUSH	H
F042 E5	PUSH	H
F043 E5	PUSH	H
F044 79	MOV	A,C
F045 B7	ORA	A
F046 CA4EF0	JZ	BG2
F049 CD36F6	CALL	ADIOB
F04C 3600	MVI	M,0
F04E 2140F7	BG2: LXI	H, TITLE
F051 CD1EF1	CALL	STRNG
F05A B7	ORA	A
;		
	COMMAND RETURN POINT	
F055 D266F0	CMNDR: JNC	START
;		
	ERROR RETURN	
F058 CD33F6	LER: CALL	ADSCR
F05B 11E3FF	LXI	D,EXIT-ENDX-7
F05E 19	DAD	D
F05F F9	SPHL	
F060 219DF0	LXI	H,ERM
F063 CD1EF1	CALL	STRNG
;		
	INPUT AND EXECUTE NEXT COMMAND	
F066 FB	START: EI	
F067 CD46F1	CALL	CRLF
F06A 0E2E	MVI	C,'.'
F06C CDD1F0	CALL	CO
F06F CD2FF1	CALL	TI
F072 D641	SUI	'A'
F074 FA58F0	JM	LER
F077 FE18	CPI	'X'-'A'+1
F079 F258F0	JP	LER
F07C 87	ADD	A
F07D 2155F0	LXI	H,CMNDR
F080 E5	PUSH	H
F081 219FF0	LXI	H,TBL
F084 1600	MVI	D,0
F086 5F	MOV	E,A
F087 19	DAD	D
F088 7E	MOV	A,M
F089 23	INX	H
F08A 66	MOV	H,M
F08B 6F	MOV	L,A
F08C 0E02	MVI	C,2
F08E E9	PCHL	

F08F 0D0A4D4F VERS: DB CR,LF,'MONITOR VI.','0' OR 80H
 F093 4E49544F
 F097 52205631
 F09B 2EB0
 F09D 0AAA ERM: DB LF,'*' OR 80H

; COMMAND JUMP TABLE

F09F 77F2	TBL:	DW	ASSIGN
F0A1 C0F2		DW	BIN
F0A3 A0F3		DW	HEXN
F0A5 1CF3		DW	DISP
F0A7 58F0		DW	LER
F0A9 3CF3		DW	FILL
F0AB 4CF3		DW	GOTO
F0AD 00F8		DW	HELP
F0AF 58F0		DW	LER
F0B1 58F0		DW	LER
F0B3 BCF3		DW	KOPY
F0B5 C6F3		DW	LOAD
F0B7 0CF4		DW	MOVE
F0B9 1EF4		DW	NULL
F0BB 58F0		DW	LER
F0BD 58F0		DW	LER
F0BF 58F0		DW	LER
F0C1 23F4		DW	READ
F0C3 89F4		DW	SUBS
F0C5 00F0		DW	TEST
F0C7 58F0		DW	LER
F0C9 58F0		DW	LER
F0CB B1F4		DW	WRITE
F0CD 3DF5		DW	X

; UTILITY SUBROUTINES

F0CF 0E20	BLK:	MVI	C, ' '
F0D1 CDEFF0	CO:	CALL	IOBR
F0D4 0110		DB	1,10H
F0D6 CDEFF0	CI:	CALL	IOBR
F0D9 0108		DB	1,8
F0DB CDEFF0	CSTS:	CALL	IOBR
F0DE 0100		DB	1,0
F0E0 CDEFF0	RI:	CALL	IOBR
F0E3 0418		DB	4,18H
F0E5 CDEFF0	PO:	CALL	IOBR
F0E8 0320		DB	3,20H

F0EA CDEFF0	LO:	CALL	IOBR
F0ED 0228		DB	2,28H
F0EF E3	IOBR:	XTHL	
F0F0 C5		PUSH	B
F0F1 46		MOV	B,M
F0F2 23		INX	H
F0F3 4E		MOV	C,M
F0F4 CD36F6		CALL	ADIOB
F0F7 7E		MOV	A,M
F0F8 0F		RRC	
F0F9 07	IOB1:	RLC	
F0FA 07		RLC	
F0FB 05		DCR	B
F0FC C2F9F0		JNZ	IOB1
F0FF E606		ANI	6
F101 81		ADD	C
F102 4F		MOV	C,A
F103 2100F6		LXI	H,IOTAB
F106 09		DAD	B
F107 7E		MOV	A,M
F108 23		INX	H
F109 66		MOV	H,M
F10A 6F		MOV	L,A
F10B C1		POP	B
F10C E3		XTHL	
F10D C9		RET	
	IOCHK:		
F10E E5		PUSH	H
F10F CD36F6		CALL	ADIOB
F112 7E		MOV	A,M
F113 E1		POP	H
F114 C9		RET	
	IOSET:		
F115 E5		PUSH	H
F116 F5		PUSH	PSW
F117 CD36F6		CALL	ADIOB
F11A 71		MOV	M,C
F11B F1		POP	PSW
F11C E1		POP	H
F11D C9		RET	
	STRNG:		
F11E 7E		MOV	A,M
F11F E67F		ANI	7FH
F121 C8		RZ	
F122 4F		MOV	C,A
F123 7E		MOV	A,M
F124 B7		ORA	A
F125 FAD1F0		JM	CO
F128 CDD1F0		CALL	CO
F12B 23		INX	H
F12C C31EF1		JMP	STRNG

F12F CDD6F0	TI:	CALL	CI
F132 E67F		ANI	7FH
F134 C5		PUSH	B
F135 4F		MOV	C,A
F136 CDD1F0		CALL	CO
F139 79		MOV	A,C
F13A C1		POP	B
F13B C9		RET	
F13C E60F	CONV:	ANI	0FH
F13E C690		ADI	90H
F140 27		DAA	
F141 CE40		ACI	40H
F143 27		DAA	
F144 4F		MOV	C,A
F145 C9		RET	
F146 0E0D	CRLF:	MVI	C,CR
F148 CDD1F0		CALL	CO
F14B 0E0A		MVI	C,LF
F14D C3D1F0		JMP	CO
F150 0E01	EXPR1:	MVI	C,1
F152 210000	EXPR:	LXI	H,0000H
F155 CD2FF1	EX0:	CALL	TI
F158 47	EX1:	MOV	B,A
F159 CDD0F1		CALL	NIBBL
F15C DA68F1		JC	EX2
F15F 29		DAD	H
F160 29		DAD	H
F161 29		DAD	H
F162 29		DAD	H
F163 B5		ORA	L
F164 6F		MOV	L,A
F165 C355F1		JMP	EX0
F168 E3	EX2:	XTHL	
F169 E5		PUSH	H
F16A 78		MOV	A,B
F16B CDE5F1		CALL	P2C
F16E D276F1		JNC	EX3
F171 0D		DCR	C
F172 C258F0		JNZ	LER
F175 C9		RET	
F176 C258F0	EX3:	JNZ	LER
F179 0D		DCR	C
F17A C252F1		JNZ	EXPR
F17D C9		RET	
F17E 0E01	EXF:	MVI	C,1
F180 210000		LXI	H,0000H
F183 C358F1		JMP	EX1

