**Title**: Diagnostic Software Development for a Real-Time Spectral Analysis System

**Authors**: Brian J. Donlan

**Performing Organization**: Rensselaer Polytechnic Institute

**Report Date**: April 1980

**Number of Pages**: 105

**Security Class**: UNCLASS

**Distribution Statement**: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

**Supplementary Notes**: APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

**Key Words**: (Continue on reverse side if necessary and identify by block number) ATTACHED

**Abstract**: (Continue on reverse side if necessary and identify by block number)
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.
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
the purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (AFIT). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

Research Title: Diagnostic Software Development for a Real-Time Spectral Analysis System

Author: Brian J. Donlan

Research Assessment Questions:

1. Did this research contribute to a current Air Force project?
   a. Yes  b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?
   a. Yes  b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?
   a. Man-years  b. $_______

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3 above), what is your estimate of its significance?


5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the back of this questionnaire for your statement(s).

NAME  GRADE  POSITION

ORGANIZATION  LOCATION  USAF SCN 75-201
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>viii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>ix</td>
</tr>
<tr>
<td>1. INTRODUCTION-HISTORY</td>
<td>1</td>
</tr>
<tr>
<td>2. SYSTEM DESCRIPTION</td>
<td>3</td>
</tr>
<tr>
<td>2.1 General Description-Data Flow</td>
<td>3</td>
</tr>
<tr>
<td>2.2 IMSAI Microcomputer</td>
<td>3</td>
</tr>
<tr>
<td>2.3 Floating Point AP-120B Array Processor</td>
<td>4</td>
</tr>
<tr>
<td>2.4 Data Acquisition</td>
<td>5</td>
</tr>
<tr>
<td>2.5 Color Graphics Display</td>
<td>6</td>
</tr>
<tr>
<td>3. TECHNICAL DISCUSSION</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Task</td>
<td>8</td>
</tr>
<tr>
<td>3.2 Design Details</td>
<td>8</td>
</tr>
<tr>
<td>3.3 Test Procedures</td>
<td>10</td>
</tr>
<tr>
<td>4. DIAGNOSTIC OPERATING SYSTEM</td>
<td>12</td>
</tr>
<tr>
<td>4.1 General Description</td>
<td>12</td>
</tr>
<tr>
<td>4.2 Detailed Description</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Program Usage</td>
<td>12</td>
</tr>
<tr>
<td>4.4 Commands</td>
<td>13</td>
</tr>
<tr>
<td>4.5 Externally Referenced Subroutines</td>
<td>21</td>
</tr>
<tr>
<td>5. MINI-MEMORY TEST</td>
<td>24</td>
</tr>
<tr>
<td>5.1 General Description</td>
<td>24</td>
</tr>
<tr>
<td>5.2 Program Details</td>
<td>24</td>
</tr>
<tr>
<td>5.3 Operation</td>
<td>25</td>
</tr>
<tr>
<td>6. COMPREHENSIVE MEMORY TEST</td>
<td>34</td>
</tr>
<tr>
<td>6.1 General Description</td>
<td>34</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.2 Program Details</td>
<td>34</td>
</tr>
<tr>
<td>6.3 Operation</td>
<td>34</td>
</tr>
<tr>
<td>7. FORMATTED DISK TEST</td>
<td>42</td>
</tr>
<tr>
<td>7.1 General Description</td>
<td>42</td>
</tr>
<tr>
<td>7.2 Program Details</td>
<td>42</td>
</tr>
<tr>
<td>8. TRACK WRITE ROUTINE</td>
<td>53</td>
</tr>
<tr>
<td>8.1 General Description</td>
<td>53</td>
</tr>
<tr>
<td>8.2 Program Details</td>
<td>53</td>
</tr>
<tr>
<td>8.3 Operation</td>
<td>54</td>
</tr>
<tr>
<td>9. TRACK READ ROUTINE</td>
<td>62</td>
</tr>
<tr>
<td>9.1 General Description</td>
<td>62</td>
</tr>
<tr>
<td>9.2 Program Details</td>
<td>62</td>
</tr>
<tr>
<td>9.3 Operation</td>
<td>63</td>
</tr>
<tr>
<td>10. UNIBUS PORT TEST</td>
<td>70</td>
</tr>
<tr>
<td>10.1 General Description</td>
<td>70</td>
</tr>
<tr>
<td>10.2 Program Details</td>
<td>70</td>
</tr>
<tr>
<td>10.3 Operation</td>
<td>71</td>
</tr>
<tr>
<td>11. UNIBUS COMMUNICATION TEST</td>
<td>76</td>
</tr>
<tr>
<td>11.1 General Description</td>
<td>76</td>
</tr>
<tr>
<td>11.2 Program Details</td>
<td>76</td>
</tr>
<tr>
<td>12. UNIBUS SNAPSHOT</td>
<td>86</td>
</tr>
<tr>
<td>12.1 General Description</td>
<td>86</td>
</tr>
<tr>
<td>12.2 Program Details</td>
<td>86</td>
</tr>
<tr>
<td>12.3 Operation</td>
<td>86</td>
</tr>
<tr>
<td>13. DIAGNOSTIC MEMORY BOARD</td>
<td>92</td>
</tr>
<tr>
<td>13.1 General Description</td>
<td>92</td>
</tr>
<tr>
<td>13.2 Detailed Description</td>
<td>92</td>
</tr>
<tr>
<td>14. CONCLUSIONS</td>
<td>94</td>
</tr>
</tbody>
</table>
15. LITERATURE ................................................. 96
APPENDIX A: Memory Board Diagrams .................. 97
APPENDIX B: Commonly Used Subroutines .......... 101
APPENDIX C: PROM Program Listings ............... 105
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Command Summary</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Phase Messages</td>
<td></td>
</tr>
<tr>
<td>11.1</td>
<td>Unibus Addresses</td>
<td></td>
</tr>
<tr>
<td>13.1</td>
<td>Memory Map</td>
<td></td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1 System Block Diagram .....................
Figure 10.1 Unibus Adapter Ports .....................
ACKNOWLEDGMENT

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 Rensslelar 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 annotation purposes are provided at the top of the screen. Under the two lines of annotation 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 program 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 probably 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 instructions 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 transferred 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 functions 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 comma 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'

13
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 inputed 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 Mitsu 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 (00H) 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'.

19
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.

IOCHX '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.

STRING '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.
<table>
<thead>
<tr>
<th>Sense Lite Display Hex</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Phase 1 Running-no errors yet</td>
</tr>
<tr>
<td>02</td>
<td>Phase 2 Running-no errors yet</td>
</tr>
<tr>
<td>03</td>
<td>Phase 3 Running-no errors yet</td>
</tr>
<tr>
<td>F1</td>
<td>Error Phase 1</td>
</tr>
<tr>
<td>F2</td>
<td>Error Phase 2</td>
</tr>
<tr>
<td>F3</td>
<td>Error Phase 3</td>
</tr>
<tr>
<td>FF</td>
<td>Test complete-no errors, move any sense switch to restart</td>
</tr>
</tbody>
</table>

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
LXI H, 0000H

LP2:
XRA A
MOV M, A
MOV B, M
CMP B
JNZ ERR1
INR A
JNZ LP1
INX H
LXI D, OFF00H
XCHG

; OUTPUT

; PHASE I LITES

OUT OFFH
LXI H, 0000H

LP1:
MOV M, A
MOVA
CMP s
JNZ ERR1
INR A
JNZ LP1
INX H
LXI D, OFF00H
XCHG

; START ADDRESS

; NEW TEST PATTERN

; CHECK FOR ERROR

; ADD TWO'S COMPLEMENT

; STOP ADDRESS

; PHASE II LITES

OUT OFFH
LXI H, 0000H

LP3:
MOV M, H
INX H
LXI D, OFF00H
XCHG

; LOW ADDRESS TO MEM

; STOP ADDRESS

; READ MEMORY

LXI H, 0000H

LP4:
MOV A, M
SUB H
JNZ ERR2
INX H
LXI D, OFF00H
XCHG

; READ MEMORY

; COMPARE

; JUMP IF ERROR

; ADD TWO'S COMPLEMENT

; ADD TWO'S COMPLEMENT

; PHASE III LITES

; READ MEMORY

MVI A, OFCH
OUT OFFH
LXI H, 0000H

LP5:
MOV M, L
INX H
LXI D, OFF00H
XCHG

; STORE HIGH ADDRESS IN ALL MEM

; READ MEMORY

LXI H, 0000H

LP6:
MOV A, L
SUB H
JNZ ERR3
INX H
LXI D, OFF00H
XCHG

; READ MEMORY

; COMPARE

; ADD TWO'S COMPLEMENT

; ADD TWO'S COMPLEMENT

; READ MEMORY
; 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
MVI A, OF1H ; PHASE I ERROR 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
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, F
MVI A, OF3H ; PHASE II ERROR TO LITES
LXI H, COMERR ; RETURN
JMP LITES

;; LITES ROUTINE ENTER WITH RETURN IN REG HAL DATA FOR LITES IN A

LITES: CMA OUT OFFH ; OUTPUT LITES
SPHL IN OFFH ; SAVE RETURN IN SP
LXI H, OFC18H ; DELAY LOOP
JZ LP8
LXI H, OFC18H ; ZEROS IN H
XRA H
ORA H
JNZ LP8
LXI H, OFFH ; READ SENSE SWITCHES
XRA H
JZ LP7
MOV A, H ; READ SWITCHES
XRA H
JNZ LP7
INX H
XRA H
JZ LP6
MOV A, H ; SEE IF THEY CHANGED
XRA H
JNZ LP6
INX H
XRA H
JZ LP5
MOV A, H ; READ SENSE SWITCHES
XRA H
JNZ LP5
INX H
XRA H
JZ LP4
MOV A, H ; READ SENSE SWITCHES
XRA H
JNZ LP4
INX H
XRA H
JZ LP3
MOV A, H ; READ SENSE SWITCHES
XRA H
JNZ LP3
INX H
XRA H
JZ LP2
MOV A, H ; READ SENSE SWITCHES
XRA H
JNZ LP2
INX H
XRA H
JZ LP1
MOV A, H ; READ SENSE SWITCHES
XRA H
JNZ LP1
INX H
XRA H
JZ LP0
MOV A, H ; READ SENSE SWITCHES
XRA H
JNZ LP0
INX H
XRA H
JZ LP8
MOV H, A ; MOVE RETURN BACK TO H & L
JMP LITES

29
; 8K MINI MEMORY TEST
;
; BRIAN DONLAN
; PROM VERSION
ORG OD600H
ENTRY:
DI
MV1 A,OFEH ;OUTPUT PHASE I LITES
OUT OFFH ;START ADDRESS
LXI H,0000H
LP2:
XRA A ;ZERO ACC
MOV M,A ;STORE TEST PATTERN IN MEM.
CMP B ;COMPARE FOR OK
JNZ ERR1 ;JUMP IF ERROR
INR A ;NEW TEST PATTERN
INX H
LXI D,0000H ;STOP ADDRESS
XCHG D ;ADD TWO'S COMPLIMENT
JNC LP2

; PHASE II

MV1 A,OFDH ;PHASE II LITES
OUT OFFH
LXI H,0000H
LP3:
MOV M,H ;LOW ADDRESS TO MEM
INX H
LXI D,0000H ;STOP ADDRESS
XCHG D
DAD D
XCHG JNC LP3
; READ MEMORY

LXI H,0000H
MOV A,M ;READ MEMORY
SUB H ;COMPARE
JNZ ERR2 ;JUMP IF ERROR
INX H
LXI D,0000H
XCHG D
DAD D
XCHG JNC LP4

; PHASE III

MV1 A,OFCH ;PHASE THREE LITES
OUT OFFH
LXI H,0000H ;STORE HIGH ADDRESS IN ALL MEM
LP5:
MOV M,L
INX H
LXI D,0000H
XCHG D
DAD D
XCHG JNC LP5
; READ MEMORY

LXI H,0000H
MOV A,M ;READ MEMORY
SUB L ;COMPARE
JNZ ERR3 ;JUMP IF ERROR
INX H
LXI D,0000H
XCHG D
DAD D
XCHG JNC LP6

30
ALL PHASE COMPLETE
MVI A, OFFH
LXI H, ENTER
JMP LITES
; GO TO LITES PROG

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

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

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

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

; LITES ROUTINE
ENTER WITH RETURN IN REG H&L
DATA FOR LITES IN A
LITES:
CMA OUT OFFH
SPHL IN OFFH
MOV H, A
LXI H, OF18H
LP7: IN OFFH
XRA H
JZ LP7
LXI H, OFC18H
JPCHL ; DELAY LOOP
LP8: INX H
XRA A
ORA H
JNZ LP8
LXI H, 0
DAD SP
; ZERO H
; MOVE RETURN BACK TO H & L
; RETURN

31
; 24K MINI-MEMORY TEST
; PROM VERSION
;
; BRIAN DONLAN
ORG 00700H
ENTRY2: DI
MVI A, OFFH ;OUTPUT PHASE I LITES
OUT OFFH ;START ADDRESS
LXI H, 0000H ;ZERO ACC
LP22: XRA A ;ZERO ACC
LP12: MOV M, A ;STORE TEST PATTERN IN MEM.
MOV B, M ;READ BACK TO B
CMP B ;COMPARE FOR OK
JNZ ERR12 ;JUMP IF ERROR
INR A ;NEW TEST PATTERN
JNZ LP12
INX H
LXI D, 0A000H ;STOP ADDRESS
XCHG DAD D ;ADD TWO'S COMPLIMENT
JNC LP22
; PHASE II
MVI A, 0FCH ;PHASE II LITES
OUT OFFH
LXI H, 0000H
LP32: MOV M, H ;LOW ADDRESS TO MEM
INX H
LXI D, 0A000H ;STOP ADDRESS
XCHG DAD D
JNC LP32
; READ MEMORY
LXI H, 0000H
LP42: MOV A, M ;READ MEMORY
SUB L ;COMPARE
JNZ ERR22 ;JUMP IF ERROR
INX H
LXI D, 0A000H
XCHG DAD D
JNC LP42
; PHASE III
MVI A, 0FCH ;PHASE THREE LITES
OUT OFFH
LXI H, 0000H
LP52: MOV M, L ;STORE HIGH ADDRESS IN ALL MEM
INX H
LXI D, 0A000H
XCHG DAD D
JNC LP52
; READ MEMORY
LXI H, 0000H
LP62: MOV A, M ;READ MEMORY
SUB L ;COMPARE
JNZ ERR32 ;JUMP IF ERROR
INX H
LXI D, 0A000H
XCHG DAD D
JNC LP62
ALL PHASE COMPLETE
MVI A, OFFH
LXI H, ENTER2
JMP LITES2 ; GO TO LITES PROG

: PHASE I ERROR
ERR12: XCHG
MOV C, A ; SAVE BAD DATA
LXI H, COMER2
MVI A, OF1H ; PHASE I ERROR LITES
JMP LITES2

: COMMON ERROR OUTPUT ROUTINE
COMER2: MOV A, D ; HIGH ADDRESS
LXI H, LOADD2
JMP LITES2
LOADD2: MOV A, E ; LOW ADDRESS TO LITES
LXI H, TPA2
JMP LITES2
TPA2: MOV A, C ; TEST PATTERN TO LITES
LXI H, ACTO2A
JMP LITES2
ACTO2: MOV A, B ; ACTUAL DATA TO LITES
LXI H, ENTER2
JMP LITES2

: PHASE II ERROR
ERR22: XCHG
ADD D ; SAVE BAD ADDRESS
MOV B, C
MOV C, D
MVI A, OF2H ; PHASE II ERROR LITES
JMP LITES2

: PHASE III ERROR
ERR32: XCHG
ADD E ; SAVE BAD ADDRESS
MOV B, A
MOV C, E
MVI A, OF3H ; PHASE III ERROR LITES
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
LP72: MOV H, A
IN OFFH ; READ SENSE SWITCHES
XRA H ; SAVE IN H
JZ LP72 ; READ SWITCHES
LP82: XLI H, OFC18H ; SEE IF THEY CHANGED
INX H
XRA A
ORA H
JNZ LP82
LXi H, O
DAD SP ; ZERO H
PCHL SP ; MOVE RETURN BACK TO H & L
JMP LITES2 ; RETURN

33
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 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,OFOM

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 RBIN

XCHG

SMLD START

LXI H,MSG3

CALL PMSG

CALL RBIN

XCHG

SMLD ENADR

LXI H,MSG8

CALL PMSG

IM CDATA ; RESET IO FLAG

BEGIN: MVI C,WO ; LOAD TEST BYTE

MTEST: MVI A,02 ; LOAD TEST BYTE

MTL: CALL STUFF ; STUFF MAJOR ALL OVER

MVI A,02 ; SET TWO AS MINOR

CALL SMTF ; STUFF MINOR

MVI A,02 ; SET 2 AGAIN

CALL CHECK ; NOW CHECK ALL LOC

PICK: LDA PART

DEC A

STA PART ; STORE NEW PART

CPI 00 ; FINISH THIS PASS ?

JZ RECYCLE ; YES

CONT: MVI A,01 ; NO CONTINUE

CALL STUFF ; STUFF MINOR SERT

MOV A,C ; LOAD MAJOR BYTE

CMA ; COMPLEMENT MAJOR BYTE

MOV C,A ; SAVE NEW BYTE

ERA A ; SAVE OTHER TEST BYTE

CALL CHECK

JMP MTLOP

RECYCLE: MOV A,C ; SAVE INVERTED TEST BYTE

STA PART ; SAVE INVERTED TEST BYTE

LXI H,MSG8 ; END OF PASS MESSAGE

CALL PMSG

LDA CODE ; CHAR CODE

ORA A ; SET FLAGS

JNZ ENTRY ; START OVER

AMA A ; CLEAR CARRY

LDA PART ; RECOVER TEST BYTE

ORA A

JZ BEGIN

RAL

CMA

MOV C,A ; NEW TEST BYTE

JMP MTEST ; ANOTHER PASS
START:  DS  2  ;LOC FOR START ADDR
ENDR:   DS  2  ;LOC FOR END ADDRESS
PART:   DS  1  ;LOC FOR PART ADDRESS
CODE:   DS  1

STUFF:  CALL STASO ;LOAD START AND END ADDR
        MOV M,C ;STUFF MAJOR ALL OVER
        CALL HILOX ;SEE IF ALL MEM DONE
        JMP DOIT ;NO KEEP ON STUFFING

DOIT:   STUFF: CALL STASO ;LOAD ADDR AGAIN
        MOV B,A ;MINOR COUNTER
        CPI GO ;MINOR WORD STUFF
        JNZ HIL ;NO
        MOV A,C ;MAJOR TEST BYTE
        CMA M,A ;MINOR IS COMPLIMENT OF MAJOR
        MOV M,A ;STUFF MINOR BYTE IN MEM
        CALL HILOX ;INC & CHK IF DONE
        DCR B ;DEC MINOR COUNTER
        JNZ HIL ;OK TO STUFF NO
        JMP MINR ;YES

CHECK:  CALL STASO ;LOAD START AND END ADDR
        MOV B,A ;MINOR COUNTER
        CPI GO ;COUNT ZERO
        JNZ MAJR ;NO GO TO MAJOR
        MOV A,C ;LOAD TEST BYTE MAJOR
        CMP M ;MINOR IS COMPLIMENT
        MVI B,03 ;START MINOR COUNT AT 3
        JMP CKEND ;CHECK FOR ERROR OR ABORT
        CMP A,C ;LOAD MAJOR TEST BYTE
        MVI B,03 ;START MAJOR COUNT AT 3
        JMP CKEND ;CHECK FOR ERROR OR ABORT
        CMP M ;READ AND COMPARE MEM WITH MAJOR
        JNZ ERR ;GO TO ERR TO PRINT IF ERROR
        POP B ;RESTORE REGS
        IN CSTAT ;CHECK KEYBOARD
        ANI 02H ;SAVE COUNT AND MAJOR
        JZ FIN ;OK
        IN CDATA ;READ KEYS
        STA CODE ;READ KEYS
        STA CODE
        JMP MAJR ;COUNT ZERO DO MAJOR
        JMP MINR ;COUNT ZERO DO MINOR

STASO:  LHLD ENADR ;LOAD END ADDR
        XCHG ;MOVE END TO CAD
        LHLD START ;LOAD START
        RET

ERR:    LHLD ENADR ;SAVE END ADDR
        CALL CRUF
        MOV D,PSW
        MOV E,L
        CALL BLM
        MV1 D,08
        CALL BLM
        POP PSW
        MOV B,A
        CALL BLM
        MV1 D,0AH
        CALL BLM
        MOV A,W
        CALL BLM
        MOV A,B
        CALL BLM
        POP D
        RET
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
SP
RET

DIFF: POP A ;CONTINUE STUFFING
RET

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

MSG1 DB ODH,OAH,'MEMORY TEST',0
MSG2 DB ODH,OAH,'ENTER START ADDRESS',0
MSG3 DB ODH,OAH,'ENTER STOP ADDRESS',0
MSG4 DB ODH,OAH,'END OF PASS',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 ;MAKE MOST SIG. BIT = 0.
RET ;RETURN.

; WRITE A CHARACTER TO THE CONSOLE DEVICE.

CONOT: MVI A,ODM ;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 BAC.
MVI B,CNULL ;GET NULL COUNT.
CALL CONOT1 ;PRINT CR.
MVI C,0 ;GET NULL CHR.
DCR B ;DECEREAMENT COUNTER.
JNZ CONUL1 ;DO NEXT NULL.
POP B ;RESTORE BAC.
MOV A,C ;RESTORE A.
RET ;RETURN.

