94DIAG VERSION 2.6

USER'S GUIDES

System Industries

System Industries PB9400-9021-03

94DIAG USER'S GUIDE

VERSION 2.6

Revision Record

Revision Number	Date	Description	Affected Pages
-01	10/1/80	First released	All
-02	11/3/81	Reformat and edit Incorporates CAR #733	All
-03	12/28/81	Second Revision Version 2.6	All

PREFACE

This manual contains sufficient information to enable the user to use the 94DIAG diagnostic program to test the 9400 Disk Controller System on DEC PDP-11 computers (UNIBUS interface and CACHE interface).

The contents of this manual have been prepared based on the following assumptions of reader knowledge:

1. The reader should have a working knowledge of Digital Equipment Corporation's PDP-11 processor and peripherals.

The information in this manual is presented in six major sections. The contents of these sections are as follows:

Section 1 - Introduction: describes the conventions, commands, test summaries, error reports, CPA registers, and hardware registers of the 94DIAG diagnostic program.

Section 2 - (%DD) Drive Diagnostic: describes the capabilities of the (%DD) drive diagnostic, including applicable error report options for each test.

Section 3 - (%CD) Computer Interface Diagnostic: describes the capabilities of this diagnostic, which include .RG Register Tests, .CF CPU/Formatter Tests, and .DC CPU DMA Tests.

Section 4 - (%SD) System Diagnostic: describes the capabilities of this diagnostic, including applicable error report options for each test.

Section 5 - Bootstrap Program: describes the formats for a two-character mnemonic boot, a two-character mnemonic write tape, a two-character mnemonic write disk, and a four-character mnemonic. This section also includes a list of the routines to be used when handloading a bootstrap.

Section 6 - Error Messages: lists and describes the 437 possible error messages that may occur when using the 94DIAG diagnostic program.

Table of Contents

1.0 OVERVIEW

The 94DIAG diagnostic program is designed to test the 9400 Disk Controller System on DEC PDP-11 computers (UNIBUS interface and CACHE interface). The 94DIAG diagnostic program gives the user a great deal of flexibility in building up a string of commands or tests to be run, running the tests, and reporting the results of these tests. The entire diagnostic, a related group of tests, or a string of tests in any order can be run with a single command. The user can specify how many times each test is to be run before going on to the next test, or how many times any combination of tests is to be run before going on. The user can specify what each test is to do in case of error (e.g. pause, loop, or ignore it) and what information will be reported in case of error.

1.1 Basic Conventions

The following subsections describe the standard features of this diagnostic.

1.1.1 Command Levels

The diagnostic has two command levels. The normal command level is the one in which the user enters commands into the command string and responds to initialization queries and requests from various tests. The normal command level is indicated by a >> prompt before each line of output or input, for example:

```
>> ENTER NEW CMDS
>>
```

The second command level is the dynamic command level. This takes priority over the normal command. The system stops what it is doing on the normal command level and begins executing the dynamic command. Dynamic command level is indicated by a ** prompt before each line of output, for example:

```
** (CTL E) ERR SPEC (C)=?
```

1.1.2 Keyboard Inputs

All inputs that the operator makes to the keyboard (except dynamic commands) must be followed by a <RETURN> for them to be entered. Before the <RETURN> key is pressed, the operator may delete characters one at a time with the <RUBOUT> key. Each time a character is rubbed out, the CRT terminal will delete the character from the screen. However, a TTY terminal will print < until there are no more characters to rub out. After several rubouts, the user may view the actual contents of the line as it exists by pressing <LINE FEED>.

When the system prints a question ending with something inside parentheses, this is either a default value or an existing parameter which may be changed. If the operator presses <RETURN> without entering anything, the default will be chosen or the existing parameter will remain unchanged. In other words, by entering nothing, the operator has actually entered the value in parentheses. Examples:

```
>> 9800 CONTROLLER (Y)?     ;If your controller is a 9400,
                             enter "N" <RETURN>.  If it is a
                             9800, enter <RETURN> only.
>> STD CPA LOC (Y)?
>> RP04 REGS BASE ADDR ('176700)? =
```

In the first example, the default answer is YES. The operator must enter "N" to indicate NO. In the second example, the existing parameter is 176700 which will remain unchanged if the operator enters nothing else.

If the operator enters something unexpected, such as a negative number in the second example above, the system will print ?? and repeat the question.

1.1.3 Number Systems

All numbers printed at the console are normally printed in the decimal number system. An octal number will be preceded by an apostrophe ('), and a hexadecimal number will be preceded by a quotation mark ("). Examples:

256	Decimal
'000400	Octal
"10C4	Hexadecimal

The number system shown in the default is the same as that assumed in the reply expected from the operator. Example:

```
** (CTL P) DATA PATTERNS
CUR ('125252)? =
```

The number entered by the operator is assumed to be octal.

1.2 Initialization

Before the diagnostic can begin testing, it must know what disk system is present as well as other details about the system. It determines this in a dialog with the operator as follows:

```
>> CRT OR TTY (C)? =
```

The system needs to know the terminal type in order to supply the correct number of fill characters. The default is CRT. If using a teletype, the operator should enter "T".

```
>> DATE (M-D-Y)
>> TIME (H:M:S)
```

The user may enter date and time exactly as shown in parentheses. The time is on the 24 hour system with the hours falling between 0 and 23. The user may not wish to enter date and time. In this case he simply presses <RETURN> to go on to the next question.

```
>>9800 CONTROLLER (Y)?     ;If your controller is a 9400,
                            enter "N" <RETURN>. If it is a
                            9800, enter <RETURN> only.
>> STD CPA LOC (Y)?
```

If the operator answers YES, the next two questions are skipped. The system will assume a 9400 interrupt vector address of 000254, and a 9400 registers base address of 176700.

```
>> 9400 INT VECT ADDR ('000254)? =
>> 9400 REGS BASE ADDR ('176700)? =
```

The operator may enter the non-standard values in place of the standard ones shown in parentheses.

If the diagnostic has been loaded from a system disk, this is the time to replace the system disk with a scratch pack.

Next, the system asks the operator to set the format enable switch to the ON position, to put the STD/EXT switch to the EXT position, and to reset the controller.

```
>> SET FMT EN SWITCH ON; PUT STD/EXT SW TO EXT
>> RST CTLR, HIT <RET>
```

It then prints a summary of what drives and drive types are present and asks the operator if this is OK. If so, the initialization is complete and testing can begin. If the system cannot identify the drive type or if the summary is not satisfactory to the operator, a series of questions and answers allow the operator to enter the drive characteristics. However, it is strongly recommended that the operator determine why the system cannot identify the drive type before continuing. This may be done by checking the RPDT and RPSN registers, using the CTL X command. See Section 1.4.18.

The system automatically defaults to the upper five cylinders of each unit. It tells the operator this in a message before inviting him to enter a command string.

1.3 Normal Commands

Once the initialization dialogue is completed, the system is ready to begin testing. The operator may specify which tests are to be run, in which sequence, and how many times. The operator may specify what action is to be taken when an error occurs, what information is to be reported in case of error, and when an error summary is to be given. The operator does this by means of the command language. This is made up of commands which tell the system what to do. Most of the commands tell the system which test or tests to run, some control the sequence of tests and some control the execution of the tests and the error reporting.

A command consists of a mnemonic and two parameters. A carriage return terminates the command entry. For example:

```
>> AV,3,L
   |  | |_____What to do in case of error
   |  |       (P=Pause, L=Loop, C=Continue
   |  |
   |  |_______How many times to run the test
   |
   |__________Which test
```

This command means: run the address verify test three times. If there is an error, loop on the error until told to stop. Note that commas are used to separate the mnemonic and the parameters.

The operator does not need to enter the parameters if he wishes to use the default values. Thus:

```
>> AV
```

means: run the address verify test once, continue on error. This is equivalent to entering:

```
>> AV,1,C
```

The operator may enter only one of the parameters and take the default value for the other. For example:

```
>> AV,P
```

is equivalent to:

```
>> AV,1,P
```

Another example:

```
>> AV,12
```

is equivalent to:

```
>> AV,12,C
```

1.3.1 Primitive Commands

Many of the commands are primitive commands. That is, they refer to a single test or operation. These commands take the following form:

```
MNEMONIC,N,(ERROR SPECIFIER)
   |     |        |             P = Pause on error
   |     |        |____________ L = Loop on error
   |     |                      C = Ignore error (default)
   |     |
   |     |                      N = No. of repetitions
   |     |_____________________ 0 = Indefinite loop
   |                            1 = Default
   |                           99 = Highest value
   |
   |___________________________ Two-letter name of test
```

Primitive commands may be strung together in any order as many as desired. For example:

```
>> AV,2,C
>> BB,3,P
>> CI,4
>> CR
>> CS
>> /
```

The slash (/) is the command to begin executing the command string. It also indicates the end of the command string. When the system has finished executing the command string, it is ready to accept a new string.

In the above examples, the commands execute from first to last and then the execution ends. It is also possible to loop back on the string a specified number of times. This is done with two additional commands:

```
>> LUP,N        Loop to LPT N times
>> LPT          Loop back to here
```

Example:

```
>> AV,2,C
>> LPT
>> BB,3
>> MW,4
>> LUP,6
>> CR
>> CS
>> /
```

After running AV twice, it will run BB three times, MW four times, then repeat this six times before running CR and CS.