F186 23	HILO:	INX	H
F187 7C		MOV	A,H
F188 B5		ORA	L
F189 37		STC	
F18A C8		RZ	
F18B 7B		MOV	A,E
F18C 95		SUB	L
F18D 7A		MOV	A,D
F18E 9C		SBB	H
F18F C9		RET	
F190 7C	LADR:	MOV	A,H
F191 CD95F1		CALL	LBYTE
F194 7D		MOV	A,L
F195 F5	LBYTE:	PUSH	PSW
F196 0F		RRC	
F197 0F		RRC	
F198 0F		RRC	
F199 0F		RRC	
F19A CD9EF1		CALL	HXD
F19D F1		POP	PSW
F19E CD3CF1	HXD:	CALL	CONV
F1A1 C3D1F0		JMP	C0
F1A4 0604	LEADS:	MVI	B,4
F1A6 0E00	LEAD:	MVI	C,0
F1A8 CDE5F0		CALL	P0
F1AB 05		DCR	B
F1AC C2A6F1		JNZ	LEAD
F1AF B7		ORA	A
F1B0 C9		RET	
F1B1 ES	MEMCK:	PUSH	H
F1B2 DS		PUSH	D
F1B3 CD33F6		CALL	ADSCR
F1B6 EB		XCHG	
F1B7 210000		LXI	H,0
F1BA 24	MEM0:	INR	H
F1BB 7E		MOV	A,M
F1BC 2F		CMA	
F1BD 77		MOV	M,A
F1BE BE		CMP	M
F1BF 2F		CMA	
F1C0 77		MOV	M,A
F1C1 CABAF1		JZ	MEM0
F1C4 2B		DCX	H
F1C5 44		MOV	B,H
F1C6 7C		MOV	A,H
F1C7 BA		CMP	D

F1C8 3EC0	MVI	A,0C0H
F1CA D1	POP	D
F1CB E1	POP	H
F1CC C8	RZ	
F1CD 3EFF	MVI	A,0FFH
F1CF C9	RET	

NIBBL:

F1D0 D630	SUI	'0'
F1D2 D8	RC	
F1D3 C6E9	ADI	'0'-'G'
F1D5 D8	RC	
F1D6 C606	ADI	6
F1D8 F2DEF1	JP	N10
F1DB C607	ADI	7
F1DD D8	RC	
F1DE C60A	ADI	10
F1E0 37	ORA	A
F1E1 C9	RET	

PCHK:

F1E2 CD2FF1	CALL	T1
F1E5 FE20	CPI	' '
F1E7 C8	RZ	
F1E8 FE2C	CPI	' '
F1EA C8	RZ	
F1EB FE0D	CPI	CR
F1ED 37	STC	
F1EE C8	RZ	
F1EF 3F	CMC	
F1F0 C9	RET	

;
BREAKPOINT ENTRY POINT

F1F1 E5	RESTR:	PUSH	H
F1F2 D5		PUSH	D
F1F3 C5		PUSH	B
F1F4 F5		PUSH	PSW
F1F5 CD33F6		CALL	ADSCR
F1F8 11EBFF		LXI	D,EXIT-ENDX+1
F1FB 19		DAD	D
F1FC EB		XCHG	
F1FD 210A00		LXI	H,000AH
F200 39		DAD	SP
F201 0604		MVI	B,4
F203 EB		XCHG	
F204 2B	RST0:	DCX	H
F205 72		MOV	M,D
F206 2B		DCX	H
F207 73		MOV	M,E
F208 D1		POP	D
F209 05		DCR	B
F20A C204F2		JNZ	RST0
F20D C1		POP	B

F20E 0B		DCX	B
F20F F9		SPHL	
F210 211400		LXI	H, TLOC
F213 39		DAD	SP
F214 7E		MOV	A, M
F215 91		SUB	C
F216 23		INX	H
F217 C21FF2		JNZ	RST1
F21A 7E		MOV	A, M
F21B 90		SUB	B
F21C CA2DF2		JZ	RST3
F21F 23	RST1:	INX	H
F220 23		INX	H
F221 7E		MOV	A, M
F222 91		SUB	C
F223 C22CF2		JNZ	RST2
F226 23		INX	H
F227 7E		MOV	A, M
F228 90		SUB	B
F229 CA2DF2		JZ	RST3
F22C 03	RST2:	INX	B
F22D 210F00	RST3:	LXI	H, LLOC
F230 39		DAD	SP
F231 73		MOV	M, E
F232 23		INX	H
F233 72		MOV	M, D
F234 23		INX	H
F235 23		INX	H
F236 71		MOV	M, C
F237 23		INX	H
F238 70		MOV	M, B
F239 C5		PUSH	B
F23A 219DF0		LXI	H, ERM
F23D CD1EF1		CALL	STRNG
F240 E1		POP	H
F241 CD90F1		CALL	LADR
F244 211400		LXI	H, TLOC
F247 39		DAD	SP
F248 1602		MVI	D, 2
F24A 4E	RST4:	MOV	C, M
F24B 3600		MVI	M, 0
F24D 23		INX	H
F24E 46		MOV	B, M
F24F 3600		MVI	M, 0
F251 23		INX	H
F252 79		MOV	A, C
F253 B0		ORA	B
F254 CA59F2		JZ	RST5
F257 7E		MOV	A, M
F258 02		STAX	B
F259 23	RST5:	INX	H
F25A 15		DCR	D
F25B C24AF2		JNZ	RST4
F25E C366F0		JMP	START

```

/          SCRATCHPAD TEMPLATE

F261 D1      EXIT:  POP      D
F262 C1      POP      B
F263 F1      POP      PSW
F264 E1      POP      H
F265 F9      SPHL
F266 FB      EI
F267 210000  LXI        H,0
F268          HLX      EQU     S-2
F26A C30000  JMP        0
F26B          PCX      EQU     S-2
F26D 0000    TIA:     DW      0      JTRAP 1 ADDR
F26F 00      DB      0      JTRAP 1 INST
F270 0000    DW      0      JTRAP 2 ADDR
F272 00      DB      0      JTRAP 2 INST
F273 0000    DW      0      JVIDEO PNTR
F275 00      DB      0      JVIDEO HOLD
F276 00      DB      0      JDPNT

          ENDX:
0005      ALOC      EQU     5
0003      BLOC      EQU     3
0002      CLOC      EQU     2
0001      DLOC      EQU     1
0000      ELOC      EQU     0
0004      FLOC      EQU     4
0010      HLOC      EQU     HLX-EXIT+9
000F      LLOC      EQU     HLX-EXIT+8
0013      PLOC      EQU     PCX-EXIT+9
0007      SLOC      EQU     7
0014      TLOC      EQU     TIA-EXIT+8

/          COMMAND IMPLEMENTATION

/          ASSIGN COMMAND

F277 CD2FF1  ASSIGN: CALL     TI
F27A 0600    MVI      B,0
F27C FE43    CPI      'C'
F27E CA93F2  JZ        ASI
F281 04      INR      B
F282 FE52    CPI      'R'
F284 CA93F2  JZ        ASI
F287 04      INR      B
F288 FE50    CPI      'P'
F28A CA93F2  JZ        ASI
F28D 04      INR      B
F28E FE4C    CPI      'L'
F290 C2BEF2  JNZ      EREXT
F293 CD2FF1  ASI:     CALL     TI
F296 FE3D    CPI      '='
F298 C293F2  JNZ      ASI