38
PRINT MESSAGE UNTIL ZERO

MESSAGE ADDRESS REG H : L

PMOSG: MOV A, M ; GET CHAR
ORA A ; IS IT A ZERO
RZ
MOVC, A ; OTHERWISE PRINT
INC H ; INC ADDRESS
JMP PMSG

MOSB DB COM, OAH, OAH, 'LOC. TEST BYTE MEMORY', O

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
ANB B ; AND WITH MASK
JZ PRNT ; JUMP IF ZERO
MVI C, 31H
PRNT:
CALL CONOT
ANB B ; ZERO CARRY
MOV A, E ; LOAD MASK
RAR
JNC OVER
RET

BLNK: MVI C, 20H ; PRINT BLANKS, # IN REG. D

LP1:
CALL CONOT
DCP D
JNZ LP1
RET

BINHA: MOV A, D
PAR
PAR
PAR
PAR
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

BINHA: CALL BINHA
MOV D, E
CALL BINHA
RET
; CONVERT FROM HEX TO ASCII
; INPUT: 4 BIOS HEX REG A
; OUTPUT: 8 BIT ASCII REG A

: BIN1:
  ANI OFH
  CPI 30H
  RC
  ADI 0TH
  RET

: INPJTS
  4 DIGITS FROM CONSOLE
  RETURN: 4 HEX DIGITS IN REG E-D

: BBIN:
  CALL CONIN
  CALL AHS1
  RAL
  RAL
  RAL
  RAL
  ANI OFOH
  MOV D, A
  CALL CONIN
  CALL AHS1
  ANI OFH
  ORA D
  MOV D, A
  CALL CONIN
  CALL AHS1
  RAL
  RAL
  RAL
  RAL
  ANI OFOH
  MOV E, A
  CALL CONIN
  CALL AHS1
  ANI OFH
  ORA E
  MOV E, A
  RET

: CONV ASCII TO HEX
; INPUT: 8 BIT ASCII REG A
; OUTPUT: 4 BIT HEX REG A

: AHS1:
  NOP
  SUI 30H
  CPI 0AH
  RC
  SUI 0TH
  RET

40
INITIATE SIO PORTS

INITIA: MVI A,0A8H ;GET DUMMY MODE WORD
        OUT CSTAT ;OUTPUT IT
        MVI A,90H ;GET RESET BIT
        OUT CSTAT ;RESET SIO BOARD
        MVI A,0CCH ;GET REAL MODE WORD
        OUT CSTAT ;SET THE MODE FOR REAL
        MVI A,37H ;GET THE COMMAND
        OUT CSTAT ;OUTPUT IT

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 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
CPI 'Y'
;CHECK IF Y
JNZ ENTRY
; ?? START OVER
CALL CRLF

LOOP6:
CALL CRLF
XRA A
;ZERO ACC
STA ERRFLG
;ZERO ERROR FLAG
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
;SET UP TO DO PAIRS
STA INNER
;START PAIRS WITH TRK07

TEST FOR CONSOLE INTERRUPT
LOOP6:
IN CSTAT
ANI 00H
JZ LOOP3
;KEYBOARD READY
IN CDATA
CPI 03H
;READ KEYS
CPI 02H
;CONTROL C
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
;FIND NEXT OUTER TRK
ADI 3B
STA OUTER
;STORE OUTER TRK
CPI 77
JNZ LOOP8
;TRK 77 YET ?
JNZ LOOP4
;NOT DONE YET
LDA LPCNT
INR A
STA LPCNT
JMP LOOP4

PATTERN ROUTINE EXPANDABLE
LDA LPCNT
JZ IST
;LOAD LOOP COUNTER
JZ SECd
;SECOND PASS
CPI 01
IST
JZ THIRD
CALL PMSG
IN CSTAT
;CHECK KEYBOARD
ANI 02H
JZ LOOP6
HLT
JMP ENTRY

IST:
MVI A, OFFH
;ALL ONES PATTERN
STA PATEN
;STORE PATTERN

SECd:
MVI A, OOH
;ALL ZERO PATTERN
STA PATEN
;STORE PATTERN

THIRD:
MVI A, 55H
;ALTER PATTERN
STA PATEN
45
WRITE INNER TRK

INWR: LDA INNER
STA TRK
BOTH: CALL SEEK
MVI A,01
STA SECT

LOOP1:
XRA A
STA REPETE
CALL WRITE
LDA REPETE
ORA A
JNZ LOOP1
LDA SECT
INR A
STA SECT
CPI 27
JNZ LOOP1

WRITE OUTER TRK

OUTWR: LDA OUTER
JMP BOTH

READ INNER TRK

INRD: LDA INNER
BOTH2: CALL SEEK
MVI A,01
STA SECT

LOOP5:
XRA A
STA ERRFLG
STA REPETE
CALL READ
XRA A
STA ERRFLG
LDA REPETE
ORA A
JNZ LOOP5
LDA SECT
INR A
STA SECT
CPI 27
JNZ LOOP5

READ OUTER TRK

OUTRD: LDA OUTER
JMP BOTH2

ERRPWT:
LXI H,MSG3
CALL PM5G
LDA ERRFLG
MOV D,A
CALL BINHA
LXI H,MSG4
CALL PM3G
MVI D,01
CALL BLNK
LDA TRK
MOV D,A
CALL BINHA
MVI D,16
CALL BLNK
LDA SECT
MOV D,A

46
CALL BINHA ;PRINT SECTOR NO.
MVI D,13 ;SPACE OVER
CALL BLNK
LDA PATEN
CALL BITS ;PRINT TEST PATTERN
MVI D,12 ;SPACE
CALL BLNK
LDA BADBT ;LAST BAD BYTE
CALL BITS ;PRINT LAST BAD BYTE

; LOOP COUNTER
LPCT: DS 1
; INNER TRK NO.
INNER: DS 1
; SPACE FOR INNER TRK NO.
; TEST PATTERN
PATEN: DS 1
; SPACE FOR TEST PATTERN
ERRFLG: DS 1
; ERROR COUNT
BADBT: DS 1
; BAD BYTE
BDTRK: DS 1
; DISK READ TRK WHEN ERR
REPETE: DS 1
; REPEAT FLAG

MSG1: DB ODH,OAH,'DISK TEST NO. 1 FORMATTED TEST ',0
MSG1A: DB ODH,OAH,' LOAD SCRATCH DISK TYPE Y WHEN READY ',0
MSG2: DB ODH,OAH,' END OF PASS ',0
MSG3: DB ODH,OAH,'DATA ERROR ON DISK CHECK ERROR COUNT IN HEX ',0
MSG4: DB ODH,OAH,' LAST ERROR ',0
MSG5: DB ODH,OAH,'HEAD POSITION ',0
MSG6: DB ODH,OAH,'DISK TRACK ',0
MSG7: DB ODH,OAH,'EXECUTION STOPPED ',0
MSG8: DB ODH,OAH,'TYPE R TO RETRY, C TO CONTINUE, ANYTHING ELSE STOP ',0

; CONSOLE STATUS PORT.
CSTAT EQU 3
; CONSOLE COMMAND PORT.
CCOM EQU 3
; CONSOLE DATA PORT.
CDATA EQU 2
; KEYBOARD READY BIT.
CKBR EQU 00000010B
; PRINT READY BIT.
CPTR EQU 00000001B
; CONSOLE NULL COUNT.
CNULL EQU 1
; DISK BASE ADDRESS.
DISK EQU OF8H
; DISK COMMAND PORT.
DSTAT EQU DISK+2
; DISK TRACK PORT.
TRACK EQU DISK+1
; DISK SECTOR PORT.
SECTP EQU DISK+2
; DISK DATA PORT.
DDATA EQU DISK+3
; DISK CONTROL PORT.
WAIT EQU DISK+4
; DISK CONTROL PORT.
DCONT EQU DISK+4
; ADDRESS FOR TRACK
TRK: DS 1
; ADDRESS FOR SECTOR
SECT: DS 1

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

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
; CONOUT:
MVI A,ODH ;IF IT'S A CR.
CMP C ;THEN HOP OUT
JZ CONOUT ;TO NULL ROUTINE.
; CONOUT:
IN CSTAT ;READ CONSOLE STATUS.
ANI CPTR ;IF NOT READY.
JZ CONOUT ;READY WHEN HIGH.
MOV A,C ;GET CHARACTER.
OUT CDATA ;PRINT IT.
RET ;RETURN.
MOVE DISK TO TRACK ZERO.

HOME:       MVI A,00H  ;CLEAR ANY PENDING COMMAND.
            OUT DCOM
            XRA A  ;ZERO ACC
            STA TRK  ;STORE TRACK NUMBER.
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 DRO.
            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   ;SAVE IN REGISTER.
            ANI 91H   ;MASK NON-ERROR BITS.
            JMP ER MSG  ;DO COMMON ERROR MSGS.

SELECT DISK NUMBER.
INTDSK:     MVI A,02   ;DRIVE NO. 1
            DSXI:  OUT DCONT  ;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,00H  ;READ BUFFER
            LDA SECT
READ1:      OUT SECTP  ;SET SECTOR INTO 1771.
            MVI A,8CH  ;CODE FOR READ W/O HD LD.
READE:      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;
RDONE:      LXI H,00H  ;HEAD OF BUFFER
            LDA PATEN  ;TEST PATTERN
            MOV B,A   ;PATTERN TO B
            MVI D,00H  ;COUNTER FOR BYTES
COMPLP:     MOV A,M   ;GET DATA
            CMP B      ;COMPARE WITH TB
            JNZ ERRRET ;ERROR
            JNZ COMPLP ;DO 128 TIMES
            IN DSTAT  ;READ DISK STATUS.
            ANI 90H   ;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.
ERRRET:     LXI H,ER MSG  ;PRINT "READ ".
ERMSG:      CALL PMSG  ;PRINT ORIGIN MESSAGE.
COMMON ERROR PRINT OUT

ERMSG1: MOV A,D ;GET ERROR BITS.
ANI 80H ;IF BIT 7 IS HIGH,
LXI H,RRMSG ;"NOT READY".
CNI 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".
CNI PMSG
MOV A,D ;GET ERROR BITS.
ANI 1H ;IF BIT 1 IS HIGH.
LXI H,BSTMSG ;PRINT "BUSY".
CNZ PMSG
PERMSG: LXI ti,ERRHSG ;PRINT "ERROR."
CALL PMSG
MOV A,D ;MOVE FLAGS TO ACC
ANI 18H ;CRC OR RECORD NOT FOUND
JZ RETRY

TRKCHK: MVI A,DC4H ;READ ADDRESS
OUT DCOM IN WAIT IN ODATA STA BDTRK
IN WAIT IN CHKS2
JM LXI H,MSG5 ;READ ERROR MESSAGE
CALL PMSG
LXI H,MSG6 ;READINGS
CALL PMSG
MVI D,05H CALL BLNK ;SPACE OVER
LDA BTTRK ;DISK TRK
MOV D,A CALL BINHA ;PRINT TRK
MVI D,05H CALL BLNK ;SPACE OVER
CALL BINHA ;PRINT TRK
MVI D,13H CALL BLNK ;SECTOR
LDA SECT ;SECTOR NO.
MOV D,A CALL BINHA ;PRINT SECT NO.
RETRY: LDA ERRFLG ;SET FLAGS
ORA A ;GO TO READ CHECK ERROR PRINT
CNZ ERREPRNT ;CLEAR KEYBOARD
IN CDAIA ;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 ;CHECK FOR C
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
DATERR: STA BADBT ;SAVE BAD BYT
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 PATENMOV B,A ;TEST PATTERN IN B
LDA SECT ;LOAD SECTOR
WRITE: OUT SECTP ;SET THE SECTOR INTO 1771.
MVI A,0ACH ;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,WTMSG ;PRINT "WRITE ".
CALL PMSG
MOV A,D ;GET ERROR BITS.
ANI 40H ;LOOK AT BIT 6.
LXI H,WPMNG ;PRINT "PROTECT ".
CNI PMSG
MOV A,D ;GET ERROR BITS.
ANI 20H ;LOOK AT BIT 5.
LXI H,WPMSG ;PRINT "FAULT ".
CNI PMSG
JMP ERMNG1 ;DO COMMON MESSAGES.

; MOVE THE HEAD TO THE TRACK IN REGISTER A.
SEEK: OUT DDATA ;TRACK TO DATA REGISTER.
BUSB: 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 A ;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
NZ ;RETURN IF NO ERROR
LXI H,SRMSG ;PRINT "SEEK ".
JMP ERMNG ;DO COMMON ERR MESSAGES.

; PRINT THE MESSAGE AT HAL UNTIL A ZERO.
PHMSG: MOV A,M ;GET A CHARACTER.
ORA A ;IF IT'S ZERO,
NZ ;RETURN.
MOV C,A ;OTHERWISE,
CALL CNOT ;PRINT IT.
INX H ;INCREMENT HAL.
JMP PHMSG ;AND GET ANOTHER.
; CB IOS MESSAGES
REM EQU 000H ;MONITOR ENTRY
;
NRMSG: DB 'NOT READY',0
RNMSG: DB 'RECORD NOT FOUND',0
CRCMSG: DB 'CRC',0
LDMMSG: DB 'LOST DATA',0
BSYMSG: DB 'BUSY',0
WPMSC: DB 'PROTECT',0
WPSGC: DB 'FAULT',0
ERRMSG: DB 'ERROR.',0
RDMSG: DB ODH,OAH,'READ',0
WTMSG: DB ODH,OAH,'WRITE',0
SKMSG: DB ODH,OAH,'SEEK',0
HEMSG: DB ODH,OAH,'HOME',0
MNTMSG: DB ODH,OAH,'MOUNT',0
;
; PRINT 8 BIT WORD IN BINARY FORMAT
; INPUT: DATA IN REG A
;=================================================================================================================================

BIT: MOV B,A ; DATA
MVI C,30H ; 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
LP: CALL CONOT
DCR D
JNZ LP
RET
;
BINHA: MOV A,D
RAR
RAR
RAR
CALL BIN1
MOV C,A
CALL CONOT
MOV A,D
CALL BIN1
MOV C,A
CALL CONOT
RET
;
; O U T P U T S F O U R H E X D I G I T S I N A S C I I
; ENTER WITH DATA IN REG PAIR E AND D
;=================================================================================================================================

BINB: CALL BINHA
MOV D,E
CALL BINHA
RET

51
CONVERGES HEX TO ASCII

INPUT: 4 BITS HEX REG A
OUTPUT: 8 BIT ASCII REG A

INITIALIZE SIO PORTS

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

END
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 lites 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 'WRTTRK.COM'.
"DISSP2: ALL TRACK WRITE
"SELECT TRACK 8 THEN SENSE SWITCHES
"END
"READ
"JN EXE
"JMP STARTC
"ORG 0100H

ENTRYA: JMP STARTA
ENTRYB: LXI H,0B0H
JMP STARTA

STARTA: LXI H,MSG1B
CALL IN17A
JNZ STARTA

READ: CALL PMSG
CALL CONT
CPI 'T'
LXI H,MSG2A
JNZ READ
CALL HOME

STARTB: CRR A
JNZ STARTA
LXI H,STARTB
JPJH N

STARTC: IN OFF
CPI ??
JNC ERRA

SEEKA: CALL SEEK
MV! B,OFFH
MV! A,OFFH
OUT DCOM

WRITLP: IN WAIT
ORA A
JP WDONE
MOV A,B
OUT DDATA
JMP WRITLP

ERRA: MV! B,OFFH
CMA OUT OFFH
LXI D,01H
LXI H,000H
ERRLPB: DAD D
JNC ERRLPB
MOV B,A
IM OFFH
CPI ??
JNC ERRLP

DELAY: LXI H,0
DEL: DAD D
JNC DEL

DELAYA: LXI H,0
DELP: DAD D
JMP DELP

DELPA: DAD D
JNC DELPA
JMP STARTC"
MOVE DISK TO TRACK ZERO.

HOME: MOV A,DOMH ;CLEAR ANY PENDING COMMAND.
       CJT DOMH
       XRA A
       STA TRK ;STORE TRACK.
       HOM1: IN DSTAT ;READ DISK STATUS.
       INC PSH ;LOOK AT LSB.
       MOV A,3
       OUT DOMH ;ISSUE HOME COMMAND.
       IN WAIT ;WAIT FOR INTRQ.
       STA A
       JMP HERR ;ERROR IF ERR.
       IN DSTAT ;READ DISK STATUS.
       MOV A,3A
       STA 91H ;SAVE IN REGISTER D.
       MOV A,D
       JMP PMSHG ;DO COMMON ERROR MSGS.

HERR: LXI H,HERRMSG ;PRINT "HOME".
       MOV A,D
       ANI 91H
       HERR ;ERROR IF NOT ERR.
       JMP HERR ;RETURN IF NO ERROR.

PERMSG: LXI H,ERRMS
       CALL PMSH ;PRINT "ERROR".
       MOV A,D
       ANI 18H
       JMP TRKCHX ;MOVE FLAGS TO ACC
       JC TRKCHX ;IF REPER MESS.

TRKCHX: MOV A,DOMH
       CJT DOMH
       STA BDTRK ;TRACK ADDRESS
       CJT DMRK
       JMP CHKS2 ;READ ADDRESS
       STA BDTRK ;DUMP REST OF DATA
       CALL PMSH ;HEAD ERROR MESSAGE
       LXI H,MSG6
       CALL PMSH ;READINGS
       LXI H,MSG5
       CALL PMSH ;HEADINGS
       LXI H,BLNK
       CALL PMSH ;SPACE OVER
LDA BDTRK ;DISK IRK
MOV D,A CALL BINHA ;PRINT IRK
MVI D,15H CALL BLNK ;SPACE OVER
MOV D,A CALL BINHA ;PRINT IRK
MVI D,13H CALL BLNK
LDA SECT ;SECTOR
MOV D,A CALL BINHA ;PRINT SECT NO.
RETRY:
IN CDATA ;CLEAR KEYBOARD
ANI OB
JNZ CONT
LXI H,MSGB CALL PMSG ;REQUEST INPUT
CALL CONIN ;READ KEYS
GPI 'K' ;CHECK FOR R
JZ FIX
GPI '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 OFD ;LOOK AT THESE BITS.
MOV D,A ;SAVE STATUS BITS
PROCER:
LXI H,WTMSG ;PRINT "WRITE ".
CALL PMSG
MOV A,D ;GET ERROR BITS.
ANI 40H ;LOOK AT BIT 6.
LXI H,WPMSG ;PRINT "PROTECT ".
CALL PMSG
MOV A,D ;GET ERROR BITS.
ANI 20H ;LOOK AT BIT 5.
LXI H,WFMSG ;PRINT "FAULT ".
CALL PMSG
JMP ERRMSG ;DO COMMON MESSAGES.

; MOVE THE HEAD TO THE TRACK IN REGISTER A.
SEEK:
OUT DDATA ;TRACK TO DATA REGISTER.
PROC:
IN DSTAT ;READ DISK STATUS.
JC BUSY ;WAIT TILL NOT BUSY.
MVI A,12H ;SET FOR 10 MS STEP.
OUT DECOM ;ISSUE SEEK COMMAND.
IN WAIT ;WAIT FOR INTRO.
IN DSTAT ;READ STATUS.
ANI 9H ;LOOK AT BITS.
MOV D,A ;SAVE STATUS.
R1LXI H,SKMSG ;PRINT "SEEK ".
JMP ERRMSG ;DO COMMON ERR MESSAGES.

; PRINT THE MESSAGE AT H & L UNTIL A ZERO.
PMSE:
MOV A,M ;GET A CHARACTER.
ORA A ;IF ITS ZERO.
REZ ;RETURN.
MOV C,A ;OTHERWISE.
CALL CONOT ;PRINT IT.
INX H ;INCREMENT H & L.
JMP PMSG ;AND GET ANOTHER.
; CBOS MESSAGES
REN T EQU 0000H ;MONITOR ENTRY

RNMSG: DB 'NOT READY ',0
RNMSG: DB 'RECORD NOT FOUND ',0
CRCMSG: DB 'CRC ',0
LDMMSG: DB 'LOST DATA ',0
BSTMG: DB 'BUST ',0
WMP2: DB 'PROTECT ',0
WPMG: DB 'FAULT ',0
ERRMSG: DB 'ERROR ',0
RLMSG: DB 'CDM,CDH,READ ',0
WMG: DB 'CDM,CDH,WRITE ',0
SKM2G: DB 'CDM,CDH,SEEK ',0
MKMSG: DB 'CDM,CDH,MOUNT ',0

BLNK: MVI C,20H ;PRINT BLANKS, # IN REG. D
LP1: CALL CON01
DCR D
JNC LP1
RET

BINHA: MOV A,D
RAR
RAR
RAR
CALL BIN1
MOV C,A
CALL CON01
MOV A,D
CALL BIN1
MOV C,A
CALL CON01
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 OFH
ADI 30H
CPI 3AH
RE
ADI 07H
RET
INITIATE SIO PORTS

INITIATE: 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 lites 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'.

63
DISC TEST FULL TRACK READ
SELECT TRACK IN SENSE SWITCHES
DISC VERSION
BRIAN DONLAN
JUNE 79
ORG 0100H
ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC

ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC

ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC

ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC

ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC

ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC

ENTRYA: JMP STARTA
ENTRYB: LXI H,0800H
ECHL STARTA: LXI H,MSG1B
READT: CALL CONIN
    LXI H,MSG2A
    JNZ READT
    CALL HOME
STARTB: ORA A
    JNZ STARTA ;ERROR START OVER
    LXI H,STARTB ;SUBROUTINE RETURN
    PUSH H
STARTC: IN OFFH
    CPI 77 ;PREVENT TRACK OVER-DRIVE
    JNC ERRA
SEEKA: CALL SEEK
    MVI A,OESH
    OUT DCOM
    JMP RDLP
RDLPLX: IN DDATA
ERRA: MVI B,0F0H
ERRLP: MOV A,B
    CMP A
    OUT OFFH
    LXI D,01H
    LXI H,000H
ERRLPB: DAD D
    JNC ERRLP
    MOV B,A
    IN OFFH
    CPI 77
    JNC ERRLP
DELAY: LXI H,0
    LXI D,01H
Delp: DAD D
    JNC DELPA
DELAYA: LXI H,0
    LXI D,01H
DELP: DAD D
    JNC DELPA
DELPA: DAD D
    JMP STARTC
ROONE: IN DSTAT ;READ STATUS
ANI 5DH
MOV D,A
RZ
LXI H,ROMSG
JMP ERMSG

;:
MSG1B: DB ODH,OH,OA, "DISK TRACK READ ROUTINE"
DB ODH,OA, "LOAD SCRATCH DISK TYPE Y WHEN READY ",O
MSG2A: DB ODH,OA, "END OF PASS ",O
MSG3: DB ODH,OA, "DATA ERROR ON DISK  
CHECK ERROR COUNT IN HEX ",O
MSG4: DB ODH,OA, "TRACK NO. SECTOR NO. TEST  
BYTE LAST ERROR ",ODH
MSG5: DB ODH,OA, "HEAD POSITION ",O
MSG6: DB ODH,OA, "DISK TRACK CONTROLLER TRACK SECTOR 
EXECUTION STOPPED ",O
MSG7: DB ODH,OA, 

LPCT: DS 1 ;SPACE FOR LOOP COUNTER
BADBT: DS 1 ;SPACE FOR BAD BYTE
BADBK: DS 1 ;SPACE FOR DISK READ TRK WHEN ERR
MSG1: DB ODH,OA, 
"LOAD SCRATCH DISK TYPE Y WHEN READY ",O
MSG2: DB ODH,OA, "END OF PASS ",O
MSG3: DB ODH,OA, "DATA ERROR ON DISK  
CHECK ERROR COUNT IN HEX ",O
MSG4: DB ODH,OA, "TRACK NO. SECTOR NO. TEST  
BYTE LAST ERROR ",ODH
MSG5: DB ODH,OA, "HEAD POSITION ",O
MSG6: DB ODH,OA, "DISK TRACK CONTROLLER TRACK SECTOR 
EXECUTION STOPPED ",O
MSG7: DB ODH,OA, 

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.
WAIT EQU DISK+3 ;DISK WAIT PORT.
DCONT EQU DISK+4 ;DISK CONTROL PORT.

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

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

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
; CONOUT: MVI A,ODH ;IF IT'S A CR,
CMP C ;THEN NOP OUT
JZ CONOUT ;TO NULL ROUTINE.
ANI CKTR ;IF NOT READY,
JZ CONOUT1 ;READY WHEN HIGH.
MOV A,C ;GET CHARACTER.
OUT CDATA ;PRINT IT.
RET ;RETURN.

; CONUL: PUSH B ;SAVE B.
MVI B,CNULL ;GET NULL COUNT.
CONUL1: CALL CONOUT1 ;PRINT CR.
MVI C,O ;GET NULL CHAR.
DCW B ;DECREMENT COUNTER.
JNZ CONUL1 ;DO NEXT NULL.
POP B ;RESTORE B.
MOV A,C ;RESTORE A.
RET ;RETURN.