1.3.2 Macro Commands

The macro command is intended to test a group of similar hardware components. Each macro command in the command string automatically calls a sequence of primitive commands. Each primitive command is run once unless told to loop on error. The macro commands take the following form:

```
.MNEMONIC,N,(ERROR SPECIFIER)
|     |    |         |
|     |    |         |            P = Pause on error
|     |    |         |__________  L = Loop on error
|     |    |                      C = Ignore error (default)
|     |    |
|     |    |                      N = No. of repetitions
|     |    |____________________  0 = Indefinite
|     |                           1 = Default
|     |                          99 = Maximum
|     |
|     |_________________________  Two-letter name of macro
|
|_______________________________  Period indicates macro
```

Example:

```
>> .RG,3,C
```

This command means: run the set of register tests three times. Continue on error. It is exactly equivalent to the following string of primitive commands:

```
>> LPT
>> EK,1,C
>> EM,1,C
>> LUP,3
```

The operator may string together several macros, use loop commands, or mix in primitive commands as he chooses.

Example:

```
>> .BC,3,C
>> LPT
>> .DR,2
>> .FI,P
>> LUP,4
>> MW
>> /
```

Macro commands are for convenience only. The same operation may be performed with the appropriate string of primitive commands.

1.3.3 Global Commands

Global commands are intended to diagnose an entire hardware module. Each global command automatically calls a sequence of macro commands, each of which in turn calls a sequence of primitive commands. The global commands take the following form:

```
%MNEMONIC,N,(ERROR SPECIFIER)
|    |     |       |
|    |     |       |            P = Pause on error
|    |     |       |__________  L = Loop on error
|    |     |                    C = Ignore error (default)
|    |     |
|    |     |                    N = No. of repetitions
|    |     |__________________  0 = Indefinite loop
|    |                          1 = Default
|    |                         99 = Maximum
|    |
|    |________________________  Two-letter name of global
|
|_____________________________  % indicates global
```

Example:

```
>> %DD,0,C
```

This command means: run the drive diagnostic, repeating indefinitely, continue in case of error. This is exactly equivalent to:

```
>> LPT
>> .IN,1,C
>> .PS,1,C
>> .DX,1,C
>> LUP,0
```

This, in turn, is exactly equivalent to:

```
>> LPT
>> ST,1,C
>> SC,1,C
>> SS,1,C
>> LS,1,C
>> OS,1,C
>> RS,1,C
>> WD,1,C
>> LUP,0
```

The operator may intermix global, macro and primitive commands, using the loop commands as desired. The global command is for convenience only. The same result may be achieved by the equivalent string of macro commands or primitive commands.

1.3.4 Utility Commands

The utility commands are described as follows:

ES Print an error summary.

FM Format the disk pack between the upper and lower limits, using the '177777 data pattern, and checking headers as follows:

- o set format switch on controller to the ON position
- o execute 94DIAG
- o enter Control U
- o enter upper limit of disk pack in the format

  A-B-C-D

where A = unit number
B = maximum cylinder number (number of cylinders per pack minus one)
C = maximum track number (number of tracks per cylinder minus one)
D = maximum sector number (number of sectors per track minus one)

WARNING

Factory formatted FMDs (e.g., the CDC 9775 and the Fujitsu 2284 drives), contain bad block information in the last track. If the drive must be reformatted, do NOT format the last track. The value for D above is equal to the number of tracks minus two.

- o enter the lower limit of disk pack in the format

  A-0-0-0

- o execute test FM

A sample disk pack format procedure follows:

```
>> 94DIAG (VERSION 2.5        10-APR-81)

>> CRT OR TTY (C)? =  T
>> DATE (M-D-Y)  9-1-81
>> TIME (H:M:S)  17:00
>> 9800 CONTROLLER (Y)?
>> STD CPA LOC (Y)?   Y
>> SET FMT EN SWITCH ON; PUT STD/EXT SW TO EXT
>> RST CTLR, HIT <RET> (R)? =
>> UNIT 1: RM03  DIRECT 9762
>> OK (Y)?   Y
>> WILL USE:
>> UNIT 1  = 1-817-0-0  THRU 1-821-4-31
>> ENTER NEW CMDS
>>
** (^U)
** UPPER LIM (1-821-4-31)   1-822-4-31
** LOWER LIM (1-817-0-0)  1-0-0-0
>> FM
>>  /
    --FM (FMT)
FORMATTING
 1-0-0-0  THRU 1-822-4-31
CHECKING HEADERS...
>> RD:  #1

>> END TESTS
>> ENTER NEW CMDS
>>
```

LPT	Loop back to this point in the command string.
LUP,N	Loop back to LPT N times, then go on.

/	End of command string. This command is also the signal to begin executing the command string.
\	Repeat the last command string. This command eliminates the need to re-enter the command string.
NP	Get the next data pattern from the table of 8 patterns and use it as the current data pattern.
PN	Print the pass number. This command may be placed anywhere in the command string but is most logically placed at the end of a loop.
RD	Read the surface of the disk between the upper and lower limits. This command sets the Bus Address Increment (BAI) bit and reads 256 sectors at a time. Any errors are reported unless they occur in known bad blocks. After an error, RD continues reading from next sector.
RE	Clear the error summary and the log of words written and read.
RL	Load a register. This command allows the user to load the front panel switches into any desired CPA or MPU register and displays the result. For this command only, the error specifier means:

L = Go into a tight scope loop. The operator may change the data pattern at any time by changing the switches. He may use the CTL C dynamic command to stop the looping, or he may use the CTL E dynamic command to change the error specifier.

P = Keep asking for a new CPA address.

C = Load one register, then go to next test

Example (where 125252 is loaded into the switches):

```
>> ENTER CMD
>> RL,P
>> /
--RL (REGISTER LOAD)
>> CPU OR MPU (C)? =
```

Type "C" for CPU registers, or "M" for MPU registers.

For CPU:

```
>> ADDRESS ('176700)? = 176702
>> BEFORE: '176400, SWITCHES '125252, AFTER: '125252
```

The display shows the contents of the register before the switches are loaded into it and after they are loaded into it. If the user enters an odd number for the address, the system will round it down to the next lower even number. If the user enters the address of a non-existent memory location, the system will first trap and then re-start the diagnostic at the beginning of the initialization.

For MPU:

```
>> ADDRESS ("1000)? = 404
>> BEFORE: '000220, SWITCHES: '125252, AFTER: '000252
```

Note that since MPU registers are only 8-bits long, only the 8 switches on the right will be loaded.

RP Get a random number and use it as the current data pattern.

RU Read utility. This command performs a READ HEADER & DATA operation. It reads the number of words specified by the operator starting at the lower address limit. It does not check for errors. Error specifier has the following meaning:

C = Read once, print the first 8 words read, then exit.

P = Read once, print the first 8 words read, then offer to read again.

L = Loop continuously, do not print words read.

TD Print the date and time.

WH	Write headers in the domain between the upper and lower limits using the current data pattern for data.
WU	Write Utility. This command performs a WRITE HEADER & DATA operation. It gets the word count from the operator, starts writing at lower address limit, and uses the contents of the switches for data. It does not check error status. The error specifier means: C = Write once, then exit. P = Write once, then offer to write again. L = Loop continuously. The operator may vary the switch settings to vary the data while looping. Hit Control C to exit.
ZR	Perform the zig-zag read test. This is the same as the read portion of the oscillating track (OT) test.

1.4 Dynamic Commands

Dynamic commands may be entered at any time. Any operation will be suspended until the dynamic command is completed. Another dynamic command may be given before the present one is completed. The present one will be aborted at that instant, and the new command will be executed. When the operation returns to the normal command level, the current line of input/output will be reviewed up to the point where the dynamic command interrupted.

Lines of output at the dynamic command level are preceded by a double asterisk (**). For example:

```
** (CTL P) DATA PATTERNS
CUR ('125252)? =
```

1.4.1 CTL A (Current Address)

This command prints the current disk address in the format:

```
UNIT-CYLINDER-HEAD-SECTOR
```

If the dynamic command is executed during or after a standard emulation test, it shows the address given in the CPA registers. If executed during or after an extended emulation test, it shows the address in the current address software registers.

1.4.2 CTL B (Bootstrap)

This command transfers control to the bootstrap program which allows the user to load software from the specified device. The system will print:

```
BOOT UNIT:-
```

The operator may then enter the unit from which he wishes to boot. If the user is in doubt about what to enter, he may enter "?" and instructions will be given. See Appendix A for more detailed instructions.

In addition, the operator may also give a command which will write the diagnostic onto a magnetic tape. This is useful in cases where the user has entered patches to the diagnostic and wishes to save the updated version.

If the user wishes to return to the diagnostic, he may do so by entering "I".

1.4.3 CTL C (Clear Command String)

This command immediately terminates the execution of the command string and requests a new command string from the user.

This command is essential for ending an infinite loop in the command string, or for terminating the unwanted remainder of a command string.

This command does not destroy the command string, and the user may re-start an existing command string by entering the \ (backslash) command.

1.4.4 CTL E (Error Specifier)

This command allows the user to change the error specifier for the test currently being run. The command prints:

```
** (CTL E) ERR SPEC (L)? =
```

The letter in parenthesis indicates the current error specifier. If no letter is entered, the current specifier is retained. The operator may enter one of the following letters:

L = Loop on error
P = Pause on error
C = Continue on error (ignore it)

The letter entered will then become the current specifier until the next test or until the next loop of the same test.

1.4.5 CTL F (Error Format)

This command allows the user to specify what information he wants reported in case of error. Each piece of information is identified by a two-letter mnemonic:

BA, = RPBA	EA, = Address-#Errors	R2, = RPER2
BE, = Bell	EM, = Test's error message	R3, = RPER3
CC, = RPCC	GB, = Good/Bad	S1, = RPCS1
DA, = RPDA	LA, = Logical address	S2, = RPCS2
DC, = RPDC	PA, = Physical address	WC, = RPWC
DS, = RPDS	PC, = Program counter ref.	
DT, = Date	R1, = RPER1	

0 = None of the above
1 = All of the above

The command first reports the current format. For example:

```
** (CTL F) FORMAT = BE,DT,PC,LA,PA,WC,DS,DA,
** NEW FORMAT =
```

The user then enters the desired mnemonics all on a single line, each one separated from the next by a comma. See Section 1.6 for a full description of the error report.