```

F29B	CD2FF1		CALL	TI
F29E	D630		SUI	'0'
F2A0	6F		MOV	L,A
F2A1	FABEF2		JM	EREXT
F2A4	FE04		CPI	4
F2A6	F2BEF2		JP	EREXT
F2A9	2603		MVI	H,3
F2AB	05	AS2:	DCR	B
F2AC	FAB4F2		JM	AS3
F2AF	29		DAD	H
F2B0	29		DAD	H
F2B1	C3ABF2		JMP	AS2
F2B4	EB	AS3:	XCHG	
F2B5	CD36F6		CALL	ADI0B
F2B8	7E		MOV	A,M
F2B9	B2		ORA	D
F2BA	AA		XRA	D
F2BB	B3		ORA	E
F2BC	77		MOV	M,A
F2BD	C9		RET	
F2BE	37	EREXT:	STC	
F2BF	C9		RET	

;
BINARY COMMAND

F2C0	CD52F1	BIN:	CALL	EXPR
F2C3	CD46F1		CALL	CRLF
F2C6	D1		POP	D
F2C7	E1		POP	H
F2C8	7A	BIN0:	MOV	A,D
F2C9	B3		ORA	E
F2CA	C2D7F2		JNZ	B0
F2CD	CDA4F1		CALL	LEADS
F2D0	0E78		MVI	C,78H
F2D2	CD11F3		CALL	PHL
F2D5	B7		ORA	A
F2D6	C9		RET	
F2D7	7B	B0:	MOV	A,E
F2D8	95		SUB	L
F2D9	7A		MOV	A,D
F2DA	9C		SBB	H
F2DB	D8		RC	
F2DC	7B	B1:	MOV	A,E
F2DD	95		SUB	L
F2DE	4F		MOV	C,A
F2DF	7A		MOV	A,D
F2E0	9C		SBB	H
F2E1	3F		CMC	
F2E2	D0		RNC	
F2E3	0C		INR	C
F2E4	C2E9F2		JNZ	B2
F2E7	0EFF		MVI	C,0FFH
F2E9	D5	B2:	PUSH	D

F2EA 59		MOV	E,C
F2EB CDA4F1		CALL	LEADS
F2EE 0E3C		MVI	C,3CH
F2F0 CDE5F0		CALL	PO
F2F3 4B		MOV	C,E
F2F4 CD11F3		CALL	PHL
F2F7 7C		MOV	A,H
F2F8 85		ADD	L
F2F9 57		MOV	D,A
F2FA 4E	B3:	MOV	C,M
F2FB 23		INX	H
F2FC 7A		MOV	A,D
F2FD 81		ADD	C
F2FE 57		MOV	D,A
F2FF CDE5F0		CALL	PO
F302 1D		DCR	E
F303 C2FAF2		JNZ	B3
F306 4A		MOV	C,D
F307 CDE5F0		CALL	PO
F30A D1		POP	D
F30B 7D		MOV	A,L
F30C B4		ORA	H
F30D C8		RZ	
F30E C3DCF2		JMP	B1
F311 CDE5F0	PHL:	CALL	PO
F314 4D		MOV	C,L
F315 CDE5F0		CALL	PO
F318 4C		MOV	C,H
F319 C3E5F0		JMP	PO

DISPLAY COMMAND

F31C CD52F1	DISP:	CALL	EXPR
F31F D1		POP	D
F320 E1		POP	H
F321 CD46F1	DI0:	CALL	CRLF
F324 CD90F1		CALL	LADR
F327 CDCFF0	DI1:	CALL	BLK
F32A 7E		MOV	A,M
F32B CD95F1		CALL	LEYTE
F32E CD86F1		CALL	HILO
F331 3F		CMC	
F332 D0		RNC	
F333 7D		MOV	A,L
F334 E60F		ANI	0FH
F336 C227F3		JNZ	DI1
F339 C321F3		JMP	DI0

FILL COMMAND

F33C 0C	FILL:	INR	C
F33D CD52F1		CALL	EXPR
F340 C1		POP	B

F341 D1		POP	D
F342 E1		POP	H
F343 71	F18:	MOV	M,C
F344 CD86F1		CALL	HILO
F347 D243F3		JNC	F18
F34A B7		ORA	A
F34B C9		RET	
;			
		GOTO	COMMAND
F34C E1	GOTO:	POP	H
F34D CDE2F1		CALL	PCHK
F350 DA98F3		JC	G03
F353 CA72F3		JZ	G00
F356 CD7EF1		CALL	EXF
F359 D1		POP	D
F35A 211300		LXI	H,PLOC
F35D 39		DAD	SP
F35E 72		MOV	M,D
F35F 2B		DCX	H
F360 73		MOV	M,E
F361 78		MOV	A,B
F362 FE0D		CPI	CR
F364 CA98F3		JZ	G03
F367 3EC3		MVI	A,(JMP 0)
F369 320800		STA	8
F36C 21F1F1		LXI	H,RESTR1
F36F 220900		SHLD	9
F372 1602	G00:	MVI	D,2
F374 211400		LXI	H,TLOC
F377 39		DAD	SP
F378 E5	G01:	PUSH	H
F379 CD50F1		CALL	EXPRI
F37C 58		MOV	E,B
F37D C1		POP	B
F37E E1		POP	H
F37F 78		MOV	A,B
F380 B1		ORA	C
F381 CA8EF3		JZ	G02
F384 71		MOV	M,C
F385 23		INX	H
F386 70		MOV	M,B
F387 23		INX	H
F388 0A		LDAX	B
F389 77		MOV	M,A
F38A 23		INX	H
F38B 3ECF		MVI	A,(RST 1)
F38D 02		STAX	B
F38E 7B	G02:	MOV	A,E
F38F FE0D		CPI	CR
F391 CA98F3		JZ	G03
F394 15		DCR	D
F395 C278F3		JNZ	G01