65
MOVE DISK TO TRACK ERR.

HOME: MVU A,DOM
    OUT DCOM
    XRA A
    STA TRK
    STORE TRACK
    RRC
    JC HOME
    WAIT FOR NOT BUSY.
    MVU A,3
    20 MS STEP RATE.
    IN
    WAIT
    ISSUE HOME COMMAND.
    IN
    WAIT
    STORE ANY?
    CLEAR?
    MVI A,DOH
    AND
    XRA A
    ;ZERO ACC
    STA IRK
    ;STORE TRACK HOME I: IN
    DSTAT ;REACH DISK STATUS.
    RRC
    ;LOOK AT LSB.
    JC HOME
    ;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 DQ.
    IN
    DSTAT
    ;READ DISK STATUS.
    ANI
    91H
    ;MASK NON-ERROR BITS.
    RZ
    ;RETURN IF NO ERROR.
    HERR: LXI H,HEMSG
    ;PRINT "HOME ".
    MOV A,D
    ;GET FLAGS.
    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 IS HIGH,
    LXI H,REMSG
    ;"NOT READY".
    CXZ PMSG
    MOV A,D
    ;GET ERROR BITS.
    ANI 10H
    ;IF BIT 4 IS HIGH,
    LXI H,REMSG
    ;PRINT "RECORD NOT FOUND"
    CXZ PMSG
    MOV A,D
    ;GET ERROR BITS.
    ANI 8H
    ;IF BIT 3 IS HIGH,
    LXI H,CRCMSG
    ;PRINT "CRC ERROR".
    CXZ PMSG
    MOV A,D
    ;GET ERROR BITS.
    ANI 4H
    ;IF BIT 2 IS HIGH,
    LXI H,LDMSG
    ;PRINT "LOST DATA".
    CXZ PMSG
    MOV A,D
    ;GET ERROR BITS.
    ANI 1H
    ;IF BIT 1 IS HIGH,
    LXI H,BSYMMSG
    ;PRINT "BUSY".
    CXZ PMSG
    PERMSG: LXI H,ERMSG
    CALL PMSG
    ;PRINT "ERROR.".
    MOV A,D
    ;MOVE FLAGS TO ACC
    ANI 18H
    ;CRC OR RECORD NOT FOUND
    JZ RETRY
    TRKCHK: MVU A,DCH
    OUT DCOM
    IN
    WAIT
    STA BDTRK
    ;READ ADDRESS
    IN
    DDATA
    ;TRACK ADDRESS
    CHKS2
    IN
    WAIT
    jm CHKS2
    ;DUMP REST OF DATA
    LXI H,MSG5
    CALL PMSG
    ;HEADINGS
    LXI H,MSG6
    CALL PMSG
    ;SPACE OVER
    CALL BLNK
    LDA BDTRK
    IDISK TRK

66
NOV
DA
CALL BINHA ; PRINT TRK
MVI D,0;15H
CALL BLNK ; SPACE OVER
MOV D,A
CALL BINHA ; PRINT TRK
MVI D,0;13H
CALL BLNK
LDA SECT ; SECTOR
MOV DA
CALL BINHA ; PRINT TRK
RETRY:
IN COFA ; CLEAR KEYBOARD
IN OFFH ; READ SENSE SWITCHES
ANI O80H ; 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
LXI H,MSGB
CALL PMSG
IN COFA ; CLEAR KEYBOARD
ANI OFFH ; LOOK AT THESE BITS.
NOV D,A ; SAVE STATUS BITS
PROCER: RZ ; RETURN IF NO ERR.
WERRO: LXI H,WTMSG ; PRINT "WRITE ".
CALL PMSG
MOV A,D ; GET ERROR BITS.
ANI 40H ; LOOK AT BIT 6.
LXI H,WPMSG ; PRINT "PROTECT ".
CNZ PMSG
MOV A,D ; GET ERROR BITS.
ANI 20H ; LOOK AT BIT 5.
LXI H,WFMSG ; PRINT "FAULT ".
CNZ PMSG
JMP ERRMSG1 ; DO COMMON ERR MESSAGES.
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 BIT 5.
MVI D,A ; SAVE STATUS.
RZ ; RETURN IF NO ERROR.
LXI H,SKMSG ; PRINT "SEEK ".
JMP ERRMSG ; DO COMMON ERR MESSAGES.
SUCCESS: 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 HAL.
JMP PMSG ; AND GET ANOTHER.
; CBIO5 MESSAGES
PENT 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
WPMMSG: DB 'PROTECT ',0
WFMMSG: DB 'FAULT ',0
ERRMSG: DB 'ERROR ',0
ROMSG: DB 0DH,0AH,'READ ',0
WIMSG: DB 0DH,0AH,'WRITE ',0
SKMSG: DB 0DH,0AH,'SEEK ',0
HMMSG: DB 0DH,0AH,'HOME ',0
MNTMSG: DB 0DH,0AH,'MOUNT ',0

;-----------------------------------------------
;
BLNK: MVI C,20H ;PRINT BLANKS, # IN REG. D
LP1: CALL CONOT1
DCR D
JNZ LP1
RET
;
BINHA: MOV A,D
PAR
PAR
PAR
PAR
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 9AH
NC
ADI 07H
RET
INITIALIZE 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 CONOUT
LF: MVI C,10 ; LF
CALL CONOUT1
MVI C,7FH
CALL CONOUT
CALL CONOUT1
RET
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 manual 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

HI Address
A 16:09
A16 A15 A14 A13 A12 A11 A10 A01 (10)

LO Address
A 08:01
A08 A07 A06 A05 A04 A03 A02 A01 (11)

HI Data
D 15:08
D15 D14 D13 D12 D11 D10 D09 D08 (12)

LO Data
D 07:00
D07 D06 D05 D04 D03 D02 D01 D00 (13)

Control 1
--- --- --- --- --- Ssn C1 Msn (14)

Control 2
Int --- --- --- --- --- Grt Req (15)

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

; DISC VERSION 24 MAY 79  B. DONLAN

ENTRY: LXI H, ENTRY
        PUSH H
        DI
        LXI H, MSG1
        CALL PMSG ; OPENING MESSAGE

; BEGINNING OF TEST

        MOV A, 01H
        MOV C, 10H
        OUT 10H
        MOV B, A
        IN 10H
        CMP B
        CNZ ERR
        CALL IF ERROR
        RLC
        JNC PORT10 ; TEST FOR A COMPLETE CYCLE

        MOV A, 01H
        MOV C, 11H
        OUT 11H
        MOV B, A
        IN 11H
        CMP B
        CNZ ERR
        CALL IF ERROR
        RLC
        JNC PORT11

        MOV A, 01H
        MOV C, 12H
        OUT 12H
        MOV B, A
        IN 12H
        CMP B
        CNZ ERR
        CALL IF ERROR
        RLC
        JNC PORT12

        MOV A, 01H
        MOV C, 13H
        OUT 13H
        MOV B, A
        IN 13H
        CMP B
        CNZ ERR
        CALL IF ERROR
        RLC
        JNC PORT13

        LXI HMSGR
        CALL PMSG ; FINISHED MESSAGE
        JMP RET ; RETURN TO MONITOR
&LH [*} ~N~ : ALL LEE AR A.<;N.AE;A, 37C 37'1, T R{R N - 30 A. SA < UAH OAH CAH, .NSTBjS PSP7 tESr',C } ; < - B AH ' AHH ? NO. 'vGc' D8 CAMl COH, .WST PATTERN o0 ISO DB CAH, AH, CZH, E.' DF S R:--N E CU- OCCOH ;YM)N:TOP ENTRY for brian donlar. 26 feo 79 AT EQU 3 ;CONSOLE STATUS POP. CM ECU 3 ;CONSOLE COMMAND PORT. CMP C ;THEN NOP OUT JZ CONUL ;IF NULL ROUTINE. CONOT: IN KSTAT ;READ CONSOLE STATUS. ANS CPTP ;IF NOT READY. JZ CONOT1 ;READY WHEN HIGH. MOV A,C ;GET CHARACTER. OUT CCDATA ;PRINT IT. RET ;RETURN. CONUL: PUSH B ;SAVE B&C. MVI A,C;NULL ;GET NULL COUNT. CONUL1: CALL CONOT1 ;PRINT CR. MVI C,0 ;GET NULL CHAR. DCF B ;DECREMENT COUNTER. JNZ CONUL1 ;DO NEXT NULL. POP B ;RESTORE B&C. MOV A,C ;RESTORE A. RET ;RETURN.

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

;STAT EQU 3 ;CONSOLE STATUS PORT.
;SSCM EQU 3 ;CONSOLE COMMAND PORT.
;SCM EQU 2 ;CONSOLE DATA PORT.
;KBR EQU 0000001B ;KEYBOARD READY BIT.
;PRT EQU 0000001B ;PRINT READY BIT.
;NULL EQU 1 ;CONSOLE NULL COUNT.

; WRITE A CHARACTER TO THE CONSOLE DEVICE.

CONOT: MVI A,CDH ; IF IT'S A CR, CMP C ; THEN NOP OUT JZ CONUL ; IF NULL ROUTINE. CONOT1: IN KSTAT ; READ CONSOLE STATUS. ANS CPTP ; IF NOT READY. JZ CONOT1 ; READY WHEN HIGH. MOV A,C ; GET CHARACTER. OUT CCDATA ; PRINT IT. RET ; RETURN.

CONUL: PUSH B ; SAVE B&C. MVI B,C;NULL ; GET NULL COUNT. CONUL1: CALL CONOT1 ; PRINT CR. MVI C,0 ; GET NULL CHAR. DCF B ; DECREMENT COUNTER. JNZ CONUL1 ; DO NEXT NULL. POP B ; RESTORE B&C. MOV A,C ; RESTORE A. RET ; RETURN.
; PRINT W-bit ASCII IN HEX FORM
; INPUT: DATA IN REG A

; ----------------------------------------
; MOV A, M ; SET DMAR
; CPA A ; IS IT A ZERO
; RZ
; MOV C, A ; OTHERWISE PRINT
; JVE H ; INC ADDRESS
; JMP PWSC

; ----------------------------------------
; PRINT W BIT ASCII IN BINARY FORMAT
; INPUT: DATA IN REG A

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

; ----------------------------------------
; OUTPUTS 2 HEX DIGITS IN ASCII
; FROM REG D

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

; ----------------------------------------
; CONVERTS HEX TO ASCII
; INPUT: W BIT HEX REG A
; OUTPUT: B BIT ASCII REG A

; ----------------------------------------
; BIN1: ANI OFH
; ADD 30H
; CPI 3AH
; RC
; ADI 07H
; RET
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.

76
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'.
<table>
<thead>
<tr>
<th>Unibus Address</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE00</td>
<td>AP-120B Formatter</td>
</tr>
<tr>
<td>FE20</td>
<td>AP-120B Word Count</td>
</tr>
<tr>
<td>FE21</td>
<td>AP-120B Host Memory Address</td>
</tr>
<tr>
<td>FE22</td>
<td>AP-120B DMA Control</td>
</tr>
<tr>
<td>FE23</td>
<td>AP-120B AP Memory Address</td>
</tr>
<tr>
<td>FE24</td>
<td>AP-120B Panel Switches</td>
</tr>
<tr>
<td>FE25</td>
<td>AP-120B Panel Functions</td>
</tr>
<tr>
<td>FE26</td>
<td>AP-120B Panel Lites</td>
</tr>
<tr>
<td>FE27</td>
<td>AP-120B Reset</td>
</tr>
<tr>
<td>FFE8</td>
<td>Filter Control Register</td>
</tr>
<tr>
<td>F800</td>
<td>Front Panel</td>
</tr>
<tr>
<td>F801</td>
<td>Front Panel</td>
</tr>
<tr>
<td>F802</td>
<td>Front Panel</td>
</tr>
<tr>
<td>0003</td>
<td>Data Acquisition Module</td>
</tr>
<tr>
<td>2102-21FF</td>
<td>Display Memory</td>
</tr>
</tbody>
</table>

Table 11.1
Unibus Addresses
UNIBUS COMMUNICATION TEST
DISC VERSION 24-MAY-79 B. DONLAN
ENTRY: EQU 0

ORG 100H
ENTRY3: LXI H,ENTRY3
PUSH H
LXI H,MSG5 ;OPENING MESSAGE
CALL PMSG
CALL BBIN ;GET HEX CHAR
PUSH D ;SAVE ADDRESS
LXI H,MSG6 ;REQUEST MODE
CALL PMSG
TPYN: CALL CONTIN
CPI '1'
JZ PUTIN ;JUMP IF INPUT MESSAGE
CPI 03H ;TEST IF CONTROL C
JZ RENT ;RETURN TO MONITOR
CPI '0'
JNZ QUEST

;OUTPUT MODE
PUTOUT: LXI H,MSG7 ;OUTPUT MESSAGE
CALL PMSG
CALL BBIN ;GET DIGITS TO OUTPUT
POP B ;RESTORE ADDRESS TO REG B & C
CALL DATA0 ;UNIBUS DRIVER
JMP DONE

PUTIN: LXI H,MSG9 ;INPUT MESSAGE
CALL PMSG
POP B ;RESTORE ADDRESS TO B & C
CALL DATA1 ;UNIBUS INPUT ROUTINE
CALL BBIN ;PRINT DATA FROM BUS
JMP DONE

DONE: LXI H,MSG10 ;PRINT END OF TEST
CALL PMSG
JMP ENTRY3

QUEST: LXI H,MSG7 ; ??
CALL PMSG
JMP TRYGN

MSG5: DB OAH,ODH,'UNIBUS COMMUNICATION TEST'
DB OAH,ODH,'ENTER UNIBUS ADDRESS ','0
MSG6: DB OAH,ODH,'INPUT (I), OUTPUT (O), EXIT (CONTROL C) '?,0
MSG7: DB OAH,ODH,' '?,0
MSG8: DB OAH,ODH,' ENTER DATA TO OUTPUT IN 4 HEX DIGITS ','0
MSG9: DB OAH,ODH,' DATA FROM BUS ','0
MSG10 DB OAH,ODH,'TRANSFER COMPLETE',0
; PROGRAM: INPUT/OUTPUT ROUTINES
; FOR 8080 SEQUENTIAL PROCESSION

; DEFINITIONS

; Pseudo direction
; INPUT: IN
; OUTPUT: OUT
; STATUS: STAT
; PORT: PRT
; COUNT: CNTR
; TALLY: CNTR
; MESSAGE: MSG
; MESSAGE ADDRESS: MSA
; MESSAGE ADDRESS: MSA

; PROGRAM

; READ A CHARACTER FROM CONSOLE.
LDA STAT ;READ CONSOLE STATUS.
ANI CRBR ;IF NOT READY,
JZ COMIN ;READY WHEN HIGH.
IN CDATA ;READ A CHARACTER.
OUT CDATA ANI 7FH ;MAKE MOST SIG. BIT = 0.
RET ;RETURN.

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
MOV A,ODH ;IF IT'S A CR,
CMP C ;THEN HOP OUT
JE COMUL ;TO NULL ROUTINE.
COMOT: IN STAT ;READ CONSOLE STATUS.
ANI CPTR ;IF NOT READY,
JZ COMOT ;READY WHEN HIGH.
MOV A,C ;GET CHARACTER.
OUT CDATA ;PRINT IT.
RET ;RETURN.

; CALL COMULT.
PUSH B ;SAVE B.
MOV B,CNULL ;GET NULL COUNT.
COMULT: CALL COMOT ;PRINT CR.
MOV C,0 ;GET NULL CHAR.
DEC B ;DECREMENT COUNTER.
JMI COMULT ;DO NEXT NULL.
POP B ;RESTORE B.
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
JZ ...
MOV C,A ;OTHERWISE PRINT CALL COMOT INX H ;INC ADDRESSS JMP PMSG ;
PRINT 3 BIT WORD IN BINARY FORMAT

INPUT: DATA IN REG A

BITS: MOV B,A; DATA
OVER: MOV A,80H; MASK
       MOV E,A
       ANA B
       JZ PRINT
       CALL C; JUMPFI ZERO
       CALL CONOT
       ANA B
       MOV A,E
       ZERO CARRY
       LOAD MASK
       INC RET OVER

PRINT BLANKS, # IN REG. D

PRINT BLANKS: MOV C,20H
LP1: CALL CONOT
      DCR D
      JNZ LP1
      RET

OUTPUTS 2 HEX DIGITS IN ASCII
FROM REG D

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

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

BINH: CALL BINHA
       MOV D,E
       CALL BINH
       RET

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

BIN1: ADD OFH
       ADI 30H
       CPI JAH
       RC
       ADI
       RET OTH
ABIN: CALL CONIN
CALL A-51
PUL
PUL
PUL
AND DFH
MOV D.A
CALL CONIN
CALL AHS1
AND 2FH
ORA 9
MOV D.A
CALL CONIN
CALL AHS1
PUL
PUL
PUL
AND DFH
MOV E,A
CALL CONIN
CALL AHS1
AND DFH
ORA E
MOV E,A
RET

;xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
DATA1: CALL GETBUS
   CALL DATI
   MOV A, &H10
   OUT 15H
   RET

DATA2: CALL GETBUS
   CALL DATO
   MOV A, 0
   OUT 15H
   RET

; ROUTINE TO INPUT A 16 BIT WORD FROM UNIBUS
; REG B = C<15:25>, REG C = C<0:15>
; DATA WILL BE CONTAINED IN REG D = C<15:03>, REG E = C<31:00>

DATI: MOV A, OFFH
   STA BIBOTI
   BZLP1: IN 14H
   ANI 0FH
   JNZ BUSY

   MOV A, B
   OUT 10H
   MOV A, C
   OUT 11H
   SUB A
   OUT 14H
   OUT 01H
   OUT 14H

SYNLP1: MOV A, OFFH
   STA SYNTI
   SYNLOOP: IN 14H
   OUT OFFH
   ANI 0FH
   JZ NOSYN1

   IN 12H
   MOV D,A
   IN 13H
   MOV E,A

   SUB A
   OUT 14H
   OUT 10H
   OUT 11H
   OUT 12H
   OUT 13H
   IN OFFH
   ANI 0B0H
   JNZ DATI
   RET

DATI: MOV A, OFFH
   STA BIBOTI
   BZLP1: IN 14H
   ANI 0FH
   JNZ BUSY

   MOV A, B
   OUT 10H
   MOV A, C
   OUT 11H
   SUB A
   OUT 14H
   OUT 01H
   OUT 14H

SYNLP1: MOV A, OFFH
   STA SYNTI
   SYNLOOP: IN 14H
   OUT OFFH
   ANI 0FH
   JZ NOSYN1

   IN 12H
   MOV D,A
   IN 13H
   MOV E,A

   SUB A
   OUT 14H
   OUT 10H
   OUT 11H
   OUT 12H
   OUT 13H
   IN OFFH
   ANI 0B0H
   JNZ DATI
   RET
null
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 in 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...
UNCLASSIFIED

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

2-2
2-30-80

END
DATE FILLED 1-81

DTIC

AFIT-CI-80-97
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
ENTRY: LXI H,FINIS
PUSH H
DI
CALL INITA
;SUBROUTINE ENTRY POINT
ENTRY: LXI H,MSG12
CALL PMSG
IN 10H
MOV D,A
IN 11H
MOV E,A
CALL BINB
; MVI D,8H
CALL BLNK
IN 12H
MOV D,A
IN 13H
MOV E,A
CALL BINB
MVI D,8H
CALL BLNK
; IN 14H
ANI 04H
JZ MOSIS
MVI C,','1'
JMP OUTSIS
MOSIS: MVI C,','0'
OUTSIS: CALL CONOT
MVI D,09H
CALL BLNK
; IN 15H
ANI 07H
JZ NOBUS
MVI C,','1'
JMP OUTBUS
NOBUS: MVI C,','0'
OUTBUS: CALL CONOT
MVI D,08H
CALL BLNK
; IN 14H
ANI 00H
JZ NOMSIS
MVI C,','1'
JMP OUTMSIS
NOMSIS: MVI C,','0'
OUTMSIS: CALL CONOT
RET
FINIS: JMP FINIS
MSG12: DB OA0,OA0,ODH,'UNIBUS SNAP-SHOT'
DB OA0,OA0,'ADDRESS'
DB OA0,ODH,'DATA'
DB OA0,ODH,'838H'
; for brian donlan 26 Feb 79
CSTAT EQU 3
CCOM EQU 3
CDATA EQU 3
CKBR EQU 00000001B
CPTR EQU 00000001B
CNUL EQU 1
; CHECK CONSOLE INPUT STATUS.

CONST: IN CSTAT ;READ CONSOLE STATUS.
ANI CKBK ;LOOK AT KB READY BIT.
MVI A,0 ;SET A=A 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 CKBK ;IF NOT READY,
JZ CONIN ;READY WHEN HIGH.
IN CDATA ;READ A CHARACTER.
OUT CDATA
ANI TFH ;MAKE MOST SIG. BIT = 0.
RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
CONOT: MVI A,ODH ;IF IT'S A CR,
CMP C ;THEN HOP OUT
JZ CONOT ;READY WHEN HIGH.
CONOTO: IN CSTAT ;READ CONSOLE STATUS.
ANI CKBK ;IF NOT READY,
MVI A,C ;GET CHARACTER.
OUT CDATA ;PRINT IT.
RET ;RETURN.

CONUL: PUSH B ;SAVE B&C.
MVI B,CNULL ;GET NULL COUNT.
CONULI: CALL CONOT ;PRINT CR.
MVI C,0 ;GET NULL COUNT.
DCR B ;DECREMENT COUNTER.
JNZ CONULI ;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
MOV B,A ; STORE MASK
MVI C,30H ; AND WITH MASK
RZ PRINT ; JUMP IF ZERO
MVI A,C ;
PRINT:  CALL CONOT
       B ; ZERO CARRY
       A, E ; LOAD MASK
       OVER

BLNK:  MVI C, 20H ; PRINT BLANKS, # IN REG. D

LP31:  CALL CONOT
       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 4 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
       AC
       ADI 0FH
       RET

Inputs 4 DIGITS FROM CONSOLE
RETURN: 4 HEX DIGITS IN REG E-D

BINW:  CALL COMIN
       CALL ANS1
       RAL
       RAL

RIN1:  ANI OFH
       ADI 30H
       CPI 3AH
       RC
       ADI 07H
       RET

Inputs 4 DIGITS FROM CONSOLE
RETURN: 4 HEX DIGITS IN REG E-D

BINW:  CALL COMIN
       CALL ANS1
       RAL
       RAL

90
VERT ASCII TO HEX
INPUT: 8 BIT ASCII REG A
OUTPUT: 4 BIT HEX REG A

;------------------------------------------
; ANS1:  NOP
; CPI OAH
; RC
; SUI 07H
; RET

;------------------------------------------
; INITIATE SIO PORTS
;------------------------------------------
; INITIA: MVI A,OAAH ;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 CONOUT
; LF: MVI C,10 ;LF
; CALL CONOUT
; CALL CONOUT
; CALL CONOUT
; RET
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.