1.4.6 CTL G (Read Header)

This command allows the user to read and display any header. The user enters UN-CY-HD-SC (Unit, Cylinder, Head, Sector). The contents of the header specified are displayed. If a read error occurs, this fact is noted, but the data is still displayed.

The command will continue asking for new addresses to read. When the user is finished, striking <RET> will return to the normal command level.

1.4.7 CTL I (Initialize)

This command allows the user to reinitialize the system. He will go through the same initialization dialogue as when he first started the diagnostic.

1.4.8 CTL K (Error Limit, Retry Limit)

This command allows the user to define how many errors a drive will be allowed to make before being dropped from testing. The default value of zero indicates infinite errors allowed. Once a drive is dropped, that fact will be indicated by a message to the console. Also, a notation will be made in the error summary. If all the drives are dropped, testing will be aborted. Clearing the error summary will allow drives to be reinstated.

This command also allows the operator to set the number of retries to be attempted before a hard error is logged.

1.4.9 CTL N (Error Summary)

The user is given the choice of clearing the error summary after printing, or allowing it to accumulate. This will not, however, clear the summary of words written and read.

This command prints an error summary in five categories for each unit being tested. In each of these categories, the first number represents hard errors, the second soft errors.

Following this is a summary of the number of words written and read by each unit being tested. Since this can be a very large number, it is given in scientific notation.

```
            ** (CTL N) ERR SUM
            >> CLR AFTER PRINTING (N)?

            ADDR VER   DATA VER   ECC ERROR    CRC ERROR   MISC ERROR

     UNIT 0  3/5       120/200       0/0          0/0          5/0

     UNIT 0  WDS WRITTEN = 2.56 E2.   READ = 2.56 E2
```

1.4.10 CTL O (ECIB Table)

If the previous operation has used extended emulation, this command prints the ECIB table associated with it.

If the previous operation has used standard emulation, the command prints the current ECIB table as well as the three previous ECIB tables. This is printed in four columns, with the current table in the leftmost column. For a description of the ECIB table, see section 1.7.

1.4.11 CTL P (Data Pattern)

This command displays the current data pattern and allows the user to change it. For example:

```
        ** (CTL P) DATA PATTERNS
        CUR ('125252)
```

The program expects an octal number. Following this, the pattern table is displayed item by item, allowing the user to change it. If the user does not want to change the table, he can strike the <ESC> key, or Control Z.

1.4.12 CTL Q (Title Suppress)

This command allows the user to allow or suppress the printing of titles.

1.4.13 CTL R (Resume)

This command allows the user to resume operation under any of the following conditions:

- o When a test has paused on error.
- o When a test is suspended by the CTL S command.

1.4.14 CTL S (Suspend)

This command suspends all testing until the CTL R command is given. Other dynamic commands may be executed during this suspension, but upon their completion, the testing will remain suspended. However, the CTL C command cancels the suspension.

1.4.15 CTL U (Disk Address)

This command allows the user to know the upper and lower limits of the disk address. The user may enter a new value or may retain the old one by pressing <RETURN> without entering a value. The program expects positive decimal integers whose value does not exceed the limits established at initialization. Example:

```
        (CTL U)
        ** UPPER LIM (2-410-18-21)
        ** LOWER LIM (0-0-0-0)
```

The first number is unit number, the second is cylinder, the third is head, and the fourth is sector. If a new address is to be entered, all four numbers must be given.

It is not advisable to change the limits while a test is running. It is recommended instead that the user abort the test with Control C, change the address limits, and then use \ (backslash) to re-execute the command string.

1.4.16 CTL V (Hardware Registers)

This command prints the contents of the hardware registers in the 9400 controller. See Section 1.8 for a more detailed description of the contents of these registers.

1.4.17 CTL W (Review Command String)

If this command is executed while the command string is still being built, it will list all commands entered before the last carriage return. If executed while another command is executing, it will list all the commands in the string. In addition, an arrow will point to the test currently running, and a number beside the arrow will indicate how many iterations are left. For example:

```
SS,1,C
LS,9,L  <--5
OS,5,P
/
```

In addition, if a loop is being executed, an additional arrow at the LUP command indicates how many iterations are left. Example:

```
LPT
SS,1,C
LS,9,L  <--5
LUP,5   <--2
LPT
WP,1,C
CR,3,L
LUP,6
```

If the user has built the command string from macro (group of tests) or global (complete diagnostics) commands, the command interpreter will expand these to the primitive commands (tests). The review of the command string will show the primitive tests called by the macros or globals. For example:

```
>> ENTER NEW CMDS
>> .PS,L
>> SC,C

            ** (CTL W) CMD SUMMARY
LPT
SS,1,L
LS,1,L
OS,1,L
RS,1,L
LUP,1
SC,1,C
/
>>
```

1.4.18 CTL X (CPA Registers)

This command prints the contents of the CPA registers for each unit specified, from the lower to the upper limit. See Section 1.7 for a full description of the CPA registers.

1.5 Test Summaries

The following subsections list the summaries for primitive macro, and global tests.

1.5.1 Summary of Primitive Tests

	GLOBAL	MACRO
AV = Address Verify	%SD	.FI
CR = Crc Error Test	%SD	.FI
CS = Cylinder Switch Test	%SD	.FI
DV = Data Verify Test	%SD	.BC
EC = ECC Error Test	%SD	.FI
EK = Echo Test	%CD	.RG
EM = Emulation Test	%CD	.RG
ES = Print Error Summary	(Utility)	
FM = Format the Disk	(Utility)	
FT = Format Test	%SD	.DR
HD = Header Test	%SD	.DR
HS = Head Switch Test	%SD	.BC
IT = CPU Interrupt Test	%CD	.CF
LPT= Start of Loop	(Utility)	
LS = Long Seek Test	%DD	.PS
LUP= Loop Command	(Utility)	
MI = MPU Interrupt Test	%CD	.CF
MW = Maximum Word Transfer	%SD	.BC
NP = Next Data Pattern	(Utility)	
OS = Oscillating Seek Test	%DD	.PS
OT = Oscillating Track Test	%DD	.DR
PM = Print Pass Number	(Utility)	
RC = Half Read Test	%CD	.DC
RD = Read Surface	(Utiility)	
RE = Reset Error Summary	(Utility)	
RL = Register Load	(Utility)	
RP = Next Random Data Pat.	(Utility)	
RS = Random Seek Test	%DD	.PS
RT = Random Track W/R Test	%SD	.DR
RU = Read Utility	(Utility)	
SC = Sector Counter Test	%DD	.IN
SS = Sequential Seek Test	%DD	.PS
ST = Unit Status Test	%DD	.IN
SW = Single Word Xfr Test	%SD	.BC
TD = Print the Date	(Utility)	
WC = Half Write CPU Test	%CD	.DC
WD = Half Wr/Rd Drive Test	%DD	.DX
WH = Write Headers	(Utility)	
WU = Write Utility	(Utility)	
ZR = Zig-zag read test	(Utility)	

1.5.2 Macro and Global Definitions

```
%CD = Computer Interface Diagnostic
     .RG = Register Tests
          EK = Echo Test
          NP = Next Data Pattern
          EK
          NP
          EK
          NP
          EK
          NP
          EM = Emulation Test
     .CF = CPU/Formatter Tests
          IT = CPA Interrupt Test
          MI = MPU Interrupt Test
     .DC = CPU/DMA Tests
          WC = Half Write CPU Test
          RC = Half Read CPU Test

%DD = Drive Diagnostic
     .IN = Interface Tests
          ST = Unit Status Tests
          SC = Sector Counter Test
     .PS = Positioner Tests
          SS = Sequential Seek Test
          LS = Long Seek Test
          OS = Oscillating Seek Test
          RS = Random Seek Test
     .DX = Drive/DMA Tests
          WD = Half Read/Write Drive Test

%SD = System Diagnostic
     .FI = Fault Injection Tests
          BB = Bad Block Error Test
          CR = CRC Error Test
          EC = ECC Error Test
          AV = Address Verify Test
          DV = Data Verify Test
     .BC = Boundary Condition Tests
          SW = Single Word Transfer Test
          MW = Maximum Word Transfer Test
          HS = Head Switch Test
          HD = Header Test
          CS = Cylinder Switch
     .DR = Data Reliability Tests
          FT = Format Test
          OT = Oscillating Track Test
          RT = Random Track Read/Write Test
```

1.6 Error Report

The error report consists of three parts:

1. The contents of various registers as requested by the user.
2. An analysis of the error bits contained in the error registers.
3. An error message from the test itself. This includes an error number referencing the Error Dictionary to which the operator may refer for a fuller analysis of the error.

Since standard emulation tests use different registers than extended emulation tests, the error report format differs for these two types of tests. The full error report for a standard emulation test will appear as follows:

```
M/D/Y - H:M:S           PC = Program Counter    RPER1 = Err Reg #1
RPER2 = Err Reg #2      RPER3 = Err Reg #3      RPCS1 = Ctl & Stat #1
RPCS2 = Ctl & Stat #2   RPDS = Drive Status     RPDA = Hd/Sect Addr
RPDC = Desired Cyl      RPCC = Current Cyl      RPBA = Bus Addr
RPWC = Word Count       GD-BAD = Exp'd-Rec'd    EA = Addr-#Errors

          >>  CPA REG STATUS: DISK ADDR = UN-CY-HD-SC
              (Analysis of error bits in cpa registers)
          ||  (Message from test)
```

For a full description of the contents of the CPA registers, see Section 1.7.