F398 CD46F1	G03:	CALL	CRLF
F39B 210800		LXI	H,0008H
F39E 39		DAD	SP
F39F E9		PCHL	

; HEXADECIMAL COMMAND

F3A0 CD52F1	HEXN:	CALL	EXPR
F3A3 D1		POP	D
F3A4 E1		POP	H
F3A5 CD46F1		CALL	CRLF
F3A8 E5		PUSH	H
F3A9 19		DAD	D
F3AA CD90F1		CALL	LADR
F3AD CDCFF0		CALL	BLK
F3B0 E1		POP	H
F3B1 7D		MOV	A,L
F3B2 93		SUB	E
F3B3 6F		MOV	L,A
F3B4 7C		MOV	A,H
F3B5 9A		SBB	D
F3B6 67		MOV	H,A
F3B7 CD90F1		CALL	LADR
F3BA B7		ORA	A
F3BB C9		RET	

; COPY COMMAND

F3BC CDE0F0	KOPY:	CALL	RI
F3BF 4F		MOV	C,A
F3C0 CDE5F0		CALL	PO
F3C3 C3BCF3		JMP	KOPY

; LOAD COMMAND

F3C6 CD50F1	LOAD:	CALL	EXPRI
F3C9 C1		POP	B
F3CA CDE0F0	L1:	CALL	RI
F3CD D8		RC	
F3CE FE3C		CPI	3CH
F3D0 CADFF3		JZ	L2
F3D3 FE78		CPI	78H
F3D5 C2CAF3		JNZ	L1
F3D8 CD01F4		CALL	LHL
F3DB D8		RC	
F3DC B5		ORA	L
F3DD C8		RZ	
F3DE E9		PCHL	
F3DF CDE0F0	L2:	CALL	RI
F3E2 D8		RC	
F3E3 5F		MOV	E,A
F3E4 CD01F4		CALL	LHL
F3E7 D8		RC	

F3E8 85		ADD	L
F3E9 57		MOV	D,A
F3EA 09		DAD	B
F3EB CDE0F0	L3:	CALL	RI
F3EE D8		RC	
F3EF 77		MOV	M,A
F3F0 82		ADD	D
F3F1 57		MOV	D,A
F3F2 23		INX	H
F3F3 1D		DCR	E
F3F4 C2EBF3		JNZ	L3
F3F7 CDE0F0		CALL	RI
F3FA D8		RC	
F3FB BA		CMP	D
F3FC CACAF3		JZ	L1
F3FF 37		STC	
F400 C9		RET	
F401 CDE0F0	LHL:	CALL	RI
F404 D8		RC	
F405 6F		MOV	L,A
F406 CDE0F0		CALL	RI
F409 D8		RC	
F40A 67		MOV	H,A
F40B C9		RET	
;			
MOVE COMMAND			
F40C 0C	MOVE:	INR	C
F40D CD52F1		CALL	EXPR
F410 C1		POP	B
F411 D1		POP	D
F412 E1		POP	H
F413 7E	MV0:	MOV	A,M
F414 02		STAX	B
F415 03		INX	B
F416 CD86F1		CALL	HILO
F419 D213F4		JNC	MV0
F41C B7		ORA	A
F41D C9		RET	
;			
NULL COMMAND			
F41E 063C	NULL:	MVI	B,60
F420 C3A6F1		JMP	LEAD
;			
READ COMMAND			
F423 CD50F1	READ:	CALL	EXPRI
F426 E1		POP	H
F427 CDE0F0	RED0:	CALL	RI
F42A D8		RC	
F42B E67F		ANI	7FH
F42D D63A		SUI	','

F42F C227F4		JNZ	RED0
F432 57		MOV	D,A
F433 E5		PUSH	H
F434 CD65F4		CALL	BYTE
F437 CA59F4		JZ	RED2
F43A 5F		MOV	E,A
F43B CD65F4		CALL	BYTE
F43E 47		MOV	B,A
F43F CD65F4		CALL	BYTE
F442 4F		MOV	C,A
F443 09		DAD	B
F444 CD65F4		CALL	BYTE
F447 CD65F4	RED1:	CALL	BYTE
F44A 77		MOV	M,A
F44B 23		INX	H
F44C 1D		DCR	E
F44D C247F4		JNZ	RED1
F450 CD65F4		CALL	BYTE
F453 E1		POP	H
F454 CA27F4		JZ	RED0
F457 37		STC	
F458 C9		RET	
F459 CD65F4	RED2:	CALL	BYTE
F45C 67		MOV	H,A
F45D CD65F4		CALL	BYTE
F460 C1		POP	B
F461 6F		MOV	L,A
F462 B4		ORA	H
F463 C8		RZ	
F464 E9		PCHL	
F465 CD76F4	BYTE:	CALL	RNBBL
F468 07		RLC	
F469 07		RLC	
F46A 07		RLC	
F46B 07		RLC	
F46C 4F		MOV	C,A
F46D CD76F4		CALL	RNBBL
F470 B1		ORA	C
F471 4F		MOV	C,A
F472 82		ADD	D
F473 57		MOV	D,A
F474 79		MOV	A,C
F475 C9		RET	
F476 CDE0F0	RNBBL:	CALL	RI
F479 DA85F4		JC	RNBER
F47C E67F		ANI	7FH
F47E CDD0F1		CALL	NIBBL
F481 DA85F4		JC	RNBER
F484 C9		RET	
F485 E1	RNBER:	POP	H
F486 E1		POP	H
F487 E1		POP	H
F488 C9		RET	

```

;      SUBSTITUTE COMMAND

F489 CD50F1  SUBS:  CALL    EXPR1
F48C CDE5F1      CALL    P2C
F48F E1        POP     H
F490 D8        RC
F491 7E        SUB:   MOV     A,M
F492 CD95F1      CALL    LBYTE
F495 0E2D      MVI     C,'-'
F497 CDD1F0      CALL    CO
F49A CDE2F1      CALL    PCHK
F49D 3F        CMC
F49E D0        RNC
F49F CAADF4      JZ      SU1
F4A2 E5        PUSH    H
F4A3 CD7EF1      CALL    EXF
F4A6 D1        POP     D
F4A7 E1        POP     H
F4A8 73        MOV     M,E
F4A9 78        MOV     A,B
F4AA FE0D      CPI     CR
F4AC C8        RZ
F4AD 23        SUB:   INX     H
F4AE C391F4      JMP     SUB0