5. Ng, Howard, Disk System for a Microcomputer Controlled Spectra Analysis System, Master's Project, RPI, 1978


APPENDIX A

MEMORY BOARD DIAGRAMS
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 TM 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.
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

ORG 00000H

0000 218000 :TEST BYTE
0003 F9 SPHL
0004 210AC0 ENTRY: LXI H,080H
0007 E5 PUSH H
0008 3E00 MVI A,00 ;ZERO ACC
000C 320A00 STA CODE
000F C07C CALL INITA ;RESET I/O PORT
0012 2120C1 LXI H,MSG1
0015 2112C1 LXI H,MSG2
0018 C01C CALL PMSG
001B C0FF CALL BBIN
0020 220A00 SHLD START
0023 2199C1 LXI H,MSG3
0026 C02C CALL PMSG
0029 3E02 CALL BBIN
002C 214900 SHLD ENADR
002F 21B9C1 LXI H,MSG8
0032 C036 CALL PMSG
0035 DB02 ;RESET 10 FLAG
0038 C03D MVI CWO ;LOAD TEST BYTE
003B 3E02 MTEST: MVI A,02 ;LOAD TEST BYTE
003E 320800 STA PART
0041 C045 3E02 MILOP: CALL STUFF ;STUFF MAJOR ALL OVER
0044 3E02 MVI A,02 ;SET TWO AS MINOR
0047 C049 CALL STUFF ;STUFF MINOR
004C 3E02 MVI A,02 ; SET 2 AGAIN
004F C04D CALL CHECK ;NOW CHECK ALL LOC
0052 3D36 DCR A
0055 320800 STA PART ;STORE NEW PART
0058 C055 CPI 00 ;FINISH THIS PASS ?
005B 3E01 CONT: MVI A,01 ;NO CONTINUE
005E 3E00 CALL STUFF ;STUFF MINOR SERF
0061 2F MOV A,C ;LOAD MAJOR BYTE
0064 C062 4F MOV C.A ;COMPLIMENT MAJOR BYTE
0067 3E04 MOV C.A ;SAVE NEW BYTE
006A C065 CALL CHECK ;ZERO OTHER TEST BYTE
006G C067 C342C0 JMP MILOP

RECYCLE:

C06A 79 MOV A,C ;SAVE INVERT TO TEMP
C06E 2162C1 STA PART ;SAVE INVERT TO TEMP
C071 C061 CALL CODE ;CHAR CODE
C074 CA0400 LDA PMSG
C077 FE03 CPI 03H ;CONTROL C
C079 CA1100 JZ RENT ;RETURN TO MONITRO
C07C 87 ORA A ;SET FLAGS
C07F 87 JNZ ENTRY ;START OVER
C082 3A0800 ANA A ;CLEAR CARRY
C085 3A0800 LDA PART ;RECOVER TEST BYTE
C088 C33BC0 JZ BEGIN
C08B 2F MOV C.A ;NEW TEST BYTE
C08E 4F JMP MTEST ;ANOTHER PASS
FO21

RENT: EQJ OF021H
MONITOR ENRy

FO09

PART: EQJ 06H
LOC OF END ADDR

FoOA

CODE: EQJ 0AH
LOC FOR CODE

CODE CDECO
STUFF: CALL STASTO ;LOAD START AND END ADDR

CODE CD06C1
DOIT: MOV M, C ;STUFF MAJOR ALL OVER

CODE C391CO
CALL MILOX ;SEE IF ALL MEM DONE
JMP DOIT ;NO KEEP ON STUFFING

CODE CD06C0
STUFM: CALL STASTO ;LOAD ADDR AGAIN

CODE CE4F
MOV B, A ;MINOR COUNTER

CODE C2E6C0
CPI 00 ;MINOR WORD STUFF

CODE C0A1 79
MINOR: MOV A, C ;MAJOR TEST BYTE

CODE C0A2 2F
CMA ;MINOR IS COMPLEMENT OF MAJOR

CODE C0A5 77
MOV M, A ;STUFF MINOR BYTE IN MEM

CODE C0A6 CD06C1
MIL: CALL MILOX ;INC & CCR IF DONE

CODE C0A9 05
DCR B ;DEC MINOR COUNTER

CODE C0AA C386C0
JMP MINOR ;YES

CODE C0B0 CDECO
CHECK: CALL STASTO ;LOAD ADDR AGAIN

CODE C0B4 F600
CPI 00 ;LOAD MINOR COUNT

CODE C0B9 79
MINR: MOV A, C ;NO GO TO MAJOR

CODE C0BA CF
MOV M, A ;MAJOR COUNT

CODE C0BC 0630
CMP M ;READ AND COMPARE MEM LOC

CODE C0BE C3C3C0
MAJR: MOV A, C ;LOAD MAJOR TEST BYTE

CODE C0C2 0630
CMP M ;READ AND COMPARE MEM WITH MAJOR

CODE C0C3 C5
CKEND: PUSH B ;SAVE COUNTER AND MAJOR

CODE C0C7 C1
CPI 00 ;GO TO ERR TO PRINT IF ERROR

CODE C0C8 D603
IN CSTAT ;CHECK KEYBOARD

CODE C0CC C400
AMO C2H

CODE C0CF CB02
JZ FIN

CODE C0D1 320000
FIN: CALL MILOX

CODE C0D4 CD06C1
FIN: CALL CODE

CODE C0D7 CD06C1
FIN: CALL MILOX

CODE C0DB C2C1C0
JMP MINR ;COUNT ZERO DO MINOR

CODE JA30D0
STASTO: LMLD ENADR ;LOAD END ADDR

CODE 32A000
LMLD START ;LOAD START
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Mnemonic</th>
<th>Op Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COE6 D5</td>
<td>ERR:</td>
<td>PUSH</td>
<td>D</td>
<td>;SAVE END ADDR</td>
</tr>
<tr>
<td>COE7 F5</td>
<td></td>
<td>PUSH</td>
<td>FSM</td>
<td></td>
</tr>
<tr>
<td>COE8 CD75C2</td>
<td>CALL</td>
<td>CRIF</td>
<td></td>
<td>;OUTPUT BAD ADDR</td>
</tr>
<tr>
<td>COE9 54</td>
<td>MOV</td>
<td>D,H</td>
<td></td>
<td>;SPACE COUNT</td>
</tr>
<tr>
<td>COEC 5D</td>
<td>MOV</td>
<td>E,L</td>
<td></td>
<td>;SPACE OVER 8</td>
</tr>
<tr>
<td>COED C01AC2</td>
<td>CALL</td>
<td>BINK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COEF 1608</td>
<td>MVI</td>
<td>D,08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COF1 F1</td>
<td>POP</td>
<td>FSM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COF2 CD8C1</td>
<td>CALL</td>
<td>BLNK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COF3 F7</td>
<td>MOV</td>
<td>B,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COF4 78</td>
<td>CALL</td>
<td>BITS</td>
<td>;PRINT TEST BYTE</td>
<td></td>
</tr>
<tr>
<td>COF5 D1</td>
<td>POP</td>
<td>D</td>
<td>;RESTORE END ADDR</td>
<td></td>
</tr>
<tr>
<td>COF6 CD1AC2</td>
<td>CALL</td>
<td>BINS</td>
<td>;OUTPUT BAD</td>
<td></td>
</tr>
<tr>
<td>COF7 DB188</td>
<td>MOV</td>
<td>A,M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COF8 CD1AC8</td>
<td>CALL</td>
<td>BITS</td>
<td>;PRINT BAD BYTE</td>
<td></td>
</tr>
<tr>
<td>COF9 F1</td>
<td>POP</td>
<td>PSW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COFA 160A</td>
<td>MVI</td>
<td>O,OAH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COFC DB188</td>
<td>CALL</td>
<td>BLNK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COFD 7E</td>
<td>MOV</td>
<td>A,B</td>
<td>;MOVE TEST BYTE BACK</td>
<td></td>
</tr>
<tr>
<td>COFE CD1AC8</td>
<td>CALL</td>
<td>BITS</td>
<td>;PRINT TEST BYTE</td>
<td></td>
</tr>
<tr>
<td>COFF 7E</td>
<td>POP</td>
<td>D</td>
<td>;RESTORE END ADDR</td>
<td></td>
</tr>
<tr>
<td>C010 F5</td>
<td>HIL:</td>
<td>PUSH</td>
<td>A</td>
<td>;SAVE ACC</td>
</tr>
<tr>
<td>C011 23</td>
<td>INX</td>
<td>H</td>
<td>;INC CURRENT ADDR</td>
<td></td>
</tr>
<tr>
<td>C012 7C</td>
<td>MOV</td>
<td>A,H</td>
<td>;LOAD HIGHER ORDER ADDR</td>
<td></td>
</tr>
<tr>
<td>C013 22C1</td>
<td>CMP</td>
<td>D</td>
<td>;COMPARE WITH END</td>
<td></td>
</tr>
<tr>
<td>C014 7D</td>
<td>MOV</td>
<td>A,L</td>
<td>;LOAD LOW ORDER</td>
<td></td>
</tr>
<tr>
<td>C015 22C1</td>
<td>CMP</td>
<td>E</td>
<td>;COMPARE LOW ORDERS</td>
<td></td>
</tr>
<tr>
<td>C016 F1</td>
<td>POP</td>
<td>A</td>
<td>;MATCH END</td>
<td></td>
</tr>
<tr>
<td>C017 33</td>
<td>INX</td>
<td>SP</td>
<td>;FAKE RETURN ONE LEVEL OUT</td>
<td></td>
</tr>
<tr>
<td>C018 33</td>
<td>INX</td>
<td>SP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C019 F1</td>
<td>RET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C01A C3F</td>
<td>RET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C01B F7</td>
<td>DIFF</td>
<td></td>
<td>;CONTINUE STUFFING</td>
<td></td>
</tr>
<tr>
<td>C01C C9</td>
<td>RET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C01D 3F</td>
<td>PROB</td>
<td></td>
<td>;forgotten</td>
<td></td>
</tr>
<tr>
<td>C01E CD8C1</td>
<td>CALL</td>
<td>CONOT</td>
<td>;PRINT ?</td>
<td></td>
</tr>
<tr>
<td>C01F C304C0</td>
<td>JMP</td>
<td>ENTRY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C020 0044D44DMS</td>
<td>DB</td>
<td>ODM,OAH,'MEMORY TEST',O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C021 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,'ENTER START ADDRESS',O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C022 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,'ENTER STOP ADDRESS',O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C023 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,'END OF PASS',O</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C024 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,CONSOLE STATUS,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C025 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,CONSOLE COMMAND PORT,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C026 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,CONSOLE DATA PORT,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C027 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,KEYBOARD READY BIT,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C028 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,PRINT READY BIT,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C029 00455E5G3</td>
<td>DB</td>
<td>ODM,OAH,CONSOLE NULL COUNT,0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C02A 0003</td>
<td>CSTAT</td>
<td>EQU</td>
<td>3</td>
<td>;CONSOLE STATUS PORT</td>
</tr>
<tr>
<td>C02B 0003</td>
<td>CCOM</td>
<td>EQU</td>
<td>3</td>
<td>;CONSOLE COMMAND PORT</td>
</tr>
<tr>
<td>C02C 0003</td>
<td>CDATA</td>
<td>EQU</td>
<td>2</td>
<td>;CONSOLE DATA PORT</td>
</tr>
<tr>
<td>C02D 0002</td>
<td>CKBR</td>
<td>EQU</td>
<td>000000010B</td>
<td>;KEYBOARD READY BIT</td>
</tr>
<tr>
<td>C02E 0001</td>
<td>CPR</td>
<td>EQU</td>
<td>00000001B</td>
<td>;PRINT READY BIT</td>
</tr>
<tr>
<td>C02F 0001</td>
<td>CNUL</td>
<td>EQU</td>
<td>1</td>
<td>;CONSOLE NULL COUNT</td>
</tr>
</tbody>
</table>

; CHECK CONSOLE INPUT STATUS.

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Mnemonic</th>
<th>Op Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C030</td>
<td>DB</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C031</td>
<td>CSTAT</td>
<td>IN</td>
<td></td>
<td>;READ CONSOLE STATUS</td>
</tr>
<tr>
<td>C032</td>
<td>CSTAT</td>
<td>EQU</td>
<td></td>
<td>;LOCK AT KB READY BIT</td>
</tr>
<tr>
<td>C033</td>
<td>CSTAT</td>
<td>EQU</td>
<td></td>
<td>;SET A&amp;D FOR RETURN</td>
</tr>
<tr>
<td>C034</td>
<td>CSTAT</td>
<td>EQU</td>
<td></td>
<td>;NOT READY WHEN ZERO</td>
</tr>
<tr>
<td>C035</td>
<td>CSTAT</td>
<td>RET</td>
<td></td>
<td>;RETURN FROM CSTAT</td>
</tr>
</tbody>
</table>

108
; READ A CHARACTER FROM CONSOLE.
C180 DB03 CONIN: IF CSTAT ;READ CONSOLE STATUS.
C182 E602 A.I CRBR ;IF NOT READY.
C184 CA0C1 JZ CONIN ;READY WHEN HIGH.
C187 DB02 IN CDATA ;READ A CHARACTER.
C189 D302 OUT CDATA
C18B E07F ANI ?FH ;MAKE MOST SIG. BIT = 0.
C18D C9 RET

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
C18E 3E00 CONOUT: MVI A,ODH ;IF IT'S A CR.
C190 89 CMP C ;THEN HOP OUT.
C191 CA9C1 JZ CONOUT ;TO NULL ROUTINE.
C194 DB03 IN CSTAT ;READ CONSOLE STATUS.
C196 E601 ANI CPTB ;IF NOT READY.
C198 79 MOV A,C ;GET CHARACTER.
C19A D302 OUT CDATA ;PRINT IT.
C19C C9 RET ;RETURN.

C19F C5 CONUL: PUSH B ;SAVE B & C.
C1A0 0601 HVT B,CMULL ;GET NULL COUNT.
C1A5 4F POP 7 ;RESTORE A.
C1A6 C9 RET ;RETURN.

; PRINT MESSAGE UNTIL ZERO
; MESSAGE ADDRESS REG H & L
C1AE 7E PMSG: MOV A,M ;GET CHAR
C1AF 87 DRA A ;IS IT A ZERO
C1B0 C8 RZ
C1B1 4F MOV C,A ;OTHERWISE PRINT
C1B2 CD9C1 CALL CONOUT ;PRINT CR.
C1B5 05 DCR B ;DECREMENT COUNTER.
C1B6 C2A2C1 JNZ CONOUTI ;DO NEXT NULL.
C1B8 C1 POP B ;RESTORE B.
C1B9 C9 RET ;RETURN.

; Print a bit word in binary format
; INPUT: DATA IN REG A
C1EB 47 BITS: MOV B,A ;DATA
C1EC 36B0 MVI A,BOH ;MASK
C1ED 0E30 MVI C,30H
C1EE 4F MOV E,A ;STORE MASK
C1EF 40 MOV A,B ;AND WITH MASK
C1F0 CAF1C1 JZ PRINT ;JUMP IF ZERO
C1F1 0E31 MVI C,31H
C1F2 CD9C1 PRINT: CALL CONOUT ;PRINT CR.
C1F4 40 MOV A,B ;ZERO CARRY
C1F5 7B MOV A,E ;LOAD MASK
C1F6 1F XRA
C1F7 23 CALL PRINT ;PRINT CR.
C1FA C9 RET
CIFB 020
BLRN: MVI C,20H ;PRINT BLANKS, 0 IN REG. D
C1FD 094C1 LP19: CALL CONOT
C200 15 DCR D
C201 C2FD1 JMI LP19
C204 C9 RET

C205 7A BINHA: MOV A,D
C206 1F RAR
C207 1F RAR
C208 1F RAR
C209 1F RAR
C20A CD2CC2 CALL BIN1
C20D 6F MOV C,A
C20E CD8EC1 CALL CONOT
C211 7A MOV A,D
C212 CD2CC2 CALL BIN1
C215 6F MOV C,A
C216 CD8EC1 CALL CONOT
C219 C9 RET

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

C21A CD05C2 BINB: CALL BINHA
C21D 93 MOV D,E
C21E CD05C2 CALL BINHA
C221 C9 RET

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

C222 E60F BIN1: ANI OFH
C224 CD30 ADI 30H
C226 FE3A CPI 3AH
C228 DB RC
C229 C807 ADI 0TH
C22B C9 RET

; INPUTS 4 DIGITS FROM CONSOLE
; RETURN: 4 HEX DIGITS IN REG E-D

C22C CD80C1 BBIN: CALL CONIN
C22F CDSBC2 CALL ANS1
C232 17 RAL
C233 17 RAL
C234 17 RAL
C235 17 RAL
C236 E6FO ANI OFTH
C238 57 MOV D,A
C239 CD80C1 CALL CONIN
C23C CD8BC2 CALL ANS1
C23F E6OF ANI OFH
C241 B2 ORA D
C242 57 MOV D,A
C243 CD80C1 CALL CONIN
C246 CD8BC2 CALL ANS1
C249 17 RAL
C24A 17 RAL
C24B 17 RAL

110
CONVERT ASCII TO HEX

INPUT: 8 BIT ASCII REG A
OUTPUT: 4 BIT HEX REG A

INIT: MVI A, OAAH ;GET DUMMY MODE WORD

OUT CSTAT ;OUTPUT IT

MVI A, 4OH ;GET RESET BIT

OUT CSTAT ;RESET 10 BOARD

MVI A, OCH ;GET REAL MODE WORD

OUT CSTAT ;SET THE MODE FOR REAL

MVI A, 37H ;GET THE COMMAND

OUT CSTAT ;OUTPUT IT

MVI C, 13 ;CR

MVI C, 10 ;LF

MVI C, 7FH

MVI C, OFH

MVI E, A

MOV E, A

CALL ANSI

CALL AHSI

ANI OFH

ORA E

MOV E, A

CALL CONIN

CALL CONOT

CALL CONOT1

CALL CONOT
MINI-MEMORY TEST
FROM VERSION FOR 0 TO 100H

BRIAN J. DONLAN

C290
C299
C296
C297
C299
C29A
C298
C294
C29F
C2A0
C2A3
C244
C2A7
C2A8
C2AA
C2AD
C2AF
C2B1
C2B4
C2B6
C2B9
C2BA
C2BC
C2BF
C2C2
C2C4
C2CD
C2DA
C2DB
C2DF
C2ED
C2ED
C2EF
C2ED
C2EF
C2EG
C2EF
C2F0
C2F2

ORG OC290H
MVI A, OFEH
OUT OFFH
LXI H, 000H
MOV M, A
LXI H, OFFOH
MOV B, M
JNZ ERR1
JMP IF ERROR
JMP IF ERROR
JMP IF ERROR

OUTPUT PHASE I LITES
START ADDRESS
STORE TEST PATTERN IN MEM.
READ BACK TO B
COMPARE FOR OK
NEW TEST PATTERN
STOP ADDRESS
ADD TWO'S COMPLEMENT

PHASE II
PHASE II LITES
LOW ADDRESS TO MEM
STOP ADDRESS
READ MEMORY
COMPARE
JUMP IF ERROR

PHASE III
PHASE THREE LITES
STORE HIGH ADDRESS IN ALL MEM
READ MEMORY
COMPARE

C290 F3
C291 3EFE
C292 0FF
C295 210000
C298 AF
C299 77
C29A 46
C29B 88
C29C C2FD
C29F 3C
C2A0 C29C2
C2A3 23
C244 1100FF
C2A7 EB
C2A8 EB
C2AA D298C2
C2AD 3EFD
C2AF 3EFF
C2B1 210000
C2B4 74
C2B6 1100FF
C2B9 EB
C2BA 19
C2BB EB
C2BC D284C2
C2BF 210000
C2C2 7E
C2C4 C223C3
C2CD 23
C2CB 1100FF
C2CB EB
C2CC 19
C2CD EB
C2CE D2C2C2
C2D1 3EFC
C2D3 0FF
C2D5 210000
C2D8 75
C2DA 1100FF
C2DE 19
C2DF EB
C2ED D2D8C2
C2E3 210000
C2E5 C
C2E7 95
C2EF C2F3
C2ED 23
C2EF 1100FF
C2EF EB
C2F0 19
C2F2 D2E6C2

OC290H
MVI A, OFEH
OUT OFFH
LXI H, 000H
MOV M, A
LXI H, OFFOH
MOV B, M
JNZ ERR1
JMP IF ERROR
JMP IF ERROR
JMP IF ERROR

112
; ALL PHASE COMPLETE
C2F5 3EFF MVI A,OFFH
C2F7 2190C2 LXI H,ENTER
C2FA 33BC3 JMP LITES ;GO TO LITES PROG

; PHASE I ERROR
C2FD EB ERR1: XCHG
C2FE 4F MOV C,A ;SAVE BAD DATA
C2FF 2107C3 LXI H,COMERR ;RETURN
C302 3EF1 MVI A,OF1H ;PHASE I ERROR LITES
C304 C33BC3 JMP LITES

; COMMON ERROR OUTPUT ROUTINE
C307 7A COMERR: NOV A,E ;HIGH ADDRESS
C308 2107C3 LXI H,LOADD ;RETURN
C30B 3EF2 HVI k,OF2H ;PHASE II ERROR TO LITES
C30C 2107C3 LXI H,COMERR ;RETURN
C30E 78 LOADD: NOV A,E ;LOW ADDRESS TO LITES
C310 2190C2 LXI H,ENTER ;START OVER
C312 C33BC3 JMP LITES

; PHASE II ERROR
C315 7B TPAT: NOV A,C ;TEST PATTERN TO LITES
C317 2118FC LXI H,ACTDAT ;RETURN
C319 C33BC3 JMP LITES
C31B 7B ACTDAT: NOV A,B ;ACTUAL DATA TO LITES
C31D 2190C2 LXI H,ENTER ;START OVER
C320 C33BC3 JMP LITES

; LITES ROUTINE ENTER WITH RETURN IN REG H&L
C323 EB ERR2: XCHG ;SAVE BAD ADDRESS
C324 82 ADD D
C325 47 MOV B,A
C326 4A MOV C,D
C327 3EF2 MVI A,OF2H ;PHASE II ERROR TO LITES
C329 2107C3 LXI H,COMERR ;RETURN
C32C C33BC3 JMP LITES

; PHASE III ERROR
C32F EB ERR3: XCHG ;SAVE BAD ADDRESS
C330 83 ADD E
C331 47 MOV B,A
C332 4B MOV C,E
C333 3EF3 MVI A,OF3H ;PHASE II ERROR TO LITES
C335 2107C3 LXI H,COMERR ;RETURN
C338 C33BC3 JMP LITES

; LITES ROUTINE ENTER WITH RETURN IN REG H&L
C33B 2F LITES: CMA ;DATA FOR LITES IN A
C33C 3EFF OUT OFFH ;OUTPUT LITES
C33E F9 SPHL OFFH ;SAVE RETURN IN SP
C33F 2BFF IN OFFH ;READ SENSE SWITCHES
C341 67 MOV A,H ;SAVE IN H
C342 2BFF INH OFFH ;READ SWITCHES
C344 EC ZR A H ;SEE IF THEY CHANGED
C345 CA42C3 ZR LZ LP7 ;DELAY LOOP
C348 2BFF LP7: IN OFFH ;READ SENSE SWITCHES
C34A 23 INX H ;SEE IF THEY CHANGED
C34C AF ZR A
C34E B4 ZRA H
C34F 2BFF LP8: INX H ;DELAY LOOP
C351 210000 LXI H,OFCH ;ZERO H
C354 39 DAD SP ;MOVE RETURN BACK TO H & L
C355 E9 PCHL ;RETURN

113
DISK TEST FOR TARBEY DISK CONTROLLER

BRIAN J. DONLAN
18 MAR 79

PROM VERSION

C800 217000 ENTRY: LXI H,070H ;SET STACK POINTER
C800 9F3 SPHL
C800 860 ENTRY: LXI H,0TH;INITI
C800 21B0CH CALL PMOS
C800 860 ENTRY: CALL PMOS
C800 958H CALL CONIN
C817 7F69 CPT 'Y' ;CHECK KEYBOARD
C819 025CB JMNI ENTRY ;CHECK IF Y
C81A 6D3CD CALL CRLF ;START OVER
C81B 3E20 CD63CD CALL CRLF
18
C822 30B800 XRA A ;ZERO ACC
C82A 232800 STA ERFAG ;ZERO ERROR FLAG
C82C 2232800 CALL HOME ;HOME DRIVE TO TRK 0
C830 3E20CD LOOP4: XRA A ;ZERO INNER TRK
C838 320600 STA OUTER ;OUTER TRK
C83A 3C26 CALL PAT ;GET PATTERN
C83C 3C32CB CALL INWRT ;WRITE INNER TRK
C83E 3CDE7C8 CALL CONIN ;CHECK KEYBOARD
C840 3CDBE0 CALL SEEK ;MOVE BACK AND CHECK TRK 00
C842 3C0500 LOOP5: IN C3STAT
C844 3C5E02 AKI 02H ;KEYBOARD READY
C84C 3C85CB CALL CONIN ;READ KEYS
C84E 3C82CB CALL PMOS ;CONTROL C
C850 3C80CB CALL HOME ;RETURN TO MONITOR
C852 3C8400 LDA LPCNT ;SET UP TO DO PAIRS
C854 3C82CB CALL CONIN ;START PAIRS WITH TRK 01
C856 3C86CB CALL CONIN ;START OVER AGAIN
C858 3C88CB CALL CONIN
C85A 3C84CB CALL CONIN
C85C 3C80CB CALL CONIN
C85E 3C82CB CALL CONIN
C860 3C84CB CALL CONIN
C862 3C86CB CALL CONIN
C864 3C88CB CALL CONIN
C866 3C80CB CALL CONIN
C868 3C82CB CALL CONIN
C86A 3C84CB CALL CONIN
C86C 3C86CB CALL CONIN
C86E 3C88CB CALL CONIN
C870 3C80CB CALL CONIN
C872 3C82CB CALL CONIN
C874 3C84CB CALL CONIN
C876 3C86CB CALL CONIN
C878 3C88CB CALL CONIN
C87A 3C80CB CALL CONIN
C87C 3C82CB CALL CONIN
C87E 3C84CB CALL CONIN
114
C917 3A0600 OUTRD: LDA OUTER ;OUTER TRK NO.
C91A 320B00 STA TRK
C91D C3EDC8 JMP BOTH2
C920 21BC9 ERRPRN: LXI H,MSG3 ;ERROR MESSAGE
C923 CD8CC CALL PMSG
C926 3A0B00 LDA ERRFLG ;ERROR COUNT
C929 57 MOV D,A
C92D 21F3C9 LXI H,MSG4 ;HEADINGS
C930 CD8CC CALL PMSG
C933 1603 MVI D,03 ;SPACE OVER
C935 CD21CD CALL BLNK
C938 3A0B00 LDA TRK ;TRACK NO.
C93B 57 MOV D,A
C93F CD2BCD CALL BINHA ;PRINT TRACK NO.
C941 1610 MVI D,16 ;SPACE OVER
C944 CD21CD CALL BLNK
C947 3A0C00 LDA SECT ;SECTOR NO.
C94A 57 MOV D,A
C94D CD2BCD CALL BINHA ;PRINT SECTOR NO.
C950 160D MVI D,13 ;SPACE OVER
C953 CD21CD CALL BLNK
C956 3A0D00 LDA PATEN
C959 CF6100 CALL BLNK
C95C 160C MVI D,12 ;SPACE OVER
C95F CD21CD CALL BLNK
C962 3A0D00 LDA BADBT ;LAST BAD BYTE
C965 CD0BCD CALL BITS ;PRINT LAST BAD BYTE
C968 C9 REI