The full error report for extended emulation tests will appear as follows:

```
M/D/Y - H:M:S        LOG ADR = Un/Cy/Hd/Sc     PHYS ADR = U/C/H/S
PC = Program Counter GD-BAD = Exp'd - Rec'd    EA = Addr-#Errors

          >> ECIB TABLE STATUS:
             (Analysis of error bits in ECIB table)
          || (Message from test)
```

If the user wishes to examine the ECIB table following an extended emulation test, he may use the CTL O dynamic command. Refer to Table 1.

TABLE 1

ECIB TABLE:

ECOS1 - OPERATION STATUS 1

Octal	Hexadecimal
'000200	"0080 - Any error
'000100	"0040 - Deferred (not implemented)
'000040	"0020 - Unit fault
'000020	"0010 - End of header timeout
'000010	"0008 - End of sector timeout
'000004	"0004 - Data buffer timeout
'000002	"0002 - Parity error high
'000001	"0001 - Parity error low

ECOS2 - OPERATION STATUS 2

Octal	Hexadecimal
'000200	"0080 - Hardware/firmware protect
'000100	"0040 - Software protect
'000040	"0020 - Address verify error
'000020	"0010 - Data verify error
'000010	"0008 - CRC error
'000004	"0004 - ECC error
'000002	"0002 - ECC correction
'000001	"0001 - Retry cor. (not implemented)

ECOS3 - OPERATION STATUS 3

Octal	Hexadecimal
'000200	"0080 - Pack overrun
'000100	"0040 - Bad block
'000040	"0020 - Last sector on pack
'000020	"0010 - Unit select fault
'000010	"0008 - Seek complete timeout
'000004	"0004 - Unit access error
'000002	"0002 - Illegal command implementation
'000001	"0001 - CPA error

ECUS1 - PRIMARY DRIVE STATUS

Octal	Hexadecimal
'000200	"0080 - Attention
'000100	"0040 - Dual Port busy
'000040	"0020 - Seek error
'000020	"0010 - Unit selected
'000010	"0008 - Write protected
'000004	"0004 - Fault
'000002	"0002 - On cylinder
'000001	"0001 - Ready

ECUS2 - MODEL BYTE OR SECTOR COUNT

ECPORT - PORT NUMBER ON WHICH DRIVE WAS FOUND (0-3)

ECCPA - COMPUTER PORT ADAPTER (CPA) NUMBER (0-3)

ECFC - FUNCTION CODE

Octal	Hexadecimal	
'000000	"0000	Clear controller
'000001	"0001	Sense model status, put in ECUS2.
'000002	"0002	Sense sector count, put in ECUS2.
'000003	"0003	Sense diagnostic status, put in ECUS2.
'000004	"0004	Sense fault status, put in ECOS1, ECOS2, ECOS3.
'000005	"0005	Firmware reserve.
'000006	"0006	Firmware reserve.
'000007	"0007	Set firmware write protect.
'000010	"0008	Clear firmware write protect.
'000011	"0009	Clear drive attention.
'000012	"000A	Initiate recalibrate.
'000013	"000B	Initiate seek.
'000014	"000C	Write data.
'000015	"000D	Read data.
'000016	"000E	Read verify data.
'000017	"000F	Write header.
'000020	"0010	Read header.
'000021	"0011	Read verify header.
'000022	"0012	Write header & data.
'000023	"0013	Read header & data.
'000024	"0014	Read verify header & data.

ECLUC - PHYSICAL DRIVE NUMBER

ECLCYH - LOGICAL CYLINDER ADDRESS (MS byte)

ECLCYL - LOGICAL CYLINDER ADDRESS (LS byte)

ECLHED - LOGICAL HEAD ADDRESS

ECLSEC - LOGICAL SECTOR ADDRESS

ECPCYH - PHYSICAL CYLINDER ADDRESS (MS byte)

ECPCYL - PHYSICAL CYLINDER ADDRESS (LS byte)

ECPHED - PHYSICAL HEAD ADDRESS

ECPSEC - PHYSICAL SECTOR ADDRESS

ECFCM - FUNCTION CODE MODIFIER

Octal	Hexadecimal
'000200	"0080 - Half operation
'000100	"0040 - Cpu side (if half operation)
'000040	"0020 - Include ECC/CRC in transfer
'000020	"0010 - ECC correction inhibit
'000010	"0008 - Address compare inhibit
'000004	"0004 - CRC error inhibit

ECMSB - MAP STATUS BYTE

Octal	Hexadecimal
'000200	"0080 - New head
'000100	"0040 - New cylinder
'000040	"0020 - Last sector on pack

ECWCH - WORD COUNT MS BYTE (2's complement)

ECWCL - WORD COUNT LS BYTE (2's complement)

ECSSO - STARTING STROBE OFFSET (not implemented)

ECRPT - REPEAT COUNT STROBE/OFFSET (not implemented)

ECRTY - TOTAL RETRY COUNT (not implemented)

ECFLAG - FLAG BYTE

Octal	Hexadecimal
'000200	"0080 - Write protected flag
'000100	"0040 - Bad sector flag

ECUSER - USER BYTE

1.7 CPA Registers

The CPA registers and a description of each are listed as follows:

'176700 - RPCS1 (Control & Status 1 Register):

'100000	Special condition: Transfer error, or attention, or I/O bus parity error.
'040000	Transfer error, data late, or write check error, or parity error, or non-existent drive, or non-existent memory, or program error, or missed transfer, or mass data bus, or drive error during transfer.
'020000	Bus parity error (not used).
'004000	Drive available (not used).
'002000	Port select (not used).
'001000	Unibus extension bit.
'000400	Unibus extension bit.
'000200	Ready.
'000100	Interrupt enable.
'000074	Read micro-control (extended emulation).
'000072	Read header & data.
'000070	Read data.
'000064	Write micro-control (extended emulation).
'000062	Write header & data.
'000060	Write data.
'000054	Jump micro-control (extended emulation).
'000050	Write check data.
'000030	Search command.
'000022	Pack acknowledge.
'000016	Return to centerline.
'000014	Offset command.
'000012	Release (dual port operation).
'000010	Drive clear.
'000006	Recalibrate.
'000004	Seek.
'000002	Unload (standby).
'000001	GO bit.
'000000	No operation.

'176702 - RPWC (Word count register)

'176704 - RPBA (Unibus address register)

```
'176706 - RPDA (Desired sector/head address register)

     'X174XX - Desired Head Address #37
        |          |      |      |
     'X004XX - Desired Head Address #1
     'XXXX27 - Desired Sector Address #27
        |          |      |      |
     'XXXX00 - Desired Sector Address #0

'176710 - RPCS2 (Control & Status 2 register)

     '100000 - Data late (not used).
     '040000 - Write check error.
     '020000 - Parity error (not used).
     '010000 - Non-existent drive.
     '004000 - Non-existent memory.
     '002000 - Program error.
     '001000 - Missed transfer.
     '000400 - Mass data bus parity error (not used).
     '000200 - Output ready
     '000100 - Input ready (not used).
     '000040 - Controller clear.
     '000020 - Parity test (not used).
     '000010 - Inhibit bus increment.
     '000007 - Select drive #7
        |         |      |
     '000000 - Select drive #0

'176712 - RPDS (Drive Status Register)

     '100000 - Attention active.
     '040000 - Error in RPER1, RPER2, or RPER3.
     '020000 - Positioning in progress (not used).
     '010000 - Medium on line.
     '004000 - Write lock.
     '002000 - Last sector transferred.
     '001000 - Programmable (not used).
     '000400 - Drive present (not used).
     '000200 - Drive ready.
     '000100 - Volume valid.
     '000001 - RM03: Offset mode.  RP04: (not used).
```

'176714 - RPER1 (Error Register #1)

'100000	Data check, ECC error.
'040000	Unsafe.
'020000	Operation incomplete (not used).
'010000	Drive timing error (not used).
'004000	Write lock error.
'002000	Invalid address error.
'001000	Address overflow error.
'000400	Header CRC error.
'000200	Header compare error.
'000100	ECC hard error.
'000040	Write clock failed (not used).
'000020	Format error.
'000010	Parity error.
'000004	Register mod refused (not used).
'000002	Illegal register (not used).
'000001	Illegal function.

'176716 - RPAS (Attention Summary Register)

'000200	Drive #7 Attention.
'000100	Drive #6 Attention.
'000040	Drive #5 Attention.
'000020	Drive #4 Attention.
'000010	Drive #3 Attention.
'000004	Drive #2 Attention.
'000002	Drive #1 Attention.
'000001	Drive #0 Attention.

'176720 - RPLA - (Not emulated)

'176722 - RPDB - (Data Buffer Register)

'176724 - RPMR - (Not emulated)

'176726 - RPDT - (Drive Type Register)

'020000	Moving head disk type.
'004000	Dual controller option available.
'000020	RP04
'000021	RP05
'000022	RP06
'000024	RM03
'000027	RM05

'176730 - RPSN (Serial Number Register)

Used to indicate further drive information:

Bits 0-7 - Logical Unit Number

If RP04/RP05/RP06	If RM03, bit 15 = 1 If RM05, bit 15 = 0
'100400 = RP04 Mapped 9762	'000400 = 9762
'101400 = RP04 Mapped 9766	'001400 = 9766
'103400 = RP06 Mapped 9766	'002200 = 9448-32 Fixed
'104400 = RP05 Mapped 9762	'002000 = 9448-32 Removable
'105400 = RP05 Mapped 9766	'002600 = 9448-64 Fixed
'140400 = RP04 Direct 9762	'002400 = 9448-64 Removable
'141400 = RP04 Direct 9766	'003200 = 9448-96 Fixed
'143400 = RP06 Direct 9766	'003000 = 9448-96 Removable
'144400 = RP05 Direct 9762	'003400 = 675 MB (W/O Fx Hd)
'145400 = RP05 Direct 9766	'004000 = 675 MB (With Fx Hd)
	'004400 = 9730-80 (W/O Fx Hd)
	'005000 = 9730-80F (With Fx Hd)
	'005400 = 160 MB (W/O Fx Hd)
	'006000 = 160 MB (With Fx Hd)
	'006400 = Mapped 160 MB
	'007000 = Mapped 675 MB

'176732 - RPOF (Offset Register)

'100000 - Sign change (not used).
'010000 - Format bit (not used).
'004000 - Error correction code inhibit.
'002000 - Header compare inhibit.