```

```

;      WRITE COMMAND

F4B1 CD52F1  WRITE:  CALL    EXPR
F4B4 CD46F1      CALL    CRLF
F4B7 D1        POP     D
F4B8 E1        POP     H
F4B9 7A        WRIT0:  MOV     A,D
F4BA B3        ORA     E
F4BB C2DBF4      JNZ     W0
F4BE CD33F5      CALL    PEOL
F4C1 0E3A      MVI     C,':'
F4C3 CDE5F0      CALL    PO
F4C6 AF        XRA     A
F4C7 57        MOV     D,A
F4C8 CD1CF5      CALL    PBYTE
F4CB CD17F5      CALL    PADR
F4CE 3E01      MVI     A,I
F4D0 CD1CF5      CALL    PBYTE
F4D3 AF        XRA     A
F4D4 92        SUB     D
F4D5 CD1CF5      CALL    PBYTE
F4D8 C31EF4      JMP     NULL
F4DB 7B        W0:   MOV     A,E
F4DC 95        SUB     L
F4DD 7A        MOV     A,D
F4DE 9C        SBB     H
F4DF D8        RC

```

F4E0 7B	WR10:	MOV	A,E
F4E1 95		SUB	L
F4E2 4F		MOV	C,A
F4E3 7A		MOV	A,D
F4E4 9C		SBB	H
F4E5 3F		CMC	
F4E6 D0		RNC	
F4E7 79		MOV	A,C
F4E8 E60F		ANI	0FH
F4EA 3C		INR	A
F4EB D5		PUSH	D
F4EC 5F		MOV	E,A
F4ED 1600		MVI	D,0
F4EF CD33F5		CALL	PEOL
F4F2 0E3A		MVI	C,'1'
F4F4 CDE5F0		CALL	PO
F4F7 7B		MOV	A,E
F4F8 CD1CF5		CALL	PBYTE
F4FB CD17F5		CALL	PADR
F4FE AF		XRA	A
F4FF CD1CF5		CALL	PBYTE
F502 7E	WR13:	MOV	A,M
F503 23		INX	H
F504 CD1CF5		CALL	PBYTE
F507 1D		DCR	E
F508 C202F5		JNZ	WR13
F50B AF		XRA	A
F50C 92		SUB	D
F50D CD1CF5		CALL	PBYTE
F510 D1		POP	D
F511 7D		MOV	A,L
F512 B4		ORA	H
F513 C8		RZ	
F514 C3E0F4		JMP	WR10
F517 7C	PADR:	MOV	A,H
F518 CD1CF5		CALL	PBYTE
F51B 7D		MOV	A,L
F51C F5	PBYTE:	PUSH	PSW
F51D 0F		RRC	
F51E 0F		RRC	
F51F 0F		RRC	
F520 0F		RRC	
F521 CD3CF1		CALL	CONV
F524 CDE5F0		CALL	PO
F527 F1		POP	PSW
F528 F5		PUSH	PSW
F529 CD3CF1		CALL	CONV
F52C CDE5F0		CALL	PO
F52F F1		POP	PSW
F530 82		ADD	D
F531 57		MOV	D,A
F532 C9		RET	
F533 0E0D	PEOL:	MVI	C,CR

F535 CDE5F0
F538 0E0A
F53A C3E5F0

CALL PO
MVI C,LF
JMP PC

REGISTER COMMAND

F53D CD2FF1	X:	CALL	TI
F540 21CCF5		LXI	H,ACTBL
F543 FE0D		CPI	CR
F545 CA9FF5		JZ	X6
F548 47		MOV	B,A
F549 BE	X0:	CMP	M
F54A CA57F5		JZ	X1
F54D 7E		MOV	A,M
F54E 17		RAL	
F54F D8		RC	
F550 23		INX	H
F551 23		INX	H
F552 23		INX	H
F553 78		MOV	A,B
F554 C349F5		JMP	X0
F557 CDCFF0	X1:	CALL	BLK
F55A 23	X2:	INX	H
F55B 7E		MOV	A,M
F55C EB		XCHG	
F55D 6F		MOV	L,A
F55E 2600		MVI	H,0
F560 39		DAD	SP
F561 EB		XCHG	
F562 23		INX	H
F563 46		MOV	B,M
F564 23		INX	H
F565 1A		LDAX	D
F566 CD95F1		CALL	LBYTE
F569 05		DCR	B
F56A CA72F5		JZ	X3
F56D 1B		DCX	D
F56E 1A		LDAX	D
F56F CD95F1		CALL	LBYTE
F572 04	X3:	INR	B
F573 0E2D		MVI	C,'-'
F575 CDD1F0		CALL	C0
F578 CDE2F1		CALI.	PCHK
F57B 3F		CMC	
F57C D0		RNC	
F57D CA95F5		JZ	X5
F580 E5		PUSH	H
F581 C5		PUSH	B
F582 CD7EF1		CALL	EXF
F585 E1		POP	H
F586 F1		POP	PSW
F587 C5		PUSH	B
F588 F5		PUSH	PSW