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 = BDTBK: EQU OAH ;SPACE FOR DISK READ TRK WHEN ERR
000D = REPETE: EQU ODM, OAH, 'DISK TEST NO. 1 FORMATTED TEST ',0
0055 = ODH, OAH, 'LOAD SCRATCH DISK TYPE Y WHEN READY',0
C9AC = D0A20454HMSG2 DB ODH, OAH, 'END OF PASS ',0
C9AF = D0A2044HMSG3 DB ODM, OAH, 'DATA ERROR OR DISK CHECK ERROR COUNT IN HEX ',0
C9F3 = D0A2054HMSG4 DB ODM, OAH, 'TRACK NO. SECTOR NO. TEST BYTE LAST ERROR'
C37 = D0A4854HMSG5: DB ODM, OAH, 'HEAD POSITION ',0
CAB = D0A4402HMSG6: DB ODM, OAH, 'DISK TRACK CONTROLLER TRACK SECTOR ',0
CAB = D0A402MSG7: DB ODM, OAH, 'II EXECUTION STOPPED II ',0
CAA2 = D0A545HMSG8: DB ODM, OAH, 'TYPE N 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 = CRBR EQU 00000010B ;KEYBOARD READY BIT.
0001 = CFTR EQU 00000011B ;PRINT READY BIT.
0001 = CNUL EQU 1 ;CONSOLE NULL COUNT.
0005 = DISK EQU OFH ;DISK BASE ADDRESS.
0005 = DCOM EQU DISK ;DISK COMMAND PORT.
0005 = DSTAT EQU DISK ;DISK STATUS PORT.
0005 = TRACK EQU DISK+1 ;DISK TRACK PORT.
0005 = SECTP EQU DISK+2 ;DISK SECTOR PORT.
0005 = DDATA EQU DISK+3 ;DISK DATA PORT.
0005 = WAIT EQU DISK+4 ;DISK WAIT PORT.
0005 = DCOMT EQU DISK+4 ;DISK CONTROL PORT.
0005 = TRK: EQU O8H ;ADDRESS FOR TRACK
0005 = SECT: EQU OCH ;ADDRESS FOR SECTOR
READ A CHARACTER FROM CONSOLE.

; READ CONSOLE STATUS.
CADA D803
CADC E602
IN CSTAT
ANI CKRB
; IF NOT READY,

CADA D802
JZ COMIN
; READY WHEN HIGH.

CADA D302
OUT CDATA
; READ A CHARACTER.

CADA E07F
ANI TFH
; MAKE MOST SIG. BIT = 0.

; WRITE A CHARACTER TO THE CONSOLE DEVICE.

; READ A CHARACTER.
CAE6 D302
OUT CDATA
;

; MAKE MOST SIG. BIT = 0.
CAE7 C9
RET
;

; MOVE DISK TO TRACK ZERO.

; CLEAR ANY PENDING COMMAND.
CB05 CDEECA
CONUL: CALL CONOT1
;

; DO COMMON ERROR MSGS.
CB2C 21FACC
HERR: LIX H, EMSG

; PRINT *HOME*.
CB29 7A
MOVE A, D
ANI 91H
;

; SELECT DISK NUMBER.

; DRIVE NO. 1
CB36 3E02
INTDSK: MVI A, 02

; SET THE LATCH WITH CODE.
CB38 D3FC
DSK1: OUT DCONT

; RETURN FROM SELDSK.
CB3A C9
RET
; READ THE SECTOR AT SECT, FROM THE PRESENT TRACK.
; SECTOR IN SECT
; HEAD LOAD FIRST

CB3B 21B000 READ: LXI H,080H  ;READ BUFFER
CB3E 3A0000 LDA SECT
CB41 D3FC READ1: OUT SECTP ;SET SECTOR INTO 1771.
CB43 E5C MOV A,SECPS ;CODE FOR READ W/O HD LD.
CB45 D3F8 READ2: OUT DCON ;SEND COMMAND TO 1771.
CB47 DBF8 RLOOP: IN WAIT ;WAIT FOR DRQ OR INTRO.
CB49 B7 ORA A  ;SET FLAGS.
CB4A F254CB JP RDDONE ;DONE IF INTRO.
CB4D D25CB IN DDATA ;READ A DATA BYTE FROM DISK.

CB4F 77 MOV M,A  ;STORE IN BUFFER
CBO5 23 INX H  ;INC BUFF POINTER

CB51 3A47CB JMP RLOOP

;COMPARE DATA WITH TEST BYTE;
CB54 21B000 RDDONE: LXI H,080H  ;HEAD OF BUFFER
CB57 3A0000 LDA PATEN ;TEST PATTERN
CB5A E3FA READ2: OUT SECTP ;SET SECTOR INTO 1771.
CB5B D3F8 READ3: OUT DCON ;SEND COMMAND TO 1771.
CB5D 7E COMPLP: MOV A,M  ;GET DATA
CB5E 88 CMP B  ;COMPARE WITH TB
CB5F C22ACC JC DATERR ;ERROR

CB62 23 ERRET: INX H  ;INC BYTE COUNT
CB63 15 DCR D  ;DEC BYTE COUNT
CB65 DBF8 IN DSTAT ;READ DISK STATUS.
CB67 B7 ORA D  ;SETフラッグ ON COMBO
CB69 B2 INX D  ;INC BYTE COUNT
CB6B 57 COMPLP: MOV A,M  ;GET DATA
CB6C 1680 MVI D,80H  ;COUNTER FOR BYTES
CB6D 7E CONPLP: NOV A,M  ;GET DATA
CB6E B8 CMP B  ;COMPARE WITH TB
CB6F C22ACC JNZ COMPLP ;DO 128 TIMES

CB71 21EICC LXI H,RDMSG ;PRINT "READ"
CB7B 3A0000 LDA ERRFLG ;READ ERROR FLAG
CB7C 02 ORA D  ;SETフラッグ ON COMBO
CB7E 23 ERRET: INX H  ;INC BYTE COUNT
CB7F 15 DCR D  ;DEC BYTE COUNT
CB81 DBF8 IN DSTAT ;READ DISK STATUS.
CB83 E680 ANI 80H  ;IF BIT 7 IS HIGH,
CB84 2197CC LXI H,MRMSG  ;"NOT READY"
CB87 E690 ANI 80H  ;IF BIT 7 IS HIGH,
CB88 2197CC LXI H,MRMSG  ;"NOT READY"
CB8B E610 ANI 80H  ;IF BIT 7 IS HIGH,
CB8C 2197CC LXI H,MRMSG  ;"NOT READY"
CB8D E610 ANI 80H  ;IF BIT 7 IS HIGH,
CB8E E3FA READ2: OUT SECTP ;SET SECTOR INTO 1771.
CB8F 02 ORA D  ;SETフラッグ ON COMBO
CB91 21EICC LXI H,RDMSG ;PRINT "READ"
CB93 C886 C£8CCC CNZ PNSG
CB95 21A2CC LXI H,RNNSG  ;PRINT "RECORD NOT FOUND"
CB96 CB8C 3A0000 LDA ERRFLG ;READ ERROR FLAG
CB97 C88F C£8CCC CNZ PNSG
CB99 21B4CC LXI H,CRCMSG ;PRINT "CRC ERROR"
CB9A C886 C£8CCC CNZ PNSG
CB9C 21B9CC LXI H,LDNSG ;PRINT "LOST DATA"
CB9D CB8C 3A0000 LDA ERRFLG ;READ ERROR FLAG
CB9E C88F C£8CCC CNZ PNSG
CB9F 21A2CC LXI H,RNNSG  ;PRINT "RECORD NOT FOUND"
CB9F E3FA READ2: OUT SECTP ;SET SECTOR INTO 1771.
CB9F 02 ORA D  ;SETフラッグ ON COMBO
CB9F E608 ANI 80H  ;IF BIT 7 IS HIGH,
CB9F 2199CC LXI H,MRMSG  ;"NOT READY"
CB9F CB8C 3A0000 LDA ERRFLG ;READ ERROR FLAG
CB9F C88F C£8CCC CNZ PNSG
CB9F 21A2CC LXI H,RNNSG  ;PRINT "RECORD NOT FOUND"

COMMON ERROR PRINT OUT

CB77 7A ERMSG1: MOV A,D  ;GET ERROR BITS.
CB78 F68D ANSI 80H  ;IF BIT 7 IS HIGH,
CB7A 2197CC LXI H,MRMSG  ;"NOT READY"
CB7D CB88 3A0000 LDA ERRFLG ;READ ERROR FLAG
CB7E CB8B 3A0000 LDA ERRFLG ;READ ERROR FLAG
CB7F 02 ORA D  ;SETフラッグ ON COMBO
CB81 21EICC LXI H,RDMSG ;PRINT "READ"

118
WRITE THE SECTOR AT SECT, ON THE PRESENT TRACK.
USE STARTING ADDRESS AT DMAADD.
LOAD HEAD FIRST.

WRITE: LDA PATEN ;TEST PATTERN IN B
MOV B,A ;LOAD SECTOR
OUT SECTP ;SET THE SECTOR INTO 1771.
MVI A,04H ;SET UP 1771 FOR WRITE.
OUT DCOM
WLOOP: IN WAIT ;WAIT FOR READY.
JP DONE ;HOP OUT WHEN DONE.

; INSERT PATTERN HERE

MOV A,B ;LOAD TEST PATTERN
CC48 D3FB OUT DDATA ;WRITE ONTO DISK.
CC4D 23 INX H ;INCREMENT MEM PTR.
CC4E C344CC JMP WLOOP ;KEEP WRITE.
CC51 DBF8 WDONE: IN DSAT ;READ DISK STATUS.
CC53 E6FD ANI OFDH ;LOOK AT THESE BITS.
CC55 57 MOV D,A ;SAVE STATUS BITS
CC56 CB PROCER: RZ ;RETURN IF NO ERR.
CC57 21E9CC WERRO: LXI H,WTMSG ;PRINT "WRITE ".
CC5A CD8CCC CALL PMSG
CC5D 7A NOV A,D ;GET ERROR BITS.
CC5E E640 ANI OFDH ;LOOK AT THESE BITS.
CC60 21D3CC LXI H,WTMSG ;PRINT "FAULT ".
CC63 C4BCCC CNZ PMSG
CC66 7A NOV A,D ;GET ERROR BITS.
CC67 E620 ANI 20H ;LOOK AT BIT 5.
CC69 21D3CC LXI H,WTMSG ;PRINT "FAULT ".
CC6C C4BCCC CNZ PMSG
CC6F C377CB JMP ERMSG1 ;DO COMMON ERR MESSAGES.
CC72 D3FB SEEK: OUT DDATA ;TRACK TO DATA REGISTER.
CC74 DBF8 BUSY: IN DSAT ;READ DISK STATUS.
CC76 0F RRC ;LOOK AT BIT 0.
CC77 DA7ACC JC BUSY ;WAIT TILL NOT BUSY.
CC7A 3E12 MVI A,12H ;SET FOR 10 MS STEP.
CC7C D3FB OUT DCOM ;ISSUE SEEK COMMAND.
CC7E DBF8 IN DSAT ;READ STATUS.
CC80 E691 ANI 91H ;LOOK AT BIT 5.
CC84 57 MOV D,A ;SAVE STATUS.
CC85 CB RZ ;RETURN IF NO ERR.
CC86 21F2CC LXI H,SKMSG ;PRINT "SEEK ".
CC89 C374CS JMP ERMSG1 ;DO COMMON ERR MESSAGES.
CC8C 7E PMSG: MOV A,M ;GET A CHARACTER.
CC8D B7 ORA A ;IF IT'S ZERO,
CC8E CB RI ;RETURN.
CC8F 4F MOV C,A ;OTHERWISE.
CC90 CDEBCA CALL CONOT ;PRINT IT.
CC93 23 INI H ;INCREMENT HAL.
CC94 C38CCC JMP ERMSG1 ;DO COMMON ERR MESSAGES.
CC97 4E4F525052NRMSG: DB 'NOT READY ',O
CC9A 524553F52RRMSG: DB 'RECORD NOT FOUND ',O
CC9B 4552F43200CRRMSG: DB 'CRC ',O
CC9C 4E55535920BSRMSG: DB 'BUSY ',O
CC9D 525553585450WPMMSG: DB 'FAULT ',O
CC9E 4E4F525252ERRMSG: DB 'ERROR ',O
CC9F 004A5451RRMSG: DB 'READ ',O
CCA0 004A575249WMTMSG: DB 'WRITE ',O
CCA1 004A48450SRSMSG: DB 'SEEK ',O
CCA2 004A424559MSMMSG: DB 'HOME ',O
CCA3 004A544559MMTMSG: DB 'MOUNT ',O

FOOD = REENT EQU 0FOODH ;MONITOR ENTRY

CC97 4E4F542052NRMSG: DB 'NOT READY ',O
CCA2 524553F52RRMSG: DB 'RECORD NOT FOUND ',O
CCB9 4552F43200CRRMSG: DB 'CRC ',O
CCBB 4E55535920BSRMSG: DB 'BUSY ',O
CCBC 525553585450WPMMSG: DB 'FAULT ',O
CCBD 4E4F525252ERRMSG: DB 'ERROR ',O
CCBE 004A5451RRMSG: DB 'READ ',O
CCBF 004A575249WMTMSG: DB 'WRITE ',O
CCC0 004A48450SRSMSG: DB 'SEEK ',O
CCC1 004A424559MSMMSG: DB 'HOME ',O
CCC2 004A544559MMTMSG: DB 'MOUNT ',O

120
PRINT 8 BIT WORD IN BINARY FORMAT

INPUT: DATA IN REG A

CD08 47
CD0C 0E00
CD10 0F5
CD11 0F1
CD12 0E31
CD17 0E8CA
CD1A 0A0
CD1B 07B
CD1C 1F
CD1D 020CD
CD20 0C9

CD21 0E20
CD23 0E8CA
CD25 15
CD27 023CD
CD2A 0C9

CD2B 07A
CD2C 1F
CD2D 1F
CD2E 1F
CD2F 1F
CD30 0D48CD
CD33 4F
CD34 0E8CA
CD37 07A
CD38 0D48CD
CD3B 4F
CD3C 0E8CA

CD40 0D2BCD
CD43 053
CD44 0D2BCD
CD47 0C9

CD4A 0E8F
CD4A 0D30
CD4C 0F3A
CD4D 0D8
CD4F 0607
CD51 0C9

CD00 0E00
CD04 0E30
CD10 0F5
CD11 0F1
CD12 0E31
CD17 0E8CA
CD1A 0A0
CD1B 07B
CD1C 1F
CD1D 020CD
CD20 0C9

CD21 0E20
CD23 0E8CA
CD25 15
CD27 023CD
CD2A 0C9

CD2B 07A
CD2C 1F
CD2D 1F
CD2E 1F
CD2F 1F
CD30 0D48CD
CD33 4F
CD34 0E8CA
CD37 07A
CD38 0D48CD
CD3B 4F
CD3C 0E8CA

CD40 0D2BCD
CD43 053
CD44 0D2BCD
CD47 0C9

CD4A 0E8F
CD4A 0D30
CD4C 0F3A
CD4D 0D8
CD4F 0607
CD51 0C9

CD00 0E00
CD04 0E30
CD10 0F5
CD11 0F1
CD12 0E31
CD17 0E8CA
CD1A 0A0
CD1B 07B
CD1C 1F
CD1D 020CD
CD20 0C9

CD21 0E20
CD23 0E8CA
CD25 15
CD27 023CD
CD2A 0C9

CD2B 07A
CD2C 1F
CD2D 1F
CD2E 1F
CD2F 1F
CD30 0D48CD
CD33 4F
CD34 0E8CA
CD37 07A
CD38 0D48CD
CD3B 4F
CD3C 0E8CA

CD40 0D2BCD
CD43 053
CD44 0D2BCD
CD47 0C9

CD4A 0E8F
CD4A 0D30
CD4C 0F3A
CD4D 0D8
CD4F 0607
CD51 0C9

121
INIrIATE SIO PORTS

CD52 3EAA
INITA: MVI A,0AAH ;GET DUMMY MODE WORD
CD54 D303
OUT CSTAT ;OUTPUT IT
CD56 3E40
MVI A,40H ;GET RESET BIT
CD58 D303
OUT CSTAT ;RESET SIO BOARD
CD5A 3ECE
MVI A,0CEH ;GET REAL MODE WORD
CD5C D303
OUT CSTAT ;SET THE MODE FOR REAL
CD5E 3E37
MVI A,37H ;GET THE COMMAND
CD60 D303
OUT CSTAT ;OUTPUT IT
CD62 C9
RET

CD63 DEOD
CRLF: MVI C,13 ;CR
CD65 DEECA
CALL CONOT
CD68 DEOA
LF: MVI C,10 ;LF
CD6A DEECA
CALL CONOT1
CD6D OE7F
MVI C,7FH
CD6F DEECA
CALL CONOT1
CD72 CD52CD
CALL CONOT
CD75 C9
NET DISC TEST FULL TRACK WRITE
SELECT TRACK IN SENSE SWITCHES
PROM VERSION
BRIAN DONLAN JUNE 79

CD80 ORG OCDB0H:
CD80 C3BBCD
ENTRYA: JMP STARTA
CD83 218000 ENTRYB: LXI H,080H ;SET STACK
CD86 F9
SPHL
CD87 FE59 CPI 'Y'
CD8B 21F7CD
STARTA: LXI H,MSG1B
CD8E CD68CC
READT: CALL PMSG ;ERROR FLAG OTHER TEST
CD91 3E00
MVI A,00H
CD93 320800
STA ERRFLG
CD96 CD68CA
CALL CONIN ;READ KEYBOARD
CD99 FE59
CPI 'Y'
CD9E 2139CE
LXI H,MSG2A
CD9F C28ECD
JNZ READT
CDAC DB88CB
CALL HOME
CDAA B7
STARTB: ORA A ;SET FLAGS
CDAB C2BBCD
JNI H,STARTA ;ERROR START OVER
CDAC 21A4CD
LXI H,STARTB ;SUBROUTINE RETURN
CDAD E5
PUSH N
CDAE DBF6
STARTC: IN OFFH ;READ SENSE SWITCHES
CDAE F40D
CPI 77 ;PREVENT TRACK OVER-DRIVE
CDB0 D2C8CD
JNC ERRA
CDB3 CD72CC
SEEKA: CALL SEEK ;MOVE HEAD TO TRACK
CDB6 0FF
MVI B,OFFH ;TEST PATTERN
CDB8 3EFA
MVI A,OFFH ;WRITE TRK COMMAND
CDBA D3FB
OUT DCOM
CDBC DBFC
WRITLP: IN WAIT
CDBE B7
ORA A
CDBF F251CC
JP WWRITE
CDD2 7B
MOV A,B
CDD3 D3FB
OUT DATA
CDD5 C3BBCD
JMP WRITLP
DISC TEST FULL TRACK READ
SELECT TRACK IN SENSE SWITCHES
PROM VERSION
BRIAN DONLAN
JUNE 79

ORG OCE4OH
ENTRYD: JMP STARTD
ENTRYE: LXI H,080H
SPHL ;STACK
DI
CD52CD CALL IMITA ;RESET SIO
CD4CE STARTD: LXI H,MSG1E
CD8CC READU: CALL MSG1E ;READ KEYBOARD
CE51 CD08CE CALL HOME
CE64 B7 STARTE: ORA A ;SET FLAGS
CE66 C24BCE NJWZ STARTD ;ERROR START OVER
CE69 2164CE LXH H,STATE ;SUBROUTINE RETURN
CE6A 211CE PUSH H
CE6B 22F DI ;READ SENSE SWITCHES
CE97 FEAD STARTF: IN OFFH ;PREVENT TRACK OVER-DRIVE
CE98 285CE IN 77 ;MOVE HEAD TO TRACK
CE99 CD72CC CD72CC: CALL SEEK
CEA0 3E5 MVI A,DESH ;READ COMMAND
CEA5 33F8 OUT DCOM
CEA9 2BCE RDLP: JMP RDLP
CEB0 24CE NJWZ
CEB2 C37ACE ORA A
CEB7 B7 ORA A
CEB9 2BACE JP RHOME
CEBE 2BFB IN DDATA
CEB2 C37ACE JMP RDLP
CE85 06FO ERRD: MVI B, OF0H
CE87 78 ERRLPD: MOV A, B
CE88 2F CHA
CE89 D3FF OUT OFFH
CE8B 110100 LXI D, 01H
CE8E 210000 LSI H, 000H
CE91 19 ERRLPE: DAD D
CE92 D291CE JNC ERRLPE
CE95 47 MOV B, A
CE96 D2FF IN OFFH
CE98 E9D CPI 77
CE9A D8F8 CPI 77
CE9B D2A3CE JNC DELPE
CEA0 110100 LXI H, 01H
CEA3 19 DELPD: DAD D
CEA4 D287CE JNC ERRLPD
CEA7 210000 DELAYE: LXI D, 01H
CEAE D200CE LXI D, 01H
CEB1 C36CCE JMP STARTF
CEB4 DBF8 RDONE: IN DSTAT
CEB6 E9D ANI 0DH
CEB8 57 MOV D, A
CEBA 21E1CC LXI H, 01H
CEBD C374CB JMP ERMSG
CEC0 05CAA9E95C1E DB ODH, OAH, OAH, "DISK TRACK READ ROUTINE"
CEC8 05C4F41 DB ODH, OAH, "LOAD SCRATCH DISK TYPE Y WHEN READY", 0
ORG 0000H