<u>RP04:</u>	<u>RM03:</u>
'000260 - Offset -1200 micro-in.	'000200 - Offset towards spindle.
'000240 - Offset -800 micro-inches	
'000220 - Offset -400 micro-inches	
'000060 - Offset +1200 micro-inches	
'000040 - Offset +800 micro-inches	
'000020 - Offset +400 micro-inches	
'000000 - Return to track centerline.	

'176734 - RPDC (Desired Cylinder Register)

'176736 - RPCC (Current Cylinder Register)

<u>RP04:</u>	<u>RM03:</u>
Current Cylinder	Not used

'176740 - RPER2 (Error Register #2)

	RP04:	RM03:
'100000 -	AC unsafe.	Not used.
'020000 -	Unsafe, R/W phase lock out of sync.	
'010000 -	30 volts unsafe.	

'176742 - RPER3 (Error Register #3)

	RP04:	RM03:
'100000 -	Off cylinder (not used).	Marked bad sector
'040000 -	Seek incomplete.	
'000100 -	Low 5 volt DC.	
'000040 -	Low AC.	
'000010 -	Head retract occurred.	
'000002 -	Velocity unsafe (not used).	
'000001 -	Pack speed unsafe (not used).	

'176744 - RPEC1 (Not emulated)

'176746 - RPEC2 (Not emulated)

'176750 - RMBA (Bus Address Extension Register)

PDP 11/70 Only

'176752 - RMSC3 (Control & Status #3)

PDP 11/70 Only

'100000 - Address parity error.
'040000 - Data parity error odd word.
'020000 - Data parity error even word.
'002000 - Last memory transfer was a double word operation.
'000100 - Interrupt enable

1.8 Hardware Registers

The hardware registers and their definitions are listed as follows:

"0800	(AVCYLH) Address Verify Cylinder Hi.
"0801	(AVCYLL) Address Verify Cylinder Lo.
"0802	(AVHD) Address Verify Head.
"0803	(AVLS) Address Verify Logical Sector.
"0804	Keyword #1 Hi
"0805	Keyword #1 Lo.
"0806	Keyword #2 Hi.
"0807	Keyword #2 Lo.
"0808	Flag Byte.
"0809	User Byte.
"080A-"080F	
"0810	(DMAWCH) DMA Word Count Hi.
"0811	(DMAWCL) DMA Word Count Lo.
"0812	(DBARH) Disk Buffer Address Hi.
"0813	(DBARL) Dsic Buffer Address Lo.
"0814	(CBARH) Computer Buffer Address Hi.
"0815	(CBARL) Computer Buffer Address Lo.
"0816	(DBCH) Buffer Counter Hi.
"0687	(DBCL) Buffer Counter Lo.

"0818 (DEVF) Data Event Flags.

"0080 - (AE) Any Error.
"0040 - (AA) Any Attention.
"0020 - (BA) Buffer Available.
"0010 - (BR) Buffer Ready.
"0008 - (BE) Buffer Empty.
"0004 - (TC) Transfer Complete.
"0002 - (ES) End of Sector.
"0001 - (EH) End of Header.

"0819 (DERFH) Data Error Flags Hi

"0020 - (BBF) Bad Block Flag.
"0010 - (WPF) Write Protect Flag.

"081A (DERFL) Data Error Flags Lo

"0080 - (SO) Sector Overrun.
"0040 - (BPH) Buffer Parity Error Hi.
"0020 - (BPL) Buffer Parity Error Lo.
"0010 - (DVE) Data Verify Error.
"0008 - (AVE) Address Verify Error.
"0004 - (ECE) ECC Error.
"0002 - (CE) CRC Error.
"0001 - (ESI) Error Stop Inhibit.

"081B (DPA) Data Port Address

"0040 - (DP2) Drive Interface Port bit 2.
"0020 - (DP1) Drive Interface Port bit 1.
"0002 - (CP2) Computer Port bit 2
"0001 - (CP1) Computer Port bit 1.

"081C (PSA) Physical Sector Address

"081D (DOC) Data Operation Control

"0080 - (RBC) Reset Buffer Controls
"0040 - (CE) Computer Enable
'0020 - (DE) Disk Enable
"0010 - (TCC) Transfer Check Code.
"0008 - (IDF) Inhibit Data Field.

"0840 (DIB) Disk Interface Input

```
"0841  (DIF) Disk Interface Flags

     "0008 - (ATN3) Drive Interface Port Att'n 3
     "0004 - (ATN2) Drive Interface Port Att'n 2
     "0002 - (ATN1) Drive Interface Port Att'n 1
     "0001 - (ATN0) Drive Interface Port Att'n 0

"0844  (DIO) Disk Interface Output

"0845  (DIC) Disk Interface Control

     "0040 - Strobe D, select tag to CDC drive.
     "0020 - Strobe C, load cyl hi.
     "0010 - Strobe B, head tag.
     "0008 - Strobe A, cylinder tag.
     "0004 - Tag Bus 2
     "0002 - Tag Bus 1
     "0001 - Tag Bus 0
                001  Select Sector Counter
                010  Select Sense
                011  Select Status
                100  Select Control

"0846  (ECCSC) ECC Shift Control

     "0008 - Shift ECC data in polynomial reg #3.
     "0004 - Shift ECC data in polynomial reg #2.
     "0002 - Shift ECC data in polynomial reg #1.
     "0001 - Shift ECC data in polynomial reg #0.

"0850  (ECCB0) ECC Byte 0

"0851  (ECCB1) ECC Byte 1

"0852  (ECCB2) ECC Byte 2

"0853  (ECCB3) ECC Byte 3

"0854  (ECCB4) ECC Byte 4

"0855  (ECCB5) ECC Byte 5

"0856  (ECCB6) ECC Byte 6
```

"0857 (ECCZF) ECC Zero Flags

"0008	- Set when ECC polynomial reg #3 is zero.
"0004	- Set when ECC polynomial reg #2 is zero.
"0002	- Set when ECC polynomial reg #1 is zero.
"0001	- Set when ECC polynomial reg #0 is zero.

2.0 (%DD) DRIVE DIAGNOSTIC

The %DD diagnostic tests those portions of the controller which are most closely associated with the drive. The tests make minimal usage of the CPA registers and the emulation firmware. All instructions and status are handled via the extended emulation.

2.1 (ST) Status Test

Execution time: 4 seconds.

Prerequisites: None.

Type of emulation: Extended.

Tests the status bits in ECUS1 as described in the following subsections.

2.1.1 Check "On Cylinder" Bit

ST performs a Recal and checks that the "On Cylinder" bit has set in a reasonable time. If not, it flags an error and returns to the monitor. If "On Cylinder" sets, it checks the status bits in ECUS1 which should be as shown:

ECUS1	
Bit 7 = 0/1	Attention
Bit 6 = 0	Dual port busy
Bit 5 = 0	Seek error
Bit 4 = 1	Unit selected
Bit 3 = 0/1	Write protected
Bit 2 = 0	Fault
Bit 1 = 1	On cylinder
Bit 0 = 1	Ready

If not, the test flags an error and returns to the monitor.

2.1.2 Seek To Cylinder #1

ST then seeks to cylinder #1 and checks that the "On Cylinder" bit has set in a reasonable time. If not, it flags an error and returns to the monitor.

2.1.3 Seek To Cylinder 1777777

The test next seeks to an impossible cylinder (177777), and waits for either the "Seek Error" or the "On Cylinder" bit to set in a reasonable time. If not set, it flags an error and returns to the monitor. If set, it checks the status bits in ECUS1 which should be as shown:

ECUS1

Bit 7 = 0/1	Attention
Bit 6 = 0	Dual port busy
Bit 5 = 1	Seek error
Bit 4 = 1	Unit selected
Bit 3 = 0/1	Write protected
Bit 2 = 0	Fault
Bit 1 = 1	On cylinder
Bit 0 = 1	Ready

2.1.4 Long Seek

ST does a long seek, checking that the "On cylinder" bit is clear while the drive is seeking, and that it sets at the end of the seek.

APPLICABLE ERROR REPORT OPTIONS:

PA:	Contains the physical drive address.
LA:	Contains the logical drive address.
EM:	Gives error message from test.

2.2 (SC) Sector Counter Test

Execution time:	10 seconds.
Prerequisites:	None.
Type of emulation:	Extended.

The test requests the sector count and places it in a buffer. It continues to do this as fast as the firmware can supply the count until the buffer is full. It then analyzes the numbers in the buffer and determines that every integer between 0 and the maximum sector count given at initialization is present. If any integers are missing, the test flags an error and returns to the monitor.

APPLICABLE ERROR REPORT OPTIONS:

GB:	Good/Bad. The first number is the expected sector count, the second number is that received (octal).
LA:	Logical address shows of failing unit.
PA:	Physical address of failing unit.
EM:	Gives error message from test.

2.3 (SS) Sequential Seek Test

Execution time:	2 minutes (80 megabyte).
Prerequisites:	Upper cylinder limit must be greater than the lower limit.
Type of emulation:	Extended

SS performs sequential seeks from the lower to upper cylinder address. One pass through the test will sequentially ascend and descend through all the cylinder addresses once. After each seek, the test checks status and reports any error.