F589 7D		MOV	A,L	
F58A 12		STAX	D	
F58B C1		POP	B	
F58C 05		DCR	B	
F58D CA93F5		JZ	X4	
F590 13		INX	D	
F591 7C		MOV	A,H	
F592 12		STAX	D	
F593 C1	X4:	POP	B	
F594 E1		POP	H	
F595 7E	X5:	MOV	A,M	
F596 B7		ORA	A	
F597 F8		RM		
F598 78		MOV	A,B	
F599 FE0D		CPI	CR	
F59B C8		RZ		
F59C C35AF5		JMP	X2	
F59F CD46F1	X6:	CALL	CRLF	
F5A2 CDCFF0	X7:	CALL	BLK	
F5A5 7E		MOV	A,M	
F5A6 23		INX	H	
F5A7 B7		ORA	A	
F5A8 F8		RM		
F5A9 4F		MOV	C,A	
F5AA CDD1F0		CALL	C0	
F5AD 0E3D		MVI	C,'='	
F5AF CDD1F0		CALL	C0	
F5B2 7E		MOV	A,M	
F5B3 23		INX	H	
F5B4 EB		XCHG		
F5B5 6F		MOV	L,A	
F5B6 2603		MVI	H,0	
F5B8 39		DAD	SP	
F5B9 EB		XCHG		
F5BA 46		MOV	B,M	
F5BB 23		INX	H	
F5BC 1A		LDAX	D	
F5BD CD95F1		CALL	LBYTE	
F5C0 05		DCR	B	
F5C1 CAA2F5		JZ	X7	
F5C4 1B		DCX	D	
F5C5 1A		LDAX	D	
F5C6 CD95F1		CALL	LBYTE	
F5C9 C3A2F5		JMP	X7	
F5CC 410701	ACTBL:	DB	'A',	ALOC+2, 1
F5CF 420501		DB	'B',	BLOC+2, 1
F5D2 430401		DB	'C',	CLOC+2, 1
F5D5 440301		DB	'D',	DLOC+2, 1
F5D8 450201		DB	'E',	ELOC+2, 1
F5DB 460601		DB	'F',	FLOC+2, 1
F5DE 481201		DB	'H',	HLOC+2, 1

F5E1 4C1101	DB	'L',	LLOC+2, 1
F5E4 4D1202	DB	'M',	HLOC+2, 2
F5E7 501502	DB	'P',	PLOC+2, 2
F5EA 530902	DB	'S',	SLOC+2, 2
F5ED FF	DB	-1	
0000	END		

```

;          SYSTEM CONFIGURATION PACKAGE

F000          ORG          0F600H

;          LOGICAL DEVICE/DEVICE DRIVER TABLES
;
;          EACH 4 ENTRY TABLE LISTS THE ADDRESSES
;          OF THE DRIVER ROUTINES TO BE USED FOR
;          THE PHYSICAL DEVICES WHICH MAY ASSIGNED
;          TO THAT LOGICAL DEVICE.

IOTAB:

;          CONSOLE STATUS

;          RETURN WITH REGISTER A = 0 IF NO
;          CONSOLE CHARACTER AVAILABLE.

F000 A0F6      CSTAB:  DW      TTST      ;0
F002 7FF6      DW      KYST      ;1
F004 7FF6      DW      KYST      ;2
F006 7FF6      DW      KYST      ;3

;          CONSOLE INPUT

;          RETURN CONSOLE INPUT CHARACTER
;          IN REGISTER A.

F008 A8F6      CITAB:  DW      TTI       ;0
F00A 65F6      DW      KYBD      ;1
F00C 66F6      DW      KYBD      ;2
F00E 66F6      DW      KYBD      ;3

;          CONSOLE OUTPUT

;          OUTPUT BYTE IN REGISTER C
;          TO CONSOLE OUTPUT DEVICE.

F010 B7F6      COTAB:  DW      TTO       ;0
F012 B7F6      DW      TTO       ;1
F014 D4F6      DW      THRM      ;2
F016 59F6      DW      CRT       ;3

;          READER INPUT

;          RETURN READER INPUT BYTE IN
;          REGISTER A, CARRY OFF.  SET
;          CARRY IF NO BYTE AVAILABLE.

```

```

F618 C2F6      RITAB:  DW      TTR      ;0
F61A 87F6      DW      RDR      ;1
F61C 66F6      DW      KYBD     ;2
F61E F0B8      DW      0B8F0H   ;3 DISK READ

```

```

;      PUNCH OUTPUT

```

```

;      OUTPUT BYTE IN REGISTER C
;      TO PUNCH DEVICE.

```

```

F620 B7F6      POTAB:  DW      TTO      ;0
F622 DFF6      DW      PUNCH     ;1
F624 59F6      DW      CRT       ;2
F626 73B9      DW      0B973H   ;3 DISK WRITE

```

```

;      LISTING OUTPUT

```

```

;      OUTPUT BYTE IN REGISTER C
;      TO LISTING DEVICE.

```

```

F628 B7F6      LOTAB:  DW      TTO      ;0
F62A 59F6      DW      CRT       ;1
F62C D4F6      DW      THRM      ;2
F62E B7F6      DW      TTO      ;3

```

```

;      SPECIAL SUBROUTINE TO LOCATE MONITOR
;      SCRATCH RAM

```

```

;      THE ADDRESS OF THE TOP OF THE SCRATCH
;      RAM AREA USED BY THE MONITOR IS RETURNED
;      IN REGISTERS D,E.

```

```

;NOTE:  THIS SUBROUTINE IS NOT CALLED IN THE
;      USUAL WAY:  INSTEAD, THE RETURN ADDRESS
;      IS PLACED IN REGISTERS D,E AND THE
;      SUBROUTINE IS ENTERED BY A JUMP INSTRUCTIO
;      RETURN IS DONE BY PLACING THE RETURN
;      ADDRESS IN H,L AND EXECUTING A PCHL INST.

```

```

F630 C33DF6    ADSCS:  JMP      ADS2

```

F633 C34FF6 ADSCR: JMP ADS1

```

; SUBROUTINE TO SET ADDRESS
; OF IOBYT
;
; THE ADDRESS OF THE BYTE USED TO
; RECORD THE CURRENT PHYSICAL/LOGICAL
; DEVICE ASSIGNMENTS IS RETURNED IN
; REGISTERS H,L.

```

F636 C34FF6 ADIOB: JMP ADS1

```

; SUBROUTINE TO SET THE USER STACK
; ADDRESS.
;
; THE ADDRESS TO BE USED AS THE
; DEFAULT VALUE OF THE USER STACK
; ADDRESS IS RETURNED IN REGISTERS H,L.

```

F639 218000 ADUST: LXI H, 80H
F63C C9 RET

F63D 213000 ADS2: LXI H, 0
F640 24 ADS3: INR H
F641 7E MOV A, M
F642 2F CMA
F643 F3 DI
F644 77 MOV M, A
F645 BE CMP M
F646 2F CMA
F647 FB EI
F648 77 MOV M, A
F649 CA40F6 JZ ADS3
F64C 2B DCX H,
F64D EB XCHG
F64E E9 PCHL

F64F D5 ADS1: PUSH D
F650 12FF00 LXI H, OFFH
F653 000000 NOP
F656 00 NOP
F657 D1 POP D
F658 C9 RET

```

; PHYSICAL DEVICE DRIVER ROUTINES
;
; REQUIREMENTS
; MAINTAIN CONTENTS OF ALL
; REGISTERS EXCEPT A AND F.
; EXIT BY RETURN INST.