; UNIBUS PORT TEST
; AND UBIBUS COMM. TEST COMBINED
; AND SNAP-SHOT
; PROM VERSION

ENTRY1: LXI H,080H
SPHL ;SET STACK POINTER
ENTRY: LXI H,ENTRY
PUSH H
DI
CD0AD2 CALL INITA ;RESET IO
CD50D2 CALL PMSG

; BEGINNING OF TEST
E501 MVI A,01H
E502 MVI C,10H ;PORT UNDER TEST
D503 D510 PORT10: OUT 10H
D507 47 MOV B,A ;SAVE TEST PATTERN
D509 DB10 IN 10H ;READ BUS
D511 BB CMP B ;COMPARE
D514 C45FD0 CMZ ERR ;CALL IF ERROR
D517 07 RLC ;TEST FOR A COMPLETE CIRCLE
D519 D216D0 JNC PORT10
D521 E01 MVI A,01H
D523 E011 MVI C,11H ;PORT 11
D525 E011 PORT11: OUT 11H
D527 47 MOV B,A ;SAVE PATTERN
D529 DB11 IN 11H ;READ BUS
D531 BB CMP B ;COMPARE
D534 C45FD0 CMZ ERR ;CALL IF ERROR
D536 07 RLC ;TEST FOR A COMPLETE CIRCLE
D538 D227D0 JNC PORT11
D540 E01 MVI A,01H
D542 E012 MVI C,12H ;PORT 12
D544 E012 PORT12: OUT 12H
D546 47 MOV B,A ;SAVE TEST PATTERN
D548 DB12 IN 12H ;READ BUS
D550 BB CMP B ;COMPARE
D552 C45FD0 CMZ ERR ;CALL IF ERROR
D554 07 RLC ;TEST FOR A COMPLETE CIRCLE
D556 D238D0 JNC PORT12
D558 E01 MVI A,01H
D560 E013 MVI C,13H
D562 E013 PORT13: OUT 13H
D564 47 MOV B,A ;SAVE TEST PATTERN
D566 DB13 IN 13H
D568 BB CMP B ;COMPARE
D570 C45FD0 CMZ ERR
D572 07 RLC ;TEST FOR A COMPLETE CIRCLE
D574 D249D0 JNC PORT13
D576 21D5D0 LXI H,MSG4 ;FINISHED MESSAGE
D578 D50D2 CALL PMSG ;RETURN TO MONITOR

125
D05F C5   ERR: PUSH B      ;SAVE ERROR PATTERN
D060 F5   PUSH A       ;SAVE TEST PATTERN
D061 51   ...V        D,C
D062 219B0D LXI H,MSG2
D063 CD50D2 CALL PMSG
D064 CD7BD2 CALL BINHA  ;PRINT 2 DIGITS
D065 21AE0D LXI H,MSG0  ;ERROR MESSAGE
D066 CD50D2 CALL PMSG
D067 7E   MOV A,B      ;LOAD TEST PATTERN
D068 CD5BD2 CALL BITS  ;PRINT TEST PATTERN
D069 2101D0 LXI H,MSG3  ;MORE TE' T
D06A CD50D2 CALL PMSG
D06B F1   POP A
D06C CD5BD2 CALL BITS  ;PRINT ERROR PATTERN
D06D C1   POP B        ;RESTORE B AND C
D06E CD57D2 MOV A,B     ;MOVE TEST PATTERN TO A
D06F CD5BD2 CALL PMSG
D070 7B   MOV A,B      ;RESTORE B AND C
D071 CD5BD2 CALL PMSG
D072 CD5BD2 CALL PMSG
D073 CD5BD2 CALL PMSG
D074 CD5BD2 CALL PMSG
D075 CD5BD2 CALL PMSG
D076 CD5BD2 CALL PMSG
D077 CD5BD2 CALL PMSG
D078 CD5BD2 CALL PMSG
D079 CD5BD2 CALL PMSG
D07A CD5BD2 CALL PMSG
D07B CD5BD2 CALL PMSG
D07C CD5BD2 CALL PMSG
D07D CD5BD2 CALL PMSG
D07E CD5BD2 CALL PMSG
D07F CD5BD2 CALL PMSG

D080 A0A0D554EMSG1 DB OAH,OAH,ODH,'UNIBUS PORT TEST',0
D081 A0A0D552EMSG2 DB OAH,OAH,ODH,'ERROR PORT NO. ',0
D082 A0A0D553EMSG3 DB OAH,OAH,ODH,'TEST PATTERN ',0
D083 A0A0D554EMSG4 DB OAH,OAH,ODH,'END OF TEST ',0
FO00 =  RET EQU CF000H     ;MONITOR ENTRY

;UNIBUS COMMUNICATION TEST

D100 ORG GO100H
D100 :8000 ENTRY2: LXI H,0800H
D101 F9   SPHL       ;SET STACK
D102 CDDAD2 CALL INITA
D103 CD57D1 ENTRY3: LXI H,ENTRY3
D104 CD5BD2 CALL PMSG
D105 CD5BD2 CALL PMSG
D106 CD5BD2 CALL PMSG
D107 CD5BD2 CALL PMSG
D108 CD5BD2 CALL PMSG
D109 CD5BD2 CALL PMSG
D10A CD5BD2 CALL PMSG
D10B CD5BD2 CALL PMSG
D10C CD5BD2 CALL PMSG
D10D CD5BD2 CALL PMSG
D10E CD5BD2 CALL PMSG
D10F CD5BD2 CALL PMSG
D110 CD5BD2 CALL PMSG
D111 CD5BD2 CALL PMSG
D112 CD5BD2 CALL PMSG
D113 CD5BD2 CALL PMSG
D114 CD5BD2 CALL PMSG
D115 CD5BD2 CALL PMSG
D116 CD5BD2 CALL PMSG
D117 CD5BD2 CALL PMSG
D118 CD5BD2 CALL PMSG
D119 CD5BD2 CALL PMSG
D11A CD5BD2 CALL PMSG
D11B CD5BD2 CALL PMSG
D11C CD5BD2 CALL PMSG
D11D CD5BD2 CALL PMSG
D11E CD5BD2 CALL PMSG
D11F CD5BD2 CALL PMSG
D120 CD5BD2 CALL PMSG
D121 CD5BD2 CALL PMSG
D122 CD5BD2 CALL PMSG
D123 CD5BD2 CALL PMSG
D124 CD5BD2 CALL PMSG
D125 CD5BD2 CALL PMSG
D126 CD5BD2 CALL PMSG
D127 CD5BD2 CALL PMSG
D128 CD5BD2 CALL PMSG

;OUTPUT MODE
D129 21CD71 PUTOUT: LXI H,MSG11 ;OUTPUT MESSAGE
D130 CD50D2 CALL PMSG
D131 CD50D2 CALL BINM  ;GET DIGITS TO OUTPUT
D132 CD5BD1 CALL DATAO ;UNIBUS DRIVER
D133 CD5BD1 JMP DONE
D134 CD5BD1 PUTIN: LXI H,MSG9 ;INPUT MESSAGE
D135 CD50D2 CALL PMSG
D136 CD50D2 CALL BINM  ;GET DIGITS TO OUTPUT
D137 CD50D2 CALL DATAI ;UNIBUS INPUT ROUTINE
D138 CD5BD1 JMP DONE
D139 CD5BD1 DOME: LXI H,MSG10 ;PRINT END OF TEST
D140 CD5BD1 JMP ENTRY3

126
D157 21C2D1  QUEST:  LXI  H, MSG7  
D15A CDS2D2  CALL  PMSG  ; ??  
D15D C31CD1  JMP  TRYON  

D160 DAO0D548E5MSG6:  DB  OAH,ODH,'UNIBUS COMMUNICATION TEST',O  
D17C DAO0D48E5MSG6:  DB  OAH,ODH,'INPUT (I), OUTPUT (O), EXIT (CONTROL C) ?',O  
D19C DAO0D20D1EMSG7:  DB  OAH,ODH,'ENTER DATA TO OUTPUT IN 4 HEX DIGITS ',O  
D1F1 DAO0D20D1MSG9:  DB  OAH,ODH,'DATA FROM BUS ',O  
D205 DAO0D548E5MSG10:  DB  OAH,ODH,'TRANSFER COMPLETE',O  

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

0003  =  CSTAT  EQU  3  ;CONSOLE STATUS PORT.  
0003  =  CCMD  EQU  3  ;CONSOLE COMMAND PORT.  
0002  =  CCDATA  EQU  2  ;CONSOLE DATA PORT.  
0002  =  CKBR  EQU  00000010B ;KEYBOARD READY BIT.  
0001  =  CPTR  EQU  00000001B ;PRINT READY BIT.  
0001  =  CNULL  EQU  0  ;CONSOLE NULL COUNT.  

; CHECK CONSOLE INPUT STATUS.  

D219 DB03  CONST:  IN  CSTAT ;READ CONSOLE STATUS.  
D21B ED02  ANI  CKBR ;LOOK AT KB READY BIT.  
D21D EDOO  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 ED02  ANI  CKBR ;IF NOT READY,  
D226 CA22D2  JZ  CONIN ;READY WHEN HIGH.  
D229 DB02  IN  CDATA ;READ A CHARACTER.  
D22B D302  OUT  CDATA  
D22D ED7F  ANI  TFH ;MAKE MOST SIG. BIT = 0.  
D22F C9  RET ;RETURN.  

; WRITE A CHARACTER TO THE CONSOLE DEVICE.  

D230 3E0D  CONOT:  MVI  A,ODH ;IF IT'S A CR,  
D232 B9  CMP  C ;THEN HOP OUT  
D233 CAX1D2  JZ  CONOT1 ;TO NULL ROUTINE.  
D236 DB03  CONOT1:  IN  CSTAT ;READ CONSOLE STATUS.  
D238 ED01  ANI  CPTR ;IF NOT READY,  
D23A CA32D2  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 CD3SD2  CONULY:  CALL  CONOT1 ;PRINT CR.  
D247 0EO0  MVI  C,0 ;GET NULL CHAR.  
D249 05  DCH  B ;DECREMENT COUNTER.  
D24A C24D2  JMP  CONUL1 ;DO NEXT NULL.  
D24D C1  POP  B ;RESTORE B&C.  
D24E 79  MOV  A,C ;RESTORE A.  
D24F C9  RET ;RETURN.
PRINT MESSAGE UNTIL ZERO
MESSAGE ADDRESS REG H & L

PRINT: MESSAGE
D250 7E PMSG: MOV A,M ;GET CHAR
D251 B7 ORA A ;IS IT A ZERO
D252 C8 RZ
D253 4F MOV C,A ;OTHERWISE PRINT
D254 CD30D2 CALL CONOT
D257 23 INX H ;INC ADDRESS
D258 C350D2 JMP PMSG

PRINT 8 BIT WORD IN BINARY FORMAT
INPur:
DATA IN REG A

D25B 47 BITS: MOV B,A ;DATA
D25C 38B0 MVI A,80H ;MASK
D25E 0E30 OVER: MVI C,30H ;STORE MASK
D260 5F MOV E,A ;AND WITH MASK
D261 AO ANA B ;ZERO CARRY
D262 CA67D2 JZ PRNT ;JUMP IF ZERO
D265 OE31 MVI C,31H
D267 CD30D2 PRINT: CALL CONOT
D26A AO ANA B ;ZERO CARRY
D26B 7B MOV A,E ;LOAD MASK
D26C 1F RAR
D26D D25ED2 JNC OVER
D270 C9 RET

PRINT BLANKS, # IN REG. D
D271 OE20 BLNK: MVI C,20H ;PRINT BLANKS, # IN REG. D
D273 CD36D2 LP17: CALL CONOT1
D275 15 DCR D
D277 C273D2 JNZ LP17
D27A C9 RET

OUTPUTS 2 HEX DIGITS IN ASCII FROM REG D

BINHA: MOV A,D
D27B 7A
D27C 1F RAR
D27D 1F RAR
D27E 1F RAR
D27F 1F RAR
D280 CD9BD2 CALL BIN1
D283 4F MOV C,A
D284 CD30D2 CALL CONOT
D287 7A MOV A,D
D288 CD9BD2 CALL BIN1
D28B 4F MOV C,A
D28C CD30D2 CALL CONOT
D28F C9 RET

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

BINB: CALL BINHA
D290 CD7BD2
D291 53 MOV D,E
D294 CD7BD2 CALL BINHA
D297 C9 RET
CONVERTS HEX TO ASCII

INPUT: 4 BITS HEX REG A
OUTPUT: 8 BIT ASCII REG A

```
D298 E60F
D29A C630
D29C FE3A
D29E D8
D29F C607
D2A1 C9

BIN1: ANI OFH
ADI 3OH
CPI 3AH
RC
ADI 0TH
RET
```

INPUTS 4 DIGITS FROM CONSOLE
RETURN; 4 HEX DIGITS IN REG E-D

```
D2A2 CD22D2
D2A5 CDD1D2
D2A8 17
D2A9 17
D2AA 17
D2AB 17
D2AC E6F0
D2AE 57
D2AF CD22D2
D2B2 CDD1D2
D2B5 E60F
D2B7 B2
D2B8 57
D2B9 CD22D2
D2BC CDD1D2
D2BF 17
D2C0 17
D2C1 17
D2C2 17
D2C3 E6F0
D2C5 5F
D2C6 CD22D2
D2C9 CDD1D2
D2CC E60F
D2CE 83
D2CF 5F
D2D0 C9
D2D1 00
D2D2 D630
D2D4 FE0A
D2D6 D8
D2D7 D607
D2D9 C9

AHS1: NOP
SUI 3OH
CPI GAH
RC
SUI 0TH
RET
```

CONVERT ASCII TO HEX

INPUT: 8 BIT ASCII REG A
OUTPUT: 4 BIT HEX REG A

```
D2D1 00
D2D2 D630
D2D4 FE0A
D2D6 D8
D2D7 D607
D2D9 C9
```
INITIATE SIG PORTS

D2DA 3EAA INITA: MVI A,0AAH ;GET DUMMY MODE WORD
D2DC D203 OUT CSIA ;OUTPUT IT
D2DE 3E40 MVI A,40H ;GET RESET BIT
D2EO D303 OUT CSIA ;RESET SIG BOARD
D2FA D303 MVI A,OCFH ;GET REAL MODE WORD
D2ED D303 OUT CSIA ;SET THE MODE FOR REAL
D2EE 3E37 MVI A,37H ;GET THE COMMAND
D2EF D303 OUT CSIA ;OUTPUT IT
D2EA C9 RET

D2EB 0ECD CRLF: MVI C,13 ;CR
D2EF 0ECD CALL CONOT
D2F3 0E3F LF: MVI C,10 ;LF
D2F5 0E3D CALL CONOT
D2F6 0E3C CALL CONOT
D2F7 0E3D CALL CONOT
D2F8 0ECD RET

D2FE CD91D3 DATAI: CALL GETBUS
D300 CD223 CALL DATI
D302 97 SUB A
D303 D315 OUT 15H
D304 C9 RET

D305 CD91D3 DATAO: CALL GETUS
D306 CD51D3 CALL DATO
D307 97 SUB A
D308 D315 OUT 15H
D309 C9 RET

;ROUTINE TO INPUT A 16 BIT WORD FROM UNIBUS
;REG B = A<15:09>, REG C = A<08:01>
;DATA WILL BE CONTAINED IN REG D = D<15:08>, REG C = D<07:00>

D312 3EFF DIATI: MVI A,OFFH ;SET LOOP COUNT
D313 328200 STA BIWCG
D317 0B14 BIZLP1: IN 14H ;CHECK FOR SYS = 0
D319 E604 ANI 04H ;FROM LAST TRANSACTION
D31A 2A22D3 JXZ BBUSYI
D31E 7B 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
D326 F001 ORI 01H ;OUTPUT MSTI=1
D327 D314 OUT 14H
D328 3EFF STWLP1: MVI A,OFFH ;LOOP COUNT
D32D 328100 STA SYNCG
D330 DB14 DILOP: IN 14H ;CHECKS IF SYN=1
D332 D3FF OUT OFFH
D334 E604 ANI 04H
D336 CAC8D3 JZ MSTI1

130
D339 DB12  ;INPUT HIGH DATA
D338 57  MOV D,A
D333 DB13  ;INPUT LOW DATA
D33E 5F  MOV E,A

D33F 97  SUB A
D340 DB14  ;CLEAR MSYN
D342 D311  OUT 10H
D341 D311  OUT 11H
D346 D312  OUT 12H
D348 D313  OUT 13H
D34A DBFF  IN OFFH
D34C 97  ORA A
D34D C21D3  JNZ DATI ;SET FLAGS
D350 C9  RET

;ROUTINE TO OUTPUT A 16 BIT WORD ON THE UNIBUS
;REG B = A<16:09>, REG C = A<08:01>
;REG D = D<15:08>, REG E = D<07:00>
D351 3EFF  ;ROUTINE TO OUTPUT A 16 BIT WORD ON THE UNIBUS
D353 328000  STA BIZCNT
D356 DB14  IN 14H
D358 E504  ANI 04H
D35A C285D3  JNZ BBUSY2

D35D 7B  MOV A,B
D35E D310  OUT 10H
D360 7B  MOV A,C
D361 D311  OUT 11H
D364 D312  OUT 12H
D366 7B  MOV A,E
D367 D313  OUT 13H
D369 3E02  MVI A,02H
D36A D314  OUT 14H

D36D 3E03  MVI A,03H
D36F D314  OUT 14H

D371 3EFF  ;DATA: MVI A,OFFH
D373 328100  STA BIZCNT
D376 DB14  IN 14H
D378 E504  ANI 04H
D37A C28D3  JNZ BBUSY2

D37C CAFD3  JZ NOSYN2
D37F 97  SUB A
D380 D314  OUT 14H
D382 D310  OUT 10H
D384 D311  OUT 11H
D386 D312  OUT 12H
D388 D313  OUT 13H
D38A DBFF  IN OFFH
D38C 97  ORA A
D390 C21D3  JNZ DATO ;SET FLAGS
D390 C9  RET

D391 3EFF  ;LOOP IF UP
D393 328000  STA GETCNT
D396 3E01  MVI A,01H
D398 D315  OUT 15H
D39A DB15  LOOP IN 15H
D39C E501  ANI 01H
D39E CA0CDA  JZ MOGET
D3A1 C9  RET

131
ON-LINE UNIBUS DIAGNOSTICS
BY BRIAN DONLAN 24 APR 79

; BBUSY1: LDA BIZCNT ;LOOP COUNT
; STA BIZCNT ;NEW COUNT
; JNZ BIZLP1 ;JUMP IF STILL COUNT
; LXI A, ERHMSG2
; CALL PMSG ;DISPLAY ERROR MESSAGE
; JMP ENTRY3

; BBUSY2: LDA BIZCNT
; DCR A
; STA BIZCNT
; JNZ BIZLP2
; LXI H, ERHMSG2
; CALL PMSG
; JMP ENTRY3

; BBUSY3: MOVE D,B ;MOV ADDRESS FOR OUTPUT
; MOV E,C
; LXI H, ERHMSG3
; CALL PMSG ;OUTPUT ADDRESS
; JMP ENTRY3

; NOSYN1: LDA SYCNT
; DCR A
; STA SYCNT
; JNZ DILoop
; IN OFFH
; ANI 040H
; JNZ DILoop
; MOV D,B
; JMP ENTRY3

; NOSYN2: LDA SYCNT
; DCR A
; STA SYCNT
; JNZ DILoop
; IN OFFH
; ANI 040H
; JNZ DILoop
; JMP ENTRY3

; NOSYN3: LDA GETCNT
; DCR A
; STA GETCNT
; JNZ LOOP
; LXI H, ERHMSG1
; CALL PMSG
; JMP ENTRY3

; GETCNT: EQU 080H
; SYCNT: EQU 081H
; BIZCNT: EQU 082H
; ERHMSG1: DB 'ODM,ODM,OAH, ERRBUSS BUSY',0
; ERHMSG2: DB 'ODM,ODM,OAH, ERROR BUSS BUSY',0
; ERHMSG3: DB 'ODM,OAH, DEVICE NO. ',0
; ERHMSG4: DB 'NO RESPONSE ',0

132
; UNIBUS SNAP SHOT ROUTINE
ORG 00500H
ENTRY: LXI H, 0F000H
PUSH H
DI INITA
;RESET I/O

; SUBROUTINE ENTRY POINT
ENTRIES: LXI H, MSG12

D500 2100F0 CALL CMSD5
D501 DF10 IN 10H
D501 57 MOV D, A
D502 D511 IN 11H
D502 5F MOV E, A
D514 CD90D2 CALL BINS
; PRINT I/O

D517 1608 MVI D, 0BH
D519 CD71D2 CALL BLNK
; SPACE OVER
D51C D812 IN 12H
D51E 57 MOV D, A
D51F DB13 IN 13H
D521 5F MOV E, A
D522 CD90D2 CALL BINS
D525 1608 MVI D, 0BH
D527 CD71D2 CALL BLNK
; PRINT UNIBUS ADDRESS

D52A DB14 IN 14H
D52C E504 ANI 04H
D52E CA36D5 JZ MOSIS
D531 0E31 MVI C, '1'
D533 CD3BD5 JMP OUTMIS
D536 OE30 MOSIS: MVI C, '0'
D538 CD3BD2 OUTMIS: CALL CONMIS
D53B 1609 MVI D, 0BH
D53D CD71D2 CALL BLNK
; PRINT SLAVE SYN

D540 DB15 IN 15H
D542 E501 ANI 01H
D544 CA4CD5 JZ NOBUS
D547 OE31 MVI C, '1'
D549 CD4ED5 JMP OUTBUS
D54C OE30 BUS: MVI C, '0'
D54E CD3BD2 OUTBUS: CALL COMOUT
D551 1609 MVI D, 0BH
D553 CD71D2 CALL BLNK
; PRINT BUS GRANT

D556 DB10 IN 14H
D558 E500 ANI 00H
D55A CA62D5 JZ MONSTN
D55D OE31 MVI C, '1'
D55F CD6ED5 JMP OUTMSN
D562 OE30 MONTSN: MVI C, '0'
D564 CD3BD2 OUTMSN: CALL COMOUT
D567 0F RET
D56F C368D5 FINISH: JMP FINISH

D56B 0A0A5D554FMSG12: DB OAH, OAH, ODH, 'UNIBUS SNAP-SHOT',
D4B0 0A04114444: DB OAH, ODH, 'ADDRESS DATA SYN GRANT M SYN'
D5B0 0A00200000: DB OAH, ODH, '0', 0
; BK MINI MEMORY TEST

; BRIAN DONLAN
; PROM VERSION D600

ORG 0D600H

D600 F3 ; ENTER: DI
D601 3EFF ; OUTPUT PHASE I LITES
D602 210000
D603 AF ; START ADDRESS
D604 77 ; ZERO ACC
D605 A6 ; STORE TEST PATTERN IN MEM.
D606 86
D607 C250D6 ; READ BACK TO B
D608 3C
D609 FF ; COMPARE FOR OK
D60A C26DD6 ; JUMP IF ERROR
D60B 3C
D60C C209D6 ; NEW TEST PATTERN
D60D 210000

D610 E8
D611 77 ; STORE TEST PATTERN IN
D612 46 MOV B,N
D613 88 CMP B
D614 C26DD6 JNZ ERR1
D615 3C INR A
D616 C209D6 JNZ LP1
D617 210000

D618 19
D619 EB ; ADD TWO'S COMPLIMENT
D61A D208D6 JNC LP2

D61B 3EFE ; PHASE III LITES
D61C 19 XCHG
D61D D24BD6 JNC LP3

D61E 1100E0
D61F 19
D620 EB
D621 XCHG
D622 D2BD6 JNC LP4

D623 210000
D624 7E
D625 23
D626 1100E0
D627 EB
D628 XCHG
D629 D224D6 JNC LP5

D62A 19
D62B EB
D62C XCHG
D62D D222D6 JNC LP6

D62E 1100EO
D62F 19
D630 EB
D631 XCHG
D632 D232D6 JNC LP4

D633 210000
D634 7E
D635 23
D636 EB
D637 XCHG
D638 D25FD6 JNC LP5

D639 210000
D63A 7E
D63B 23
D63C EB
D63D XCHG
D63E D255D6 JNC LP6
D655 3EFF   MVI  A,OFFH  ;ALL PHASE COMPLETE
D657 2100D6  LXI  H,ENTER  
D666A C3ABD6  JMP  LITES  ;GO TO LITES PROG