APPLICABLE ERROR REPORT OPTIONS:

PA:	Contains the physical seek address which caused the error.
LA:	Contains the logical seek address which caused the error.
EM:	Gives error message from test.

2.4 (LS) Long Seek Test

Execution time:	1 second.
Prerequisites:	Upper cylinder limit must be greater than lower.
Type of emulation:	Extended.

LS seeks from the lower limit cylinder address to the upper limit cylinder address and back again. It checks status after each seek. If the status shows an error, it flags an error and returns to the monitor.

APPLICABLE ERROR REPORT OPTIONS:

PA:	Contains the physical seek address which caused the error.
LA:	Contains the logical seek address which caused the error.
EM:	Gives error message from test.

2.5 <u>(OS) Oscillating Seek Test</u>

Execution time:	5 minutes, 20 seconds (80 Megabyte).
Prerequisites:	Upper cylinder limit must be greater than lower.
Type of emulation:	Extended.

OS seeks from the lowest cylinder to the lowest+1 and back again. Then it seeks from the lowest to the lowest+2 and back again. It continues seeking from the lowest to successively higher addresses until the upper limit is reached. Then it successively reduces the upper address until it equals lowest+1 again.

APPLICABLE ERROR REPORT OPTIONS:

PA:	Contains the physical seek address which caused the error.
LA:	Contains the logical seek address which caused the error.
EM:	Gives error message from the test.

2.6 <u>(RS) Random Seek Test</u>

Execution time:	1 minute, 30 seconds (80 Megabyte).
Prerequisites:	Upper cylinder limit must be greater than the lower limit.
Type of emulation:	Extended

RS seeks from one random cylinder address to another within the domain bounded by the upper and lower limits of the cylinder address as set by the dynamic command CTL U. It continues until all the cylinders within that domain have been addressed. It checks status after each seek and reports the first error. It tallies successive errors in the error summary.

APPLICABLE ERROR REPORT OPTIONS:

PA: Contains the physical seek addresses which caused the error.

LA: Contains the logical seek address which caused the error.

EM: Gives error message from the test.

2.7 (WD) Half Read/Write Test

Execution time: 2 seconds.

Prerequisites: None.

Type of emulation: Extended.

WD fills all the RAM buffer with an incrementing data pattern, one word at a time, using the WMC command in extended emulation. It does a half write header and sends data to the drive. WD then clears the RAM, and does a half read header and sends data from the drive. It checks data in the RAM one word at a time.

The RAM data buffer is loaded as follows:

```
        Location (Hex)      Contents (Hex)

        1000                00------|
        1001                10      |
        1002                00      |
        1003                00      |---Header (for disk
        1004                00      |    address 0-0-0-0)
        1005                00      |
        1006                00      |
        1007                00-------
        1008                00------|
        1009                01      |
        100A                02      |
          |                 |       |---Data
        1107                FF      |
        1108                00      |
        1109                01      |
          |                 |       |
        1206                FE      |
        1207                FF------|
```

APPLICABLE ERROR REPORT OPTIONS:

GB:	This gives the expected data and the actual data. Since the data is stored in RAM in 8-bit bytes, the right hand side of the number will be the data byte, the left hand side of the number will always be zero.
EA:	This gives the address in RAM (octal) where the first error occurred (the one reported in GB). The number to the right of this is the total number of errors that occurred during the transfer.
EM:	Gives error message from the test.

3.0 (%CD) COMPUTER INTERFACE DIAGNOSTIC

The %CD diagnostic tests those portions of the computer interface which communicate the least with the disk, i.e. this diagnostic deals only with computer interface-controller communications where possible.

These tests include:

- .RG Register Tests
  - EK Register Echo Tests
  - EM Register Emulation Tests
- .CF CPU/Formatter Tests
  - IT CPU Interrupt Test
  - MI Microprocessor Interrupt Test
- .DC CPU DMA Tests
  - RC RAM Buffer Test
  - WC Half Write CPU Test

3.1 (EK) Register Echo Test

Execution time:	1 second.
Prerequisites:	Must have version 1.3 (RM03) or 2.1 (RP04) of the firmware, and CPA board 9400-6031 date code A913 or A907.
Type of emulation:	Standard.

EK tests the ability of each computer interface register to accept and hold data. This is accomplished by targeting each register for test. The target register is loaded with the user specified pattern (see CTL P), except control and read only bits. All other computer interface registers are loaded with the complement of the user specified pattern. This allows detection of addressing or continuous loading errors. Then the contents of the target register is recalled and compared to the user specified pattern. If the comparison is not equal, an error report is displayed.

APPLICABLE ERROR REPORT OPTIONS:

EA: The first number shows the target register address; the second number shows the number of errors (always 1).

GB: The first number is the data pattern which was loaded into the target register and which was expected to be read back; the second number is the number actually read back from the target register.

EM: EM gives an error message from the test.

NOTE

To insure a complete test, (EK) should be run four times in sequence, using the following four data patterns:

1. '177777
2. '000000
3. '125252
4. '052525

This is done automatically when running the %CD global command.

3.2 (EM) Register Emulation Test

Execution time: 1 second.

Prerequisites: Version 1.3 (RM03) or 2.1 (RP04) of the firmware.

Type of emulation: Standard.

EM tests those bits of the computer interface registers which could not, for one reason or another, be tested by the (EK) test. The bits are (in order tested):

1. MOL, DRY, or VV stuck at zero.
2. ATA stuck at one.
3. SC, TRE stuck at one.
4. RDY stuck at zero.
5. RDY stuck at zero after controller function.
6. ATA stuck at zero.
7. ERR stuck at one.
8. RDY stuck at one during Write.

9. GO stuck at zero.
10. GO stuck at one after Write.
11. PGE stuck at one.
12. PGE stuck at zero.
13. IR, CLR stuck at zero.
14. SC, TRE stuck at zero.
15. ERR stuck at zero.

Error report options are the same as for the EK test.

3.3 (IT) CPU Interrupt Test

Execution time: 1 second.

Prerequisites: None.

Type of emulation: Standard.

It requests a computer interface interrupt at a priority level of 7 and tests that an interrupt does not occur at level 6 and 7. It tests that an interrupt does occur at level 5, and that it does not occur after it has been serviced. It also detects any interrupts occurring at an unexpected vector.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from test.

3.4 (MI) Microprocessor Interrupt Test

Execution time: 1 second.

Prerequisites: None.

Type of emulation: Standard.

MI loads unit #7 into RPCS2, loads 40 (controller clear) into RPSC2, and waits for the IR bit to be set by the microprocessor. If the IR bit is not set after a small wait, an error is reported.

APPLICABLE ERROR REPORT OPTIONS:

EM: Error message from the test.

3.5 (RC) RAM Data Buffer Test

Execution time: 4 seconds.

Prerequisites: None.

Type of emulation: Extended.

RC tests the two extended emulation functions, write microcode and read microcode. Consequently, it tests the 1024 words of microprocessor RAM. The test is implemented by writing and reading microcontrol one byte at a time of the user specified pattern over to and back from the RAM area. The data sent is compared to the data received, and errors are reported.

APPLICABLE ERROR REPORT OPTIONS:

EA: First number is the RAM address of the first error; the second number is the total number of errors found.

GB: First number is the expected data pattern to be found in the RAM location shown in EA above; the second number is the data actually read from the RAM location.

EM: Gives an error message from the test.

3.6 (WC) Half Write CPU Test

Execution time: 2 seconds.

Prerequisites: None.

Type of emulation: Extended.

WC tests the ability of the CPU interface-controller to write DMA to the microprocessor RAM area. This is verified by reading the RAM back to the CPU by using read microcontrol commands. The data sent is compared to the data received and errors are reported.

APPLICABLE ERROR REPORT OPTIONS:

GB: The first number is the pattern written; the second number is the pattern read back.

EA: The first number is the RAM address of the first error detected; the second number is the total number of RAM errors detected.

EM: EM gives an error message from the test.

4.0 (%SD) SYSTEM DIAGNOSTIC

The %SD diagnostic tests all the parts of the disk system which have previously been tested individually, or which have not yet been tested. In the Data Reliability tests, the standard emulation firmware is tested for the first time, and all elements of the system are brought into play.

In the test descriptions below, the execution time and number of bits transferred is given for an 80 megabyte mapped drive, with the diagnostic operating in 28K of memory. Larger capacity drives will require longer execution times and transfer more bits of data in many of the tests. Smaller memory size will increase the execution time in many of the tests.

4.1 (BB) Bad Block Error Test

Execution time:	3 seconds.
Prerequisites:	None.
Type of emulation:	Extended
Data pattern:	'000000 and '177777.

BB tests the operation of the bad block bit in the header. It checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. The test then writes a sector with all zero data and the bad block bit set in the header. It attempts to write all ones into that sector, and checks that the bad block error bit is set. It attempts to read data from that sector, and checks that the bad block error bit sets. It removes the bad block bit from the header, reads the header and verifies that the data is still all zeros. If it is all ones, then the system wrote in spite of the bad block bit in the header.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from test which explains the contents.

4.2 (CR) CRC Error Test

Execution time:	3 seconds.
Prerequisite:	None.
Type of emulation:	Extended.
Data pattern:	'177777.

CR checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It checks the validity of the CRC check word in the header. First, it writes a header, then reads the header and CRC word. It then compares this CRC word with a software generated CRC word and verifies that the two are the same. Finally, it writes a header with an erroneous CRC word and verifies that the CRC error bit is set when the header is read back. The 9400 uses a 56 bit ECC check here, whereas the 9800 uses a 32 bit ECC check.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.3 (EC) ECC Error Test

Execution time:	7 seconds.
Prerequisite:	None.
Type of emulation:	Extended.
Data pattern:	Current pattern.