```

;
VIDEO DRIVER

F65J 79	CRT:	MOV	A,C	
F65A B7		ORA	A	;CHECK FOR NULL
F65B C8		RZ		
F65C E5		PUSH	H	
F65D CD36F6		CALL	AD10B	
F660 2B		DCX	H	
F661 2B		DCX	H	
F662 2B		DCX	H	
F663 C3B4F7		JMP	0F7B4H	

;
KEYBOARD DRIVER

F666 DB02	KYBD:	IN	2	
F668 E601		ANI	1	
F66A C266F6		JNZ	KYBD	
F66D DB03		IN	3	
F66F E67F		ANI	7FH	
F671 FE61		CPI	61H	;LOWER CASE A
F673 DA7DF6		JC	KBI	
F676 FE7B		CPI	7AH:1	;LOWER CASE Z +1
F678 D27DF6		JNC	KBI	
F67B E6DF		ANI	0DFH	;DELET ONE BIT
F67D B7	KBI:	ORA	A	
F67E C9		RET		

;
KEYBOARD STATUS DRIVER

F67F DB02	KYST:	IN	2
F681 E601		ANI	1
F683 D601		SUI	1
F685 9F		SBB	A
F686 C9		RET	

;
READER DRIVER

F687 E5	RDR:	PUSH	H
F688 210000		LXI	H,0
F68B DB04	RD:	IN	4
F68D E601		ANI	1
F68F CA9BF6		JZ	RD2
F692 2B		DCX	H
F693 7C		MOV	A,H
F694 B5		ORA	L
F695 C28BF6		JNZ	RD
F698 37		STC	
F699 E1		POP	H
F69A C9		RET	

F69B DB05	RD2:	IN	5
F69D B7		ORA	A
F69E E1		POP	H
F69F C9		RET	

TELETYPE STATUS DRIVER

F6A0 DB03	TTST:	IN	3
F6A2 E602		ANI	2
F6A4 D602		SUI	2
F6A6 9F		SBB	A
F6A7 C9		RET	

TELETYPE INPUT DRIVER

F6A8 AF	TTI:	XRA	A
F6A9 D300		OUT	0
F6AB DB03	TTI1:	IN	3
F6AD E602		ANI	2
F6AF CAABF6		JZ	TTI1
F6B2 DB02		IN	2
F6B4 E67F		ANI	7FH
F6B6 C9		RET	

TELETYPE OUTPUT DRIVER

F6B7 DB03	TT0:	IN	3
F6B9 E601		ANI	1
F6BB CAB7F6		JZ	TT0
F6BE 79		MOV	A,C
F6BF D302		OUT	2
F6C1 C9		RET	

TELETYPE READER DRIVER

F6C2 3E01	TTR:	MVI	A,1
F6C4 D300		OUT	0
F6C6 3E00		MVI	A,0
F6C8 D300		OUT	0
F6CA DB00	TTR1:	IN	0
F6CC E601		ANI	1
F6CE C2CAF6		JNZ	TTR1
F6D1 DB01		IN	1
F6D3 C9		RET	

```

;          THERMAL PRINTER DRIVER

F6D4 DB02    THRM:  IN      2
F6D6 E680    ANI      80H
F6D8 C2D4F6  JNZ      THRM
F6DB 79      MOV      A,C
F6DC D303    OUT      3
F6DE C9      RET

;          PUNCH DRIVER

F6DF DB04    PUNCH: IN      4
F6E1 E680    ANI      80H
F6E3 C2DFF6  JNZ      PUNCH
F6E6 79      MOV      A,C
F6E7 D305    OUT      5
F6E9 C9      RET

```

CP/M ASSEMBLER - VER 1.0

```

F800          ORG      OF800H
F800 2109F8    ENTRY:  LXI      H,MESS
F803 CD1EF0    CALL     STRING
F806 C321F0    JMP      RENT
F01E =        STRING: EQU     OF01EH

F809 0D0A2020MESS: DB      0DH,0AH,'      HELLO !!  YOU HAVE ENTERED THE '
F830 0A0D574F52 DB      0AH,0DH,'WORLD OF DIAGNOSTICS.  THIS LIST WILL ACQUAINT'
F860 0D0A594F55 DB      0DH,0AH,'YOU WITH SOME OF THE COMMANDS OF THE DIAGNOSTIC'
F891 0D0A4F5045 DB      0DH,0AH,'OPERATING SYSTEM.  MANY OF THE FUNCTIONS ARE VERY'
F8C4 0D0A53494D DB      0DH,0AH,'SIMILAR TO CPM/DDT.'
F8D9 0D0A434F4D DB      0DH,0AH,'COMMAND                                FUNCTION'
F8FC 0DCA      DB      0DH,0AH
F8FE 0D0A202020 DB      0DH,0AH,'      A                      ASSIGNS I/O DEVICES ( PHYSICAL'
F92E 0D0A202020 DB      0DH,0AH,'                      TO LOGICAL DEVICE )'
F956 0D0A      DB      0DH,0AH
F958 0D0A202020 DB      0DH,0AH,'      B                      DUMP MEMORY IN BINARY ON PUNCH DEVICE'
F98F 0D0A      DB      0DH,0AH
F991 0D0A202020 DB      0DH,0AH,'      C                      HEXADECIMAL ARITHMETIC'
F989 0D0A      DB      0DH,0AH
F98B 0D0A202020 DB      0DH,0AH,'      D                      DISPLAY A BLOCK OF MEMORY'
F9E6 0D0A      DB      0DH,0AH
F9E8 0D0A202020 DB      0DH,0AH,'      F                      FILLS A BLOCK OF MEMORY WITH A CONSTANT'
FA21 0D0A      DB      0DH,0AH
FA23 0D0A202020 DB      0DH,0AH,'      G                      GO TO ADDRESS AND EXECUTE, OPTIONAL'
FA58 0D0A202020 DB      0DH,0AH,'                      BREAK POINTS.'
FA7A 0A0D      DB      0AH,0DH
FA7C 0D0A202020 DB      0DH,0AH,'      H                      HELP, THIS PROGRAM'
FAA0 0D0A      DB      0DH,0AH
FAA2 0D0A202020 DB      0DH,0AH,'      K                      COPY FROM READER TO PUNCH'
FACD 0D0A      DB      0DH,0AH
FACF 0D0A202020 DB      0DH,0AH,'      L                      LOAD BINARY TAPE, OPTIONAL BIAS'
FB00 0D0A      DB      0DH,0AH
FB02 0D0A202020 DB      0DH,0AH,'      M                      MOVE A BLOCK OF MEMORY TO ANOTHER LOC'
FB39 0D0A      DB      0DH,0AH
FB3B 0D0A202020 DB      0DH,0AH,'      N                      OUTPUTS 60 NULLS TO PUNCH DEVICE'
FB6D 0D0A      DB      0DH,0AH
FB6F 0D0A202020 DB      0DH,0AH,'      R                      LOAD A HEX TAPE FROM READER DEVICE'
FBA3 0D0A      DB      0DH,0AH
FBA5 0D0A202020 DB      0DH,0AH,'      S                      DISPLAY AND CHANGE ANY MEM LOC'
FBD5 0D0A      DB      0DH,0AH
FBD7 0D0A202020 DB      0DH,0AH,'      T                      TEST LIST AND EXECUTION PROGRAM'
FC08 0D0A      DB      0DH,0AH
FC0A 0D0A202020 DB      0DH,0AH,'      W                      DUMP MEMORY IN HEX ON PUNCH DEVICE'
FC3E 0D0A      DB      0DH,0AH
FC40 0D0A202020 DB      0DH,0AH,'      X                      CPU REGISTER DISPLAY AND CHANGE',0
F021 =        RENT:   EQU     OF021H
;
;
;
FD00          ORG      OFD00H
FD00 0BFF      ENTRY1: IN      OFFH
FD02 EC11      ANI      01H
FD04 C213FD    JNZ      FART
FD07 219AFD    ENTRY2: LXI      H,MESS2
FD0A CD1EF0    CALL     STRING      ;PRINT LIST OF TESTS
FD0D 2153FD    ENTRY3: LXI      H,MESS3
FD10 CD1EF0    CALL     STRING      ;PROMT FOR INPUT
FD13 CD2FF1    FART:   CALL     TI
FD16 FE03      CPI      03H
FD18 CA21F0    JZ       RENT
FD1B FE40      CPI      40H
FD1D DA2EFD    JC       NUM
FD20 FE50      CPI      50H
FD22 DA3EFD    JC       LETTER