D66D EB  PHASE I ERROR  
D66E 4F  ERR1: XCHG  
D66F 2177D6  LXI  H,COMERR  ;RETURN
D671 C3ABD6  JMP  LITES

D677 7A  COMMON ERROR OUTPUT ROUTINE  
D67B 217ED6  LXI  H,LOADD  ;HIGH ADDRESS
D67C 3ABD6  JMP  LITES  ;RETURN
D67D 7B  LOADD: MOV  A,E  ;LOW ADDRES TO LITES
D67F 3ABD6  JMP  LITES  ;RETURN
D680 79  TPAT: MOV  A,C  ;TEST PATTERN TO LITES
D683 3ABD6  JMP  LITES  ;RETURN
D685 7B  ACTDAT: MOV  A,B  ;ACTUAL DATA TO LITES
D688 3ABD6  JMP  LITES  ;START OVER

D693 EB  PHASE II ERROR  
D694 82  ERR2: XCHG  
D695 47  SAVE BAD ADDRESS
D696 44A  ADD  D  ;SAVE BAD ADDRESS
D699 3EF2  MVI  A,OF2H  ;RETURN
D69C C3ABD6  JMP  LITES

D69F EB  PHASE III ERROR  
D6A0 83  ERR3: XCHG  ;SAVE BAD ADDRESS
D6A1 47  ADD  E  ;SAVE BAD ADDRESS
D6A2 48  MOV  B,A  ;SAVE BAD ADDRESS
D6A3 3EF3  MOV  C,E  ;SAVE BAD ADDRESS
D6A5 2177D6  LXI  H,COMERR  ;RETURN
D6A8 C3ABD6  JMP  LITES

D6AB 2F  LITES ROUTINE  
D6A4 C3ABD6  JMP  LITES  ;ENTER WITH RETURN IN REG H&L
D6AC D3FF  OUT  OFFH  ;DATA FOR LITES IN A
D6AE F5  SPHL  ;OUTPUT LITES
D6AF DBFF  ;SAVE RETURN IN SP
D6BB 67  MOV  H,A  ;READ SENSE SWITCHES
D6BC DBFF  ;SAVE IN H  
D6BD 3C  IN  OFFH  ;READ SWITCHES
D6BE 4C  JZ  OFC18H  ;SEE IF THEY CHANGED
D6BF 2118FC  JZ  LP7  ;DELAY LOOP
D6C0 AF  INX  H  ;READ RETURN BACK TO H & L
D6C1 2100D6  LXI  H,0  ;RETURN
D6C4 39  DAD  SP
D6C5 E9  PCHL  ;RETURN
; 24K MINI-MEMORY TEST
; PROM VERSION

; BRIAN DONLAN

D700 F3 ENTER2: DI
D701 3EFE MVI A, 0FEH
D702 210000 LXI H, 0000H ; START ADDRESS
D703 77 LP12: MOV M, A ; STORE TEST PATTERN IN MEM.
D704 46 MOV B, M ; READ BACK TO B
D705 B5 CMP B ; COMPARE FOR OK
D706 C26D7 JNZ ERR12 ; JUMP IF ERROR
D707 D3FF OUT OFFH ; OUTPUT PHASE I LITES
D708 210000 LXI H, 0000H ; OUTPUT PHASE I LITES
D709 3C INR A ; NEW TEST PATTERN
D70A C209D7 JNZ LP12
D70B 1100AO LXI D, 0A000H ; STOP ADDRESS
D70C 19 DAD D ; ADD TWO'S COMPLIMENT
D70D 224D7 JNC LP22
D70E 3EFC MVI A, 0FCF
D70F D3FF OUT OFFH
D710 210000 LXI H, 0000H ; LOW ADDRESS TO MEM
D711 23 INX H
D712 1100AO LXI D, 0A000H ; STOP ADDRESS
D713 19 DAD D
D714 224D7 JNC LP32
D715 210000 LXI H, 0000H ; READ MEMORY
D716 3EFC MVI A, 0FCF
D717 210000 LXI H, 0000H ; READ MEMORY
D718 94 SUB L ; COMPARE
D719 C287D7 JNZ ERR22 ; JUMP IF ERROR
D71A 23 INX H
D71B 1100AO LXI D, 0A000H
D71C 19 DAD D
D71D 224D7 JNC LP32
D71E 3EFC MVI A, 0FCF
D71F D3FF OUT OFFH
D720 210000 LXI H, 0000H ; STORE HIGH ADDRESS IN ALL MEM
D721 75 MOV M, L
D722 23 INX H
D723 1100AO LXI D, 0A000H
D724 19 DAD D
D725 224D7 JNC LP52
D726 210000 LXI H, 0000H ; READ MEMORY
D727 75 MOV M, L
D728 95 SUB L ; COMPARE
D729 23 INX H
D72A 1100AO LXI D, 0A000H
D72B 19 DAD D
D72C 224D7 JNC LP52
D72D 210000 LXI H, 0000H ; READ MEMORY
D72E 75 MOV M, L
D72F 95 SUB L ; COMPARE
D730 224D7 JNC LP52

D731 3EFC MVI A, 0FCF
D732 210000 LXI H, 0000H ; STORE HIGH ADDRESS IN ALL MEM
D733 75 MOV M, L
D734 95 SUB L ; COMPARE
D735 23 INX H
D736 1100AO LXI D, 0A000H
D737 19 DAD D
D738 224D7 JNC LP52
D739 210000 LXI H, 0000H ; READ MEMORY
D73A 75 MOV M, L
D73B 95 SUB L ; COMPARE
D73C 23 INX H
D73D 1100AO LXI D, 0A000H
D73E 19 DAD D
D73F EE XCHG
D740 19 DAD D
D741 224D7 JNC LP52

D742 210000 LXI H, 0000H ; READ MEMORY
D743 75 MOV M, L
D744 95 SUB L ; COMPARE
D745 23 INX H
D746 1100AO LXI D, 0A000H
D747 EE XCHG
D748 19 DAD D
D749 224D7 JNC LP52
D74A 210000 LXI H, 0000H ; READ MEMORY
D74B 75 MOV M, L
D74C 95 SUB L ; COMPARE
D74D 23 INX H
D74E 1100AO LXI D, 0A000H
D74F EE XCHG
D750 19 DAD D
D751 224D7 JNC LP52
D752 210000 LXI H, 0000H ; READ MEMORY
D753 75 MOV M, L
D754 95 SUB L ; COMPARE
D755 23 INX H
D756 1100AO LXI D, 0A000H
D757 EE XCHG
D758 19 DAD D
D759 224D7 JNC LP52
D75A 210000 LXI H, 0000H ; READ MEMORY
D75B 75 MOV M, L
D75C 95 SUB L ; COMPARE
D75D 23 INX H
D75E 1100AO LXI D, 0A000H
D75F EE XCHG
D760 19 DAD D
D761 EE XCHG
D762 D255D7 JNC LP62

136
### System Configuration Interface

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>F600</td>
<td>SCP</td>
<td>EQU $0F600H</td>
<td>SCP</td>
</tr>
<tr>
<td>F600</td>
<td>IOTAB</td>
<td>EQU SCP</td>
<td>SCP</td>
</tr>
<tr>
<td>F633</td>
<td>ADDCS</td>
<td>EQU SCP+48</td>
<td>SCP+48</td>
</tr>
<tr>
<td>F636</td>
<td>ADIOD</td>
<td>EQU SCP+51</td>
<td>SCP+51</td>
</tr>
<tr>
<td>F639</td>
<td>ADUST</td>
<td>EQU SCP+54</td>
<td>SCP+54</td>
</tr>
</tbody>
</table>

### ASCII Characters

<table>
<thead>
<tr>
<th>ASCII Code</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>00DCR</td>
<td>EQU $0DH</td>
<td>SCP+57</td>
</tr>
<tr>
<td>004A</td>
<td>EQU $0AH</td>
<td>SCP+51</td>
</tr>
</tbody>
</table>

### Externally Referenced Subroutine Jump Table

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>F824</td>
<td>$0E01</td>
<td>BEGIN; MVI</td>
<td>C, 1</td>
</tr>
<tr>
<td>F826</td>
<td>C33F00</td>
<td>LXI D, OFFH</td>
<td></td>
</tr>
<tr>
<td>F829</td>
<td>C3EAF0</td>
<td>JMP INITA</td>
<td></td>
</tr>
<tr>
<td>F82C</td>
<td>EB</td>
<td>XCHG</td>
<td></td>
</tr>
<tr>
<td>F82D</td>
<td>0615</td>
<td>MVI B, ENDX-EXIT-1</td>
<td></td>
</tr>
<tr>
<td>F82E</td>
<td>1176F2</td>
<td>LXI D, ENDX-1</td>
<td></td>
</tr>
<tr>
<td>F832</td>
<td>1B</td>
<td>DCX D</td>
<td></td>
</tr>
<tr>
<td>F833</td>
<td>1A</td>
<td>LDAX D</td>
<td></td>
</tr>
<tr>
<td>F834</td>
<td>28</td>
<td>DCX H</td>
<td></td>
</tr>
<tr>
<td>F835</td>
<td>77</td>
<td>MOV M, A</td>
<td></td>
</tr>
<tr>
<td>F836</td>
<td>35</td>
<td>DCR B</td>
<td></td>
</tr>
</tbody>
</table>
F037 C232F0 JNZ BG1
F03A F9 SPHL
F03B CD39F6 CALL ADIUST
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 ADIOS
F04C 3600 MOV MOO
F04E 2140F7 LSI H, TITLE
F051 CD1EF1 CALL STRNG
F054 B7 ORA A

; COMMAND RETURN POINT

F055 D266F0 CMNDR: JNC START

; ERROR RETURN

F058 CD33F6 LER: CALL ADSCR
F05A 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 CRLF
F067 CD46F1 CALL CRLF
F06A 0E2E MVI C,'.'
F06C CD1F0 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 2E5 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 99 PCHL
| BLK: | MV1 | C: |,
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F0D1 CDEFF0</td>
<td>CO</td>
<td>CALL</td>
<td>IOBR</td>
</tr>
<tr>
<td>F0D4 0110</td>
<td>DB</td>
<td>1,10H</td>
<td></td>
</tr>
<tr>
<td>F0D6 CDEFF0</td>
<td>CI</td>
<td>CALL</td>
<td>IOBR</td>
</tr>
<tr>
<td>F0D9 0108</td>
<td>DB</td>
<td>1,0</td>
<td></td>
</tr>
<tr>
<td>F0DB CDEFF0</td>
<td>CSTS</td>
<td>CALL</td>
<td>IOBR</td>
</tr>
<tr>
<td>F0DE 0108</td>
<td>DB</td>
<td>1,0</td>
<td></td>
</tr>
<tr>
<td>F0E0 CDEFF0</td>
<td>RI</td>
<td>CALL</td>
<td>IOBR</td>
</tr>
<tr>
<td>F0E3 0418</td>
<td>DB</td>
<td>4,18H</td>
<td></td>
</tr>
<tr>
<td>F0E5 CDEFF0</td>
<td>PO</td>
<td>CALL</td>
<td>IOBR</td>
</tr>
<tr>
<td>F0EB 0320</td>
<td>DB</td>
<td>3,20H</td>
<td></td>
</tr>
</tbody>
</table>
FOEA CDEFF0  LO: CALL  IOBR
FOED 0228  DB 2,28H

F0EF E3  IOBR: XTHL
F0F0 CS  PUSH  B
F0F1 46  MOV  B,M
F0F2 23  INX  H
F0F3 4E  MOV  C,M
F0F4 CD36F6  CALL ADI0B
F0F7 7E  MOV  A,M
F0F8 0F  RRC
F0F9 07  RLC
F0FA 07  RLC
F0FB 05  DCR  B
F0FC C2F9F0  JNZ  IOBJ
F0FF E636  ANI  6
F131 81  ADD  C
F132 4F  MOV  C,A
F133 2100F6  LXI  H,10TAB
F136 09  DAD  B
F137 7E  MOV  A,M
F138 23  INX  H
F139 66  MOV  H,M
F13A 6F  MOV  L,A
F13B C1  POP  B
F13C E3  XTHL
F13D C9  RET

I0CHK:
F10E E5  PUSH  H
F10F CD36F6  CALL ADI0B
F112 7E  MOV  A,M
F113 E1  POP  H
F114 C9  RET

I0SET:
F115 E5  PUSH  H
F116 F5  PUSH PSW
F117 CD36F6  CALL ADI0B
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  JMP CO
F128 CD1F0  CALL CO
F12B 23  INX  H
F12C C31EF1  JMP STRNG

141
<table>
<thead>
<tr>
<th>TI:</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F120 C0D60F</td>
<td>CALL</td>
<td>CI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F132 E67F</td>
<td>ANI</td>
<td>7FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F134 C5</td>
<td>PUSH</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F135 4F</td>
<td>MOV</td>
<td>C,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F136 C0D10F</td>
<td>CALL</td>
<td>CO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F139 79</td>
<td>MOV</td>
<td>A,C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F13A 41</td>
<td>POP</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F13B C9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F13C E60F</td>
<td>ANI</td>
<td>0FH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F13E C69F</td>
<td>ADI</td>
<td>98H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F140 27</td>
<td>DAA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F141 C0E40</td>
<td>ACI</td>
<td>40H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F143 27</td>
<td>DAA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F144 4F</td>
<td>MOV</td>
<td>C,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F145 C9</td>
<td></td>
<td>RET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F146 0E2D</td>
<td>MVI</td>
<td>C,CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F148 C0D10F</td>
<td>CALL</td>
<td>CO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F14B 0E0A</td>
<td>MVI</td>
<td>C,LF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F14D C0D10F</td>
<td>JMP</td>
<td>CO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F150 0E01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F152 2100000</td>
<td>LXI</td>
<td>H,00000H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F155 C0D20F</td>
<td>CALL</td>
<td>TI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F158 47</td>
<td>MOV</td>
<td>B,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F159 C0D0F1</td>
<td>CALL</td>
<td>NIBBL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F15C 0D68F1</td>
<td>JC</td>
<td>EX2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F15F 29</td>
<td>DAD</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F160 29</td>
<td>DAD</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F161 29</td>
<td>DAD</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F162 29</td>
<td>DAD</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F163 B5</td>
<td>ORA</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F164 6F</td>
<td>MOV</td>
<td>L,A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F165 03F5F1</td>
<td>JMP</td>
<td>EX0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F168 4E3</td>
<td>XTHL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F169 4E5</td>
<td>PUSH</td>
<td>H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F16A 78</td>
<td>MOV</td>
<td>A,B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F16B C0D5F1</td>
<td>CALL</td>
<td>P2C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F16C 0D76F1</td>
<td>JNC</td>
<td>EX3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F171 0D</td>
<td>DCR</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F172 C025F3</td>
<td>JNZ</td>
<td>LER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F175 C9</td>
<td>RET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F176 C025F3</td>
<td>JNZ</td>
<td>LER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F179 0D</td>
<td>DCR</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17A C025F1</td>
<td>JNZ</td>
<td>EXPR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17D C9</td>
<td>RET</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17E 0E31</td>
<td>MVI</td>
<td>C,1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F180 2100000</td>
<td>LXI</td>
<td>H,00000H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F183 C035F1</td>
<td>JMP</td>
<td>EX1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

142
F186 23 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 MVI A,D
F18E 9C SBB H
F18F C9 RET

F190 7C MOV A,H
F191 CD95F1 CALL LBYTE
F194 7D MOV A,L

F195 F5 PUSH PSW
F196 0F RRC
F197 0F RRC
F198 0F RRC
F199 0F RRC
F19A CD9EF1 CALL HXD
F19D F1 POP PSW

F19E CD9CF1 CALL CONV
F1A1 C3DIF0 JMP CO

F1A4 0604 MVI B,A
F1A6 0E00 MVI C,O
F1A8 CDE5F0 CALL PO
F1AB 05 DCR B
F1AC C2A6F1 JNZ LEADS
F1AF B7 ORA A
F1B0 C9 RET

F1B1 E5 PUSH H
F1B2 D5 PUSH D
F1B3 CD33F6 CALL ADSCR
F1B6 EB XCHG
F1B7 210000 LXI H,0
F1BA 24 INC H,0
F1BB 7E MOV A,M
F1BC 2F CMA
F1BD 77 MOV M,A
F1BE BE CMP M
F1BF 2F CHA
F1C0 77 MOV M,A
F1C1 CABAFL JMP MEM0
F1C4 2B DCX H
F1C5 44 MOV B,H
F1C6 7C MOV A,H
F1C7 BA CMP D
FIC8 3EC0  MVI A, $00
F1CA D1  POP D
F1CB E1  POP H
F1CC C8  RZ
F1CD 3EFF  MVI A, $FF
F1CF C9  RET

F1D0 D630  SUI '0'
F1D2 D8  RC
F1D3 6E9  ADI '0'-'G'
F1D5 D8  RC
F1D6 C606  ADI 6
F1D8 F2DEF1  JP NI0
F1DB C607  ADI 7
F1DD D8  RC
F1DE C60A  NI0: ADI 10
F1E0 37  ORA A
F1E1 C9  RET

F1E2 E2EFF1  CALL TI
P2G:
P135 FE20  CPI '
P1E7 C8  RZ
P1E8 FE2C  CPI '
P1EA C8  RZ
P1EB FE0D  CPI CR
P1ED 37  STC
P1EE C8  RZ
P1EF 3F  CMG
P1F0 C9  RET

BREAKPOINT ENTRY POINT

F1F1 E5  RESTRT: 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
F1F9 19  DAD D
F1FC E8  XCHG
F1FD 210A00  LXI H,0000AH
F200 39  DAD SP
F201 0604  MVI B,4
F203 EB  XCHG
F204 2B  RST9: 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 RST9
F20B C1  POP B
<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
<th>Decimal Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F20E 08</td>
<td>DCX</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F20F F9</td>
<td>SPHL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F210 211400</td>
<td>LXI</td>
<td>H_, TLOC</td>
<td></td>
</tr>
<tr>
<td>F213 F39</td>
<td>DAD</td>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>F214 7E</td>
<td>MOV</td>
<td>A, M</td>
<td></td>
</tr>
<tr>
<td>F215 H1</td>
<td>SUB</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F216 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F217 C21FF2</td>
<td>JNZ</td>
<td>RST1</td>
<td></td>
</tr>
<tr>
<td>F21A 7E</td>
<td>MOV</td>
<td>A, M</td>
<td></td>
</tr>
<tr>
<td>F21B H0</td>
<td>SUB</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F21C CA2DF2</td>
<td>JZ</td>
<td>RST3</td>
<td></td>
</tr>
<tr>
<td>F21F 23</td>
<td>INX</td>
<td>H</td>
<td>RST1:</td>
</tr>
<tr>
<td>F220 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F221 7E</td>
<td>MOV</td>
<td>A, M</td>
<td></td>
</tr>
<tr>
<td>F222 H1</td>
<td>SUB</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>F223 C22CF2</td>
<td>JNZ</td>
<td>RST2</td>
<td></td>
</tr>
<tr>
<td>F226 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F227 7E</td>
<td>MOV</td>
<td>A, M</td>
<td></td>
</tr>
<tr>
<td>F228 H0</td>
<td>SUB</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F229 CA2DF2</td>
<td>JZ</td>
<td>RST3</td>
<td></td>
</tr>
<tr>
<td>F22C 03</td>
<td>INX</td>
<td>B</td>
<td>RST2:</td>
</tr>
<tr>
<td>F22D 210F00</td>
<td>LXI</td>
<td>H_, TLOC</td>
<td>RST3:</td>
</tr>
<tr>
<td>F230 F39</td>
<td>DAD</td>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>F231 7E</td>
<td>MOV</td>
<td>M, E</td>
<td></td>
</tr>
<tr>
<td>F232 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F233 H1</td>
<td>MOV</td>
<td>M, D</td>
<td></td>
</tr>
<tr>
<td>F234 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F235 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F236 71</td>
<td>MOV</td>
<td>M, C</td>
<td></td>
</tr>
<tr>
<td>F237 23</td>
<td>INX</td>
<td>K</td>
<td></td>
</tr>
<tr>
<td>F238 70</td>
<td>MOV</td>
<td>M, B</td>
<td></td>
</tr>
<tr>
<td>F239 C5</td>
<td>PUSH</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F23A 219DF0</td>
<td>LXI</td>
<td>H_, ERM</td>
<td></td>
</tr>
<tr>
<td>F23D CD1EF1</td>
<td>CALL</td>
<td>STRNG</td>
<td></td>
</tr>
<tr>
<td>F240 E1</td>
<td>POP</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F241 CD90F1</td>
<td>CALL</td>
<td>LADR</td>
<td></td>
</tr>
<tr>
<td>F244 211400</td>
<td>LXI</td>
<td>H_, TLOC</td>
<td></td>
</tr>
<tr>
<td>F247 F39</td>
<td>DAD</td>
<td>SP</td>
<td></td>
</tr>
<tr>
<td>F248 1602</td>
<td>MVI</td>
<td>D, 2</td>
<td></td>
</tr>
<tr>
<td>F24A 4E</td>
<td>MOV</td>
<td>C, M</td>
<td>RST4:</td>
</tr>
<tr>
<td>F24B 3600</td>
<td>MVI</td>
<td>M, 0</td>
<td></td>
</tr>
<tr>
<td>F24D 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F24E 46</td>
<td>MOV</td>
<td>B, M</td>
<td></td>
</tr>
<tr>
<td>F24F 3603</td>
<td>MVI</td>
<td>M, 0</td>
<td></td>
</tr>
<tr>
<td>F251 23</td>
<td>INX</td>
<td>H</td>
<td></td>
</tr>
<tr>
<td>F252 79</td>
<td>MOV</td>
<td>A, C</td>
<td></td>
</tr>
<tr>
<td>F253 B0</td>
<td>ORA</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F254 CA59F2</td>
<td>JZ</td>
<td>RST5</td>
<td></td>
</tr>
<tr>
<td>F257 7E</td>
<td>MOV</td>
<td>A, M</td>
<td></td>
</tr>
<tr>
<td>F258 02</td>
<td>STAX</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>F259 23</td>
<td>INX</td>
<td>H</td>
<td>RST5:</td>
</tr>
<tr>
<td>F25A 15</td>
<td>DCR</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>F25B C24AF2</td>
<td>JNZ</td>
<td>RST4</td>
<td></td>
</tr>
<tr>
<td>F25E C366F0</td>
<td>JMP</td>
<td>START</td>
<td></td>
</tr>
</tbody>
</table>
SCRATCHPAD TEMPLATE

F261 D1  EXIT:  POP  D
F262 E1  POP  B
F263 F1  POP  PSW
F264 E1  POP  H
F265 F9  SPHL
F266 FB  EI
F267 210000  LXI  H, 0
F268 HLX  EQU  $-2
F26A C30000  JMP  8
F26B PCX  EQU  $-2
F26D 0000  TIA:  DV  0  JMP  1 ADDR
F26E 03  DB  0  JMP  1 INST
F270 0000  LV  0  JMP  2 ADDF
F272 00  DB  0  JMP  2 INST
F273 0000  D4  0  JVIDEO PTR
F275 00  DB  0  JVIDEO HOLD
F276 00  DS  0  JNOPYT
ENDX:

J005  ALOC  EQU  5
J003  BLOC  EQU  3
J002  CLOC  EQU  2
J001  DLOC  EQU  1
J000  ELOC  EQU  0
J004  FLOC  EQU  4
J010  HLOC  EQU  Hlx-EXIT+9
J00F  LLOC  EQU  Hlx-EXIT+8
J013  PLOC  EQU  PCX-EXIT+9
J007  SLOC  EQU  7
J014  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 C28EF2  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 DCR B
F2AC FA84F2 JM AS3
F2AF 29 DAD H
F2B0 29 DAD H
F2B1 C3ABF2 JMP AS2
F2B4 EB XCHG
F2B5 CD36F6 CALL ADIOB
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