EC checks the validity of the ECC error detection and correction system. It checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. The test writes a sector of data and reads it back, checking status and data. It compares the ECC remainder with a software generated ECC remainder and verifies that they are the same. It then creates 11 consecutive errors in the data and writes it and the ECC remainder. Upon reading the data again, it verifies that the data error was corrected, and that a correctable ECC error bit was posted. This time it creates 12 consecutive errors in the data and writes it and the ECC remainder. Upon reading the data back again, it verifies that the hard ECC error bit was posted. The 9400 uses a 56 bit ECC check here, whereas the 9800 uses a 32 bit ECC check.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents

4.4 (AV) Address Verify Test

Execution time:	14 seconds (approximately).
Bits transferred:	3.28×10^4 bits.
Prerequisite:	None.
Type of emulation:	Extended.
Data pattern:	'000000 and '177777.

AV checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It checks the ability of the system to detect a discrepancy between the address written in the header, and the actual physical address of the sector being accessed. It does this by writing an incorrect address in a header and then attempting to read the sector, verifying that the address verify error bit sets. It performs the test four times, setting errors in each of the four elements of the header address: Head, Sector, Cylinder hi, and Cylinder lo.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.5 (DV) Data Verify Test

Execution time:	1 second.
Bits transferred:	4096 bits.
Prerequisite:	None.
Type of emulation:	Standard.
Data pattern:	Current pattern.

DV verifies the operation of the write check function. It checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It writes 32 sectors of data, then performs a write check operation, and checks status. It alters the data in CPU memory and performs a write check operation again, verifying that the write check error bit sets.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.6 (SW) Single Word Transfer Test

Execution time:	1 second.
Bits transferred:	16 bits.
Prerequisite:	None.
Type of emulation:	Standard.
Data pattern:	'125252.

SW tests the system's ability to transfer less than a full sector of data, in this case a single word. It checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It fills the entire data buffer with '125252, but writes only a single word. It then clears the buffer, reads back a single word, and verifies that only that single word got read back into the buffer.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.7 (MW) Maximum Word Transfer Test

Execution time:	1 second.
Size of transfer:	Depending on the memory size and the length of the command string, will transfer 11-12 sectors worth.
Prerequisite:	None.
Type of emulation:	Standard.
Data pattern:	Current pattern.

MW tests the system's ability to handle large multi-sector data transfers. It checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It sizes the available memory space in the CPU and makes the largest data transfer possible. Since the memory space is variable depending on the size of the command string and the amount of memory available, the test prints out the size of the data transfer for the operator's information. The test formats and individually writes data in enough sectors to cover the transfer. It then reads all the data in a single multi-sector read.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.8 (HS) Head Switch Test

Execution time:	4 seconds.
Bits transferred:	2.95 x 10^5 bits.
Prerequisite:	None.
Type of emulation:	Standard.
Data pattern:	'125252, '000000, '177777.

HS tests the system's ability to perform multi-sector transfers across head boundaries. It checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. First, it individually writes a sector on either side of the boundary, then does a 2-sector read of the two sectors. Then it does a 2-sector write across a head boundary and performs two single sector reads of the same sectors. It repeats this over all the head boundaries.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.9 (HD) Header Test

Execution time:	14 seconds.
Bits transferred:	3.28 x 10^4 bits.
Prerequisite:	None.
Type of emulation:	Extended.
Data pattern:	'000000 and '177777.

HD checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It checks the ability of the system to detect a discrepancy between the address written in the header, and the actual physical address of the sector being accessed. It does this by writing an incorrect address in a header and then attempting to read the sector, checking that the address verify bit sets. It performs the test four times, setting errors in each of the four elements of the header address: Head, Sector, Cylinder hi, and Cylinder lo.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.10 (CS) Cylinder Switch Test

Execution time:	1 second.
Bits transferred:	6.55 x 10^4 bits.
Prerequisite:	None.
Type of emulation:	Standard.
Data pattern:	'125252, '000000, '177777.

CS checks the known Bad Block tables written on the last track and selects an area free of known bad blocks. It tests the system's ability to perform multi-sector transfers across cylinder boundaries. First, it individually writes a sector on either side of the boundary, then does a 2-sector read of the two sectors. Then it does a 2-sector write across a cylinder boundary and performs two single sector reads of the same sectors. It repeats this over four different cylinder boundaries.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.11 (FT) Format Test

Execution time:	5 minutes, 15 seconds (80 megabyte).
Bits transferred:	5.44×10^8 bits written. 5.39×10^8 bits read.
Prerequisites:	None.
Type of emulation:	Standard.
Data pattern:	Current pattern.

FT formats the disk between the limits set by the operator, using the largest multi-sector transfers that memory size will allow. It reads the headers in a mass transfer, checking status. Any error occurring at a known bad block is ignored. The table below shows the header information:

	RP04		RM03
	HEADER		HEADER
Wd 1	'100000+Cylinder	Wd 1	'150000+cylinder
Wd 2	Head-Sector	Wd 2	Head-sector
Wd 3	'000000		
Wd 4	'000000		
	DATA		DATA
Wd 1	'000001	Wd 1	'000001
Wd 2	'000000	Wd 2	'000000
Wd 3	'000000	Wd 3	'000000
Wd 4	'000000	Wd 4	'000000
Wd 5	Current Pattern	Wd 5	Current Pattern
| |	| |		| | | |
Wd 256	Current Pattern	Wd 256	Current Pattern

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

NOTE

When formatting a pack for system use, use the FM command (instead of this FT test). This will automatically write the correct data for proper handling of bad blocks. (See Section 1.3.4.)

4.12 (OT) Oscillating Track Test

Execution time:	7 minutes, 5 seconds (80 Megabyte).
Bits transferred:	1.69 x 10^7 bits.
Prerequisite:	The disk must first be formatted (use the FT test, the WH command or the FM command).
Type of emulation:	Standard.
Data pattern:	Current pattern.

This is a data reliability test wherein data is written on ascending cylinder addresses, and read from oscillating cylinder addresses. Data is read one sector at a time, always from sector 0 of any head or cylinder. First from the lowest cylinder, lowest head, then from the highest cylinder, highest head. After each read, the lower head address is incremented and the upper head address is decremented. The test reads back and forth between the upper and lower addresses until the two meet in the middle. Known bad blocks are ignored.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.13 (RT) Random Track Test

Execution time:	10 minutes (80 Megabyte).
Bits transferred:	2.70 x 10^8 bits.
Prerequisite:	The disk must first be formatted (use the FT test, the WH command or the FM command).
Type of emulation:	Standard.
Data pattern:	Random data.

Using the current data pattern as the initial seed, RT writes random word counts of random data to random cylinder, head and sector addresses within the test domain. It clears the memory buffer, then reads back the data just written. If error status is OK, RT does a word by word check of the data. Known bad blocks are ignored.

APPLICABLE ERROR REPORT OPTIONS:

EM: EM gives an error message from the test which explains the contents.

4.14 (WH) Write Header Utility

Execution time:	3 minutes, 14 seconds (80 Megabyte).
Bits transferred:	5.44×10^8 bits Write 0 bits Read.
Prerequisite:	None.
Type of emulation:	Standard.
Data Pattern:	Current pattern.

This command allows the operator to write headers and data. The operator can write header for a single sector, or for any contiguous group of sectors, or for all the sectors on a disk. The command sizes the available memory space to perform the largest multi-sector write possible. It writes header and data in all sectors between the lower and upper address limits.

4.15 (ZR) Zig-Zag Read Utility

This is identical to the read portion of the OT test. (See Section 4.12).

5.0 BOOTSTRAP PROGRAM

When the routine is called, the following message will appear on the console terminal:

```
BOOT V8.2     26-AUG-80

BOOT UNIT:-
```

The operator may enter ? if help is required and a help message will appear. The operator uses this one entry point either to boot a device or write the 94DIAG program to a Magtape. There are two formats he may use to do this: the two-character mnemonic and the four-character mnemonic.

Two-character mnemonic:

> The device to be booted from may contain any operating system having a boot 0 boot starting at address 0.

Four-character mnemonic:

> The device must contain a System Industries-modified RT-11 system. At boot-up, word 474 and 476 will be modified with the selected CSR and Vector addresses.
>
> If the device is:
>
> RP04, RP05, RP06
> RM03
> 9400
> 9500
>
> then the unit number which follows will be treated as a logical rather than physical unit number.

5.1 Two-Character Mnemonic Boot Format

The format for a two-character mnemonic boot is:

```
XXN:SSSSSS
```

Where:	XX	= Mnemonic
	N	= Physical unit number (default = 0)
	SSSSSS	= CSR address (default given below)

The two-character boot mnemonics are:

MNEMONIC	TYPE	CSR ADDRESS
AR	RK05 or Aries	170400
DB	9400	176700
DK	RK05 or Aries	177400
DP	RP02 or RP03	176710
DR	9400	176300
KA	9500	174700
MM	TU16	172440
MT	TU10	172520
MX	TU16 PE	172440
RB	9400	176700
RK	RK05 or Aries	177400
RP	RP02 or RP03	176710
RR	9400	176300
SF	4500	166100
ST	3500	166100

The two-character mnemonics default to the CSR address shown. Use them if the operating system disk has been generated for the CSR given above. If the operating system to be booted does not recognize the CSR address for the desired device, use the four-character mnemonic.

5.2 Two-Character Mnemonic Write Tape Format

The format for a two-character mnemonic write tape is:

```
XXN:
```

Where: XX = Mnemonic
N = Physical unit number (default = 0)

The two-character write tape mnemonics are:

MNEMONIC	TYPE	CSR ADDRESS
WM	TU16 Write	172440
WT	TU10 Write	172522
WX	TU16 PE Write	172440

5.2.1 Writing Bootable Magtapes

The operator must place a blank tape in the drive, then enter the command. The tape will rewind if required, then a filename will be written out, followed by the boot block. Both the filename and the boot block are 256 words long. Following these, the program will be written. Upon completion of the write, the following message will appear:

```
* WRITTEN *
```

5.2.2 Booting From TU10 or TU16 Systems

The program will then return to the entry point. The program written on the tape will be bootable from either TU10 or TU16 tape systems if written at 800 BPI.