```

```

FD25 214DFD ; ERROR: LXI H,MESS4
FD28 CD1EFO ; CALL STRING ;PRINT ERROR >?
FD2B C30DFD ; JMP ENTRY3 ;REPRINT
;
;
FD2E FE30 NUM: CPI 30H ;SEE IF NUMBER
FD30 DA25FD JC ERROR ;JUMP IF NO
FD33 FE3A CPI 3AH
FD35 E60F ANI 0FH ;REMOVE LEAD NIBBLE
FD37 87 ADD A ;DOUBLE FOR TABLE LOOK UP
FD38 21DDFE LXI H,NUMTAB ;NUMBER TABLE
FD3B C344FD JMP COMMON
;
;
FD3E E60F LETTER: ANI 0FH
FD40 87 ADD A ;DOUBLE FOR TABLE
FD41 21F1FE LXI H,LETTAB ;LETTER TABLE
FD44 1600 COMMON: MVI D,0
FD46 5F MOV E,A
FD47 19 DAD D ;ADD OFFSET TO TABLE ADDRESS
FD48 7E MOV A,M
FD49 23 INX H
FD4A 66 MOV H,M
FD4B 6F MOV L,A
FD4C E9 PCHL ;JUMP TO TEST PROGRAM
;
;
FD4D 0D0A203F20MESS4: DB 0DH,0AH,' ? ',0
FD53 0D0A454E54MESS3: DB 0DH,0AH,'ENTER TEST ID NO. TO RUN TEST'
FD73 0D0A454E54 DB 0DH,0AH,'ENTER CONTROL C TO RETURN TO MONITOR',0
FD9A 0D0A2A2054MESS2: DB 0DH,0AH,0AH,' TESTS AVAILABLE'
FDAD 0D0A202031 DB 0DH,0AH,' 1 - COMPREHENSIVE MEMORY TEST'
FDCF 0D0A202032 DB 0DH,0AH,' 2 - MINI-MEMORY 0 - .1K'
FDEC 0D0A202033 DB 0DH,0AH,' 3 - MINI-MEMORY 0 - 8K'
FE09 0D0A202034 DB 0DH,0AH,' 4 - MINI-MEMORY 0 - 24K'
FE26 0D0A202035 DB 0DH,0AH,' 5 - FORMATTED DISK TEST'
FE42 0D0A202036 DB 0DH,0AH,' 6 - DISK TRACK READ'
FE5A 0D0A202037 DB 0DH,0AH,' 7 - DISK TRACK WRITE'
FE73 0D0A202038 DB 0DH,0AH,' 8 - UNIBUS PORT TEST'
FE8C 0D0A202039 DB 0DH,0AH,' 9 - UNIBUS COMMUNICATION TEST'
FEAE 0D0A202041 DB 0DH,0AH,' A - UNIBUS SNAPSHOT'
FEC6 0D0A202042 DB 0DH,0AH,' B - DISPLAY TESTS',0
;
;ADD MORE TO DIRECTORY HERE
;
F12F = ; TI: EQU 0F12FH
;
FEDD 25FD ; NUMTAB: DW ERROR
FEDF 00C0 DW 0C000H ;MEMORY TEST
FEE1 90C2 DW 0C290H ;MINI .1K
FEE3 00D6 DW 0D600H ;MINI 8K
FEE5 00D7 DW 0D700H ;MINI 24K
FEE7 00C8 DW 0C800H ;FOMAT DSK
FEE9 40CE DW 0CE40H ;TRK RD
FEEB 80CD DW 0CD80H ;TRK WRT
FEED 00D0 DW 0D000H ;UB PORT
FEF 00D1 DW 0D100H ;UB COMM
FEF1 25FD LETTAB: DW ERROR
FEF3 00D5 DW 0D500H ;SNAPSHOT
FEF5 00E0 DW 0E000H ;DISPLAY
FEF7 25FD DW ERROR
FEF9 25FD DW ERROR
FEFB 25FD DW ERROR
FEFD 25FD DW ERROR
FEFF 25FD DW ERROR
FF01 25FD DW ERROR
FF03 25FD DW ERROR
FF05 25FD DW ERROR
FF07

```

END

DATE
FILMED

1-81

DTIC