I 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
F2DB 8E78 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 CD4AF1 CALL LEADS
F2EE B3C CALL MVI C,3CH
F2F0 CDE5F0 CALL PO
F2F3 4B MOV C,E
F2F4 CD11F3 CALL PHL
F2F7 7C MOV A,H
F2FB 85 ADD L:
F2FF 57 MOV D,A
F2FA 4E MOV C,M
F2FB 23 INX H
F2FC 7A MOV A,D
F2FD B1 ADD C
F2FE 57 MOV D,A
F2FF CDE5F0 CALL PO
F302 1D DCR E
F303 C2FAF2 JNZ E3
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 D10 CALL CRLF
F324 CD90F1 CALL LADR
F327 CUCFF0 D11 CALL BLK
F32A 7E MOV A,M
F32B CD95F1 CALL BYTEZ
F32E CD86F1 CALL HILO
F331 3F CMC
F332 D0 RNC
F333 7D MOV A,L
F334 E60F ANI 0FH
F336 C227F3 JNZ D1
F339 C321F3 JMP D10

FILL COMMAND

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

148
GOTO COMMAND
F34C E1       GOTO:  POP  H
F34D CDE2F1   CALL  PCHK
F350 DA98F3   JC   GO3
F353 GA72F3   JZ   GO6
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   GO3
F367 3EC3     MVI  A,(JMP 0)
F369 320800    STA  B
F36C 21F1F1   LXI  H,RESTRT
F372 1682     SHLD  9
F374 211400    LXI  H,TLOC
F377 39       DAD  SP
F378 E5       PUSH  H
F379 CD50F1   CALL  EXPR1
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   GO2
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       MOV  A,E
F38F FE0D     CPI  CR
F391 CA98F3   JZ   GO3
F394 15       DCR  D
F395 G278F3   JNZ  G01
; MOVE COMMAND
F400 0C       MOVE:    INR    C
F400 CD52F1   CALL    EXPR
F410 C1       POP     B
F411 D1       POP     D
F412 E1       POP     H
F413 7E       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 CDSOOF1  READ:  CALL    EXPR
F426 E1       POP     H
F427 CDE0F0   RED0:  CALL    RI
F42A D8       RC
F42B E67F     ANI     7FH
F42D D63A     SUI    '1'

151
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 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 CDD4F1 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

<table>
<thead>
<tr>
<th>Code</th>
<th>Action</th>
<th>Code</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>F489</td>
<td>SUBS: CALL</td>
<td>F49C</td>
<td>CALL P2C</td>
</tr>
<tr>
<td>F48F</td>
<td>SUBS: CALL</td>
<td>F490</td>
<td>POP H</td>
</tr>
<tr>
<td>F491</td>
<td>SU: MOV A,M</td>
<td>F495</td>
<td>MVI C,-'</td>
</tr>
<tr>
<td>F492</td>
<td>CALL LBYTE</td>
<td>F497</td>
<td>CALL CO</td>
</tr>
<tr>
<td>F49A</td>
<td>CALL PCHK</td>
<td>F49D</td>
<td></td>
</tr>
<tr>
<td>F49E</td>
<td></td>
<td>F49F</td>
<td></td>
</tr>
<tr>
<td>F4A2</td>
<td></td>
<td>F4A3</td>
<td></td>
</tr>
<tr>
<td>F4A6</td>
<td></td>
<td>F4A7</td>
<td></td>
</tr>
<tr>
<td>F4A8</td>
<td></td>
<td>F4A9</td>
<td></td>
</tr>
<tr>
<td>F4AA</td>
<td></td>
<td>F4AC</td>
<td></td>
</tr>
<tr>
<td>F4AD</td>
<td></td>
<td>F4AE</td>
<td></td>
</tr>
</tbody>
</table>

WRITE COMMAND

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F4A1</td>
<td>WRITE: CALL</td>
<td>F4B4</td>
<td>CALL CRLF</td>
<td>F4B7</td>
<td>POP D</td>
<td>F4B8</td>
<td>POP H</td>
</tr>
<tr>
<td>F4B9</td>
<td>WRITE: MOV A,D</td>
<td>F4B9</td>
<td>MOV A,D</td>
<td>F4B3</td>
<td>ORA E</td>
<td>F4B8</td>
<td>JNZ W0</td>
</tr>
<tr>
<td>F4BA</td>
<td></td>
<td>F4BE</td>
<td>CALL PEOL</td>
<td>F4C1</td>
<td>MVI C,-'</td>
<td>F4C3</td>
<td>CALL PO</td>
</tr>
<tr>
<td>F4C6</td>
<td></td>
<td>F4C7</td>
<td></td>
<td>F4C8</td>
<td>CALL PBYTE</td>
<td>F4C9</td>
<td></td>
</tr>
<tr>
<td>F4C7</td>
<td></td>
<td>F4C9</td>
<td></td>
<td>F4CE</td>
<td>MVI A,1</td>
<td>F4DE</td>
<td></td>
</tr>
<tr>
<td>F4D0</td>
<td></td>
<td>F4DE</td>
<td></td>
<td>F4D5</td>
<td>CALL PBYTE</td>
<td>F4DF</td>
<td></td>
</tr>
<tr>
<td>F4D3</td>
<td></td>
<td>F4D4</td>
<td></td>
<td>F4D8</td>
<td>JMP NULL</td>
<td>F4DB</td>
<td></td>
</tr>
<tr>
<td>F4D5</td>
<td></td>
<td>F4DC</td>
<td></td>
<td>F4DB</td>
<td>MOV A,E</td>
<td>F4DD</td>
<td></td>
</tr>
<tr>
<td>F4DE</td>
<td></td>
<td>F4DF</td>
<td></td>
<td>F4DH</td>
<td>MOV A,D</td>
<td>F4E0</td>
<td></td>
</tr>
</tbody>
</table>

153
<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Address</th>
<th>Instruction</th>
<th>Address</th>
<th>Instruction</th>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>F4E0 7B</td>
<td>WR10: MOV A, E</td>
<td>F4E1 95</td>
<td>SUB L</td>
<td>F4E2 4F</td>
<td>MOV C, A</td>
<td>F4E3 7A</td>
<td>MOV A, D</td>
</tr>
<tr>
<td>F4E5 3F</td>
<td>CMC</td>
<td>F4E6 00</td>
<td>PUSH RNC</td>
<td>F4E7 79</td>
<td>MOV A, C</td>
<td>F4E8 E62F</td>
<td>ANI $0FH</td>
</tr>
<tr>
<td>F4EB 5F</td>
<td>PUSH D</td>
<td>F4ED 1600</td>
<td>MOV E, A</td>
<td>F4EF CD33F5</td>
<td>CALL PEOl</td>
<td>F4F2 0E3A</td>
<td>MVI D, 0</td>
</tr>
<tr>
<td>F4F7 7B</td>
<td>MOV A, E</td>
<td>F4F8 CD1CF5</td>
<td>CALL PBYTE</td>
<td>F4F9 CD17F5</td>
<td>CALL PADR</td>
<td>F4FE AF</td>
<td>XRA A</td>
</tr>
<tr>
<td>F502 7E</td>
<td>WR11: MOV A, M</td>
<td>F503 23</td>
<td>INX H</td>
<td>F504 CD1CF5</td>
<td>CALL PBYTE</td>
<td>F507 ID</td>
<td>DCR E</td>
</tr>
<tr>
<td>F50A AF</td>
<td>XRA A</td>
<td>F50B 92</td>
<td>SUB D</td>
<td>F50D CD1CF5</td>
<td>CALL PBYTE</td>
<td>F510 D1</td>
<td>POP D</td>
</tr>
<tr>
<td>F512 B4</td>
<td>ORA H</td>
<td>F513 C8</td>
<td>RZ</td>
<td>F514 C3E0F4</td>
<td>JMP WR10</td>
<td>F517 7C</td>
<td>PADR: MOV A, H</td>
</tr>
<tr>
<td>F51C F5</td>
<td>PBYTE: PUSH PSW</td>
<td>F51D 0F</td>
<td>RRC</td>
<td>F51E 0F</td>
<td>RRC</td>
<td>F51F 0F</td>
<td>RRC</td>
</tr>
<tr>
<td>F521 CD3CF1</td>
<td>CALL CONV</td>
<td>F524 CDE5F0</td>
<td>CALL PO</td>
<td>F527 F1</td>
<td>POP PSW</td>
<td>F528 F5</td>
<td>PUSH PSW</td>
</tr>
<tr>
<td>F52C CDE5F0</td>
<td>CALL PO</td>
<td>F52F F1</td>
<td>POP PSW</td>
<td>F530 82</td>
<td>ADD D</td>
<td>F531 57</td>
<td>MOV D, A</td>
</tr>
<tr>
<td>F533 0E0D</td>
<td>PEOl: MVI C, CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
F535  CDE5F0
F538  0E9A
F53A  C3E5F0
               CALL  P0
               MVI  C,LF
               JMP  PC

               REGISTER COMMAND

F53D  CD2FF1  X1:  CALL  TI
F540  21CCF5
F543  FE0D
F545  CA9FF5
F548  47
F549  BE  X0:
F54A  CA57F5
F54D  7E
F54E  17
F54F  D8
F550  23
F551  23
F552  23
F553  78
F554  C349F5
F557  CDCCF0  X1:  CALL  BLK
F55A  23  X2:
F55B  7E
F55C  EB
F55D  6F
F55E  2600
F560  39
F561  EB
F562  23
F563  46
F564  23
F565  1A
F566  CD95F1
F569  05
F56A  CA72F5
F56D  1B
F56E  1A
F56F  CD95F1
F572  04  X3:
F573  0E2D
F575  CDD1F0
F578  CDE2F1
F57B  3F
F57C  D0
F57D  CA95F5
F580  E5
F581  C5
F582  CD7EF1
F585  E1
F586  F1
F587  C5
F588  F5
               MOV  B,A
               JZ  X6
               CMP  M
               JZ  X1
               MOV  A,M
               RAL
               RC
               INX  H
               INX  H
               INX  H
               MOV  A,B
               JMP  X0
               CALL  BLK
               INX  H
               MOV  A,M
               MOV  L,A
               MOV  H,0
               DAD  SP
               XCHG
               INX  H
               MOV  B,M
               INX  H
               LDAX  D
               CALL  LBYTE
               DCR  B
               JZ  X3
               DCX  D
               LDAX  D
               CALL  LBYTE
               INR  B
               MVI  C,'-'
               CALL  CO
               CALL  PCHK
               GMC
               RNC
               JZ  X5
               PUSH  H
               PUSH  B
               CALL  EXF
               POP  H
               POP  P5W
               PUSH  B
               PUSH  P5W

155
F589 7D: MOV A,L
F58A 12: STAX D
F58B C1: POP B
F58C 05: DCR B
F58D CA93F5: JZ X4
F58E 13: INX D
F58F 7C: MOV A,H
F590 12: STAX D
F591 C1: POP B
F592 E1: POP H
F593 7E: MOV A,M
F594 B7: ORA A
F595 F8: RM
F596 78: MOV A,B
F597 FE8D: CPI CR
F598 C8: RZ
F599 G35AF5: JMP X2
F59A 046F1: CALL CRLF
F59B CDCFF0: CALL BLK
F59C 72: MOV A,M
F59D 23: INX H
F59E B7: ORA A
F59F F8: RM
F59G 4F: MOV C,A
F59H GDD1F0: CALL CO
F59I 0E3D: MVI C, "="
F59J GDD1F2: CALL CO
F59K 7E: MOV A,M
F59L 23: INX H
F59M EB: XCHG
F59N 6F: MOV L,A
F59O 260J: MVI H,0
F59P 39: DAD SP
F59Q EB: XCHG
F59R 46: MOV B,M
F59S 23: INX H
F59T 1A: LDAX D
F59U CD95F1: CALL LBYTE
F59V 35: DCR D
F59W CAA2F5: JZ X7
F59X 1B: DCX D
F59Y 1A: LDAX D
F59Z CD95F1: CALL LBYTE
F5A0 C3A2F5: JMP X7
F5A1 4107A1: ACTBL: DB 'A', ALOC+2, 1
F5A2 420501: 'B', BLOC+2, 1
F5A3 430401: 'C', CLOC+2, 1
F5A4 440301: 'D', DLOC+2, 1
F5A5 450201: 'E', ELOC+2, 1
F5A6 460101: 'F', FLOC+2, 1
F5A7 481201: 'H', HLOC+2, 1

156
<table>
<thead>
<tr>
<th>Location</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Value 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>F5E1 4C1101</td>
<td>DB 'L',</td>
<td>LLOC+2, 1</td>
<td></td>
</tr>
<tr>
<td>F5E4 4D1202</td>
<td>DB 'M',</td>
<td>HLOC+2, 2</td>
<td></td>
</tr>
<tr>
<td>F5E7 501502</td>
<td>DB 'P',</td>
<td>PLOC+2, 2</td>
<td></td>
</tr>
<tr>
<td>F5EA 530902</td>
<td>DB 'S',</td>
<td>SLOC+2, 2</td>
<td></td>
</tr>
<tr>
<td>F5ED FF</td>
<td>DB -1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ORG 0F600H

I

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

IOTAB1:

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

F600 A6F6  CSTAB1  DW  TTST  10
F602 7FF6  DW  KYST  11
F604 7FF6  DW  KYST  12
F606 7FF6  DW  KYST  13

I
CONSOLE INPUT
I
I
RETURN CONSOLE INPUT CHARACTER
I
IN REGISTER A.

F608 A6F6  CITAB1  DW  TTI  10
F60A 65F6  DW  KYBD  11
F60C 66F6  DW  KYBD  12
F60E 66F6  DW  KYBD  13

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

F610 A7F6  COTAB1  DW  TTO  10
F612 B7F6  DW  TTO  11
F614 D4F6  DW  THRMB  12
F616 59F6  DW  CRT  13

I
READER INPUT
I
I
RETURN READER INPUT BYTE IN
I
REGISTER A, CARRY OFF, SET
I
CARRY IF NO BYTE AVAILABLE.
(Continued from previous page)

```
F618 C2F6
F61A 87F6
F61C 66F6
F61E F0B8
RITAB: DW TTR $10
DW RDR $11
DW KYBD $12
DW 0B8F0H $13 DISK READ

; PUNCH OUTPUT

; OUTPUT BYTE IN REGISTER C
; TO PUNCH DEVICE.

F620 87F6
F622 DFF6
F624 59F6
F626 73B9
POTAB: DW TTO $10
DW PUNCH $11
DW CRT $12
DW 0B973H $13 DISK WRITE

; LISTING OUTPUT

; OUTPUT BYTE IN REGISTER C
; TO LISTING DEVICE.

F628 B7F6
F62A 59F6
F62C D4F6
F62E B7F6
LOTAB: DW TTO $10
DW CRT $11
DW THRM $12
DW TTO $13

; 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 INSTRUCTION;
; RETURN IS DONE BY PLACING THE RETURN
; ADDRESS IN H,L AND EXECUTING A PCHL INST.

F630 C33DF6
ADSCS: JMP ADS2
```

159
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.

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.

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

MOV A,C   JCHECK FOR NULL
ORA 'A
RZ
PUSH H
CALL ADIOB
DCX H
DCX H
DCX H
JMP $F784H

KEYBOARD DRIVER

IN 2
ANI 1
JNZ KYBD
IN 3
ANI 7FH
CPI 61H
JC KBI
CPI 7AH-1
JNC KBI
ANI $DFH
IDELET ONE BIT
OR A
RET

KEYBOARD STATUS DRIVER

IN 2
ANI 1
SUI 1
SBB A
RET

READER DRIVER

PUSH H
LXI H, 0
IN 4
ANI 1
JZ RD2
DCX H
MOV A, H
ORA L
JNZ RD
STC
POP H
RET
RD2: IN 5
ORA A
POP H
RET

TELETYPE STATUS DRIVER

F6A0 DB03
F6A2 E602
F6A4 D602
F6A6 9F
F6A7 C9

TTST: IN 3
ANI 2
SUI 2
SBB A
RET

TELETYPE INPUT DRIVER

F6A8 AF
F6A9 D300
F6AB DB03
F6AD E602
F6AF CAABF6
F6B2 DB02
F6B4 E67F
F6B6 C9

TTI: XRA A
OUT 0
ANI 2
JZ TTII
IN 2
ANI 7FH
RET

TELETYPE OUTPUT DRIVER

F6B7 DB03
F6B9 E601
F6BA C87F6
F6BE 79
F6BF D302
F6C1 C9

TTO: IN 3
ANI 1
JZ TTO
MOV A8C
OUT 2
RET

TELETYPE FEADER DRIVER

F6C2 3E01
F6C4 D300
F6C6 3E23
F6C8 D302
F6CA DB00
F6CC E601
F6CE C2CAF5
F6D1 DB01
F6D3 C9

TTR: MVI A,1
OUT 0
MVI A,0
OUT 0
IN 0
ANI 1
JNZ TTR1
IN 1
RET

162
; THERMAL PRINTER DRIVER

THRM:
F6D4 D802 IN 2
F6D6 E680 ANI 80H
F6D8 C2D4F6 JNZ THRM
F6DB 79 MOV A,C
F6DC D393 OUT 3
F6DE C9 RET

; PUNCH DRIVER

PUNCH:
F6DF D804 IN 4
F6E1 E680 ANI 80H
F6E3 C2DFF6 JNZ PUNCH
F6E6 79 MOV A,C
F6E7 D395 OUT 5
F6E9 C9 RET
CP/M ASSEMBLER - VER 1.0

F800  ORG OF800H
F800 2109FB ENTRY: LXI H,MESS
F900  CD1EFO CALL STRING
F906  C321FD JMP RENT
F01E * STRING: EQU OF01EH
F809 0D0A202020MESS: DB ODH,OH,' HELLO !! YOU HAVE ENTERED THE '
F830 0A0D574F52 DB ODH,OH,' WORLD OF DIAGNOSTICS. THIS LIST WILL ACQUAINT'
F860 0D0A504F55 DB ODH,OH,' YOU WITH SOME OF THE COMMANDS OF THE DIAGNOSTIC'
F891 0D0A5394D4 DB ODH,OH,' SIMILAR TO CP/M/DDT.'
F8FC CDH4 DB ODH,OH
F8FE 0D0A202020 DB ODH,OH,' A ASSIGNs I/O DEVICES (PHYSICAL'
F92E 0D0A202020 DB ODH,OH,' TO LOGICAL DEVICE')
F956 0D0A202020 DB ODH,OH,' B DUMP MEMORY IN BINARY ON PUNCH DEVICE'
F989 0D0A202020 DB ODH,OH,' C HEXADECIMAL ARITHMETIC'
F9BB 0D0A202020 DB ODH,OH,' D DISPLAY A BLOCK OF MEMORY'
F9E6 0D0A202020 DB ODH,OH,' E FILLS A BLOCK OF MEMORY WITH A CONSTANT'
FA21 0D0A202020 DB ODH,OH,' F GO TO ADDRESS AND EXECUTE, OPTIONAL'
FA58 0D0A202020 DB ODH,OH,' G BREAK POINTS.'
FA7A 0D0A202020 DB ODH,OH,' H HELP, THIS PROGRAM'
FACD 0D0A202020 DB ODH,OH,' K COPY FROM READER TO PUNCH'
FACF 0D0A202020 DB ODH,OH,' L LOAD BINARY TAPE, OPTIONAL BIAS'
FB00 0D0A202020 DB ODH,OH,' M MOVE A BLOCK OF MEMORY TO ANOTHER LOCATION'
FB39 0D0A202020 DB ODH,OH,' N OUTPUTS 60 NULLS TO PUNCH DEVICE'
FB6D 0D0A202020 DB ODH,OH,' R LOAD A HEX TAPE FROM READER DEVICE'
FBD5 0D0A202020 DB ODH,OH,' S DISPLAY AND CHANGE ANY MEM LOCATION'
FBE4 0D0A202020 DB ODH,OH,' T TEST LIST AND EXECUTION PROGRAM'
FC08 0D0A202020 DB ODH,OH,' W DUMP MEMORY IN HEX ON PUNCH DEVICE'
FC3E 0D0A202020 DB ODH,OH,' X CPU REGISTER DISPLAY AND CHANGE',0
F021 = RENT: EQU OF021H

164
FD25 214DFD  ERROR:  LXI  H,MESS4
FD28 CD1EFD  CALL  STRING  ;PRINT ERROR ??
FD2B C30DFD  JMP  ENTRY3  ;REPRINT

FD32 FE30  NUM:  CPI  10H  0
FD35 E6DF  ANI  OFH  ;REMOVE LEAD NIBBLE
FD38 87  ADD  A  ;DOUBLE FOR TABLE LOOK UP
FD3B C10DFD  LXI  H,NUMTAB
FD3C C34FD  JMP  COMMON  ;NUMBER TABLE

FD3E 600F  LETTER:  ANI  OFH
FD40 87  ADD  A  ;DOUBLE FOR TABLE
FD41 21FFE  LXI  H,LETTab
FD44 1600  COMMON:  MOV  D,O
FD46 6F  DAD  D  ;ADD OFFSET TO TABLE ADDRESS
FD48 7E  MOV  A,M
FD4A 56  MOV  H,M
FD4B 6F  MOV  L,A
FD4C 89  PCNL  ;JUMP TO TEST PROGRAM

FD4D 00A203F20E554:  DB  ODH,OAH,'?',O
FD50 0DD444E554MESS3:  DB  ODH,OAH,'ENTER TEST ID NO. TO RUN TEST',O
FD53 0DD444E554:  DB  ODH,OAH,'ENTER CONTROL C TO RETURN TO MONITOR',O
FD56 0DD444E554MESS2:  DB  ODH,OAH,'TESTS AVAILABLE!',O
FD59 0DD4A202031:  DB  ODH,OAH,'1 - COMPREHENSIVE MEMORY TEST'
FD5C 0DD4A202032:  DB  ODH,OAH,'2 - MINI-MEMORY 0 - .1K'
FD5F 0DD4A202033:  DB  ODH,OAH,'3 - MINI-MEMORY 0 - 8K'
FE02 0DD4A202034:  DB  ODH,OAH,'4 - MINI-MEMORY 0 - 24K'
FE05 0DD4A202035:  DB  ODH,OAH,'5 - FORMATTED DISK TEST'
FE08 0DD4A202036:  DB  ODH,OAH,'6 - DISK TRACK READ'
FE0B 0DD4A202037:  DB  ODH,OAH,'7 - DISK TRACK WRITE'
FE0E 0DD4A202038:  DB  ODH,OAH,'8 - UNIBUS PORT TEST'
FE11 0DD4A202039:  DB  ODH,OAH,'9 - UNIBUS COMMUNICATION TEST'
FE14 0DD4A20203A:  DB  ODH,OAH,'A - UNIBUS SNAPSHOT'
FE17 0DD4A20203B:  DB  ODH,OAH,'B - DISPLAY TESTS'

F12F = 16FH  ;ADD MORE TO DIRECTORY HERE

FECD 25FD  NUMTAB:  DW  ERROR
FEED 00C0  DW  OCOOH  ;MEMORY TEST
FE10 00C2  DW  OCOOH  ;MINI 1K
FE13 00C6  DW  OCOOH  ;MINI 8K
FE16 00C9  DW  OCOOH  ;MINI 24K
FE1F 00CE  DW  OCEOH  ;FORMAT DISK
FE28 00CD  DW  OCOEH  ;TRK RD
FE30 0000  DW  O0000H  ;TRK WR
FE6E 00D0  DW  O1000H  ;JUMP PORT
FEFE 00D1  DW  O1100H  ;JUMP COMM
FEF1 25FD  LETTab:  DW  ERROR
FEF5 00D5  DW  ODDOH  ;SNAPSHOT
FEF6 00D6  DW  OODOH  ;DISPLAY
FEF7 25FD  DW  ERROR
FEF9 25FD  DW  ERROR
FEF0 25FD  DW  ERROR
FEF1 25FD  DW  ERROR
FEF2 25FD  DW  ERROR
FEF3 25FD  DW  ERROR
FEF4 25FD  DW  ERROR
FF01 25FD  DW  ERROR
FF03 25FD  DW  ERROR
FF05 25FD  DW  ERROR
FF07 25FD  DW  ERROR