To return to the diagnostic, CTL I is used.

5.3 Two-Character Mnemonic Write Disk Format

The format for a two-character mnemonic write disk is:

```
XXN:SSSSSS/CCCC
```

Where:		
	XX	= Mnemonic
	N	= Physical unit number (default = 0)
	CCCC	= Octal cylinder number (default = 0)
	SSSSSS	= CSR address (default given below)

The two-character write disk mnemonics are:

MNEMONIC	TYPE	CSR ADDRESS
WA	RK05/Aries write	170400
WB	9400 write	176700
WD	RK05/Aries write	177400
WK	9500 write	174700
WR	9400 write	176300

5.3.1 Writing Bootable Disks

The operator must spin up a formatted disk on the drive, then enter the command. This writes a bootable memory image of 94DIAG starting at the first block of the specified cylinder. Specifying cylinder 0 allows a standard hardware bootstrap to be used for loading the program.

To return to the diagnostic, hit CTL I.

5.3.2 Booting From The Disk Produced

The bootstrap program, whether hardware or hand-loaded, must read the first block of the specified cylinder into memory starting at location 000000, then without issuing a RESET command, start execution at location 000000. See Section 5.5 for examples of hand-loaded bootstraps.

5.4 Four-Character Mnemonic Format

The format for a four-character mnemonic is:

XXXXN:SSSSSS,VVV

Where:		
XXXX	=	Mnemonic
N	=	Logical unit number (default = 0)
SSSSSS	=	CSR address (default given below)
VVV	=	Vector (default given below)

The four-character mnemonics are:

MNEMONIC	TYPE	CSR	VECTOR
AR05	RK05 or Aries	170400	160
KA60	9500 80 MB Drive	174700	170
KA62	9500 with 9762 Drive	174700	170
KA66	9500 with 9766 Drive	174700	170
RK05	RK05 or Aries	177400	220
RM03	9400	176300	150
RM66	9400 with 9766 Drive	176300	150
RP04	9400 with 9762 Drive	176700	254
RP06	9400 with 300 MB Drive	176700	254
RP62	9400 with 9762 Drive	176700	254
RP66	9400 with 9766 Drive	176700	254
3500	3500	166100	220
4500	4500	166100	220

For the disk bootstraps, block 0 of the requested unit is read into memory location 0 and then executed from that location.

For the Magtape bootstraps, the first record of the tape (identity record) is skipped and the second record (256. words) is read into memory address 0 and then executed from that location.

To return to the diagnostic, CTL I is used.

5.5 Handloaded Bootstraps

The operator must use the following routines when a hardware bootstrap is not available or when the program has been loaded to a cylinder other than 0.

9400 (RP04, RP05, RP06, RM03, RM05):

LOCATION	CONTENTS
001000	012700
001002	A
001004	012710
001006	B
001010	012760
001012	000023
001014	177770
001016	012760
001020	C
001022	000024
001024	012740
001026	000000
001030	012740
001032	000000
001034	012740
001036	177400
001040	012740
001042	000071
001044	105710
001046	100376
001050	005007

A = Address of Drive Status Register (176710 is standard).
B = Unit number (Bits 0-2)
C = Cylinder number.

<u>RK05 OR ARIES</u>:

LOCATION	CONTENTS
001000	012700
001002	A
001004	012710
001006	B
001010	012740
001012	000000
001014	012740
001016	177400
001020	012740
001022	000005
001024	105710
001026	100376
001030	005007

A = Address of Disk Address Register (177412 is standard).
B = Bits 0-3: Sector number (normally 0).
Bit 4: Head number (normally 0).
Bits 5-12: Cylinder number.
Bits 13-15: Unit number.

<u>9500</u>:

LOCATION	CONTENTS
001000	012700
001002	A
001004	012740
001006	000000
001010	012740
001012	000000
001014	012740
001016	B
001020	012740
001022	000400
001024	012740
001026	000005
001030	105710
001032	100376
001034	005007

A = Address of the Error Register (176712 is standard).
B = Bits 0-9: Cylinder number.
Bits 10-12: Unit number.

6.0 ERROR MESSAGES

The following lists the error message numbers and their definitions.

1. Timed out trying to initialize the controller.
2. Timed out trying a random write operation.
3. Timed out trying a random read operation.
4. Timed out trying an incrementing write operation.
5. Timed out trying to read.
6. Error status occurred after a read operation.
7. Error status after read was OK, but the data read back did not compare to that written.
8. Timed out trying to write two single headers spanning a cylinder boundary.
9. Error status occurred after a two-sector read across a cylinder boundary.
10. Timed out trying to do a two-sector read across a cylinder boundary.
11. Data was written across a cylinder boundary in two single sector write operations. Data was read in a single two-sector read operation. The data read back does not match that written.
12. Error status occurred while doing a two-sector read across a cylinder boundary.
13. Timed out trying to do a two-sector write across a cylinder boundary.
14. Error status occurred while doing a two-sector write across a cylinder boundary.
15. Timed out while trying the first 1-sector read of data written in a single two-sector write.
16. Data was written in a single two-sector write operation. Data was read in two single-sector read operations. The data read from the first sector does not match that written.

17. Data was written in a single two-sector write operation. Data was read in two single-sector read operations. Error status occurred during the first single-sector read.

18. Data was written in a single two-sector write operation. Data was read in two single-sector read operations. Controller timed out while trying the second single-sector read.

19. Data was written in a single two-sector write operation. Data was read in two single-sector read operations. The data from the second single-sector read does not match that written.

20. Data was written in a single two-sector write operation. Data was read in two single-sector read operations. Error status occurred during the second single-sector read.

21. Timed out trying to write two single headers and data spanning a head boundary.

22. Error status during two single-sector writes across a head boundary.

23. Timed out trying a two-sector read across a head boundary.

24. Wrote data in two single-sector writes across a head boundary. Read data in a single two-sector read. The data read does not match that written.

25. Error status occurred during a two-sector read across a head boundary.

26. Timed out trying a two-sector write across a head boundary.

27. Error status occurred during a two-sector write across a head boundary.

28. Did a two-sector write across a head boundary. Timed out trying to do the first single-sector read.

29. Did a two-sector write across a head boundary. Read data in two single-sector reads. The data in the first single-sector read does not match that written.

30. Did a two-sector write across a head boundary. Did two single-sector reads. Error status occurred during the first single-sector read.

31. Did a two-sector write across a head boundary. Read data in two single-sector reads. Timed out reading the second sector.

32. Did a two-sector write across a head boundary. Read data in two single-sector reads. The data read from the second sector does not match that written.

33. Did a two-sector write across a head boundary. Read data in two single-sector reads. Error status occurred during the second read.

34. Timed out trying to write header and data.

35. Error status occurred while writing a single header and data.

36. Timed out trying a maximum-sized multi-sector read.

37. Data read in the maximum-sized multi-sector read does not match that written in multiple single-sector writes.

38. Error status occurred during a maximum-sized multi-sector read.

39. Timed out trying to write a single word.

40. Error status occurred while writing a single word.

41. Timed out trying to read a single word.

42. Wrote a single word, then read it back. The word read back does not match that written.

43. Error status occurred while reading a single word.

44. Did a single-word transfer. The single word was OK, but the fill characters were not all zeros.

45. Timed out tring to put the header address into the RAM buffer.

46. Timed out trying to load zeros into the ram buffer.

47. Timed out trying to write header & data using extended emulation.

48. Error status occurred in ECIB table after writing header and data.

49. Timed out trying to put an incorrect header address into the RAM buffer.

50. Timed out trying to write an incorrect header using extended emulation.

51. Timed out trying to read the sector with the incorrect header, using extended emulation.

52. The "ADDRESS Verify" error bit did not set in ECOS2 after reading an incorrect header.

53. Timed out trying to load all 1's into the RAM buffer.

54. Timed out tring to write data into a sector with an incorrect header.

55. The "Address Verify" error bit did not set in ECOS2 after trying to write into a sector with an incorrect header.

56. Timed out trying to load the correct header into the RAM buffer.

57. Timed out tring to write the correct header using extended emulation.

58. Error status in ECIB table after writing correct header.

59. Timed out trying to load RAM with data.

60. Timed out trying to read the sector with the corrected header.

61. Error status occurred in the ECIB table after reading the sector with the corrected header.

62. Unexpected data was read into the RAM buffer from the disk. The data in the sector was originally all zeros. The test attempted to write 177777 when the sector had an incorrect header. If the RAM contains 177777, this indicates that the controller wrote into the sector in spite of the incorrect header. If the RAM contains 125252, this means no read took place after the header had been corrected. The RAM buffer should contain all zeros at this point.

63. Timed out while attempting to write headers.

64. Showed error status after writing headers.

65. Timed out while attempting to write data.

66. Error status occurred after writing data.

67. Timed out attempting to do a write check of good data.

68. The "Write Check" error bit in RPCS2 set when no write check error was intended. Possibly caused by bad surface on the disk.

69. The "Write Check" error bit in RPCS2 did not set when there was a write check error.

70. Timed out trying to load header into RAM buffer.

71. Timed out trying to load data into the RAM buffer.

72. Timed out trying to write header and data using extended emulation.

73. Error status in ECIB table after writing header and data.

74. Timed out trying to fill RAM buffer with zeros.

75. Timed out trying to read data and the ECC polynomial.

76. Error status in the ECIB table after reading the data and the ECC polynomial.

77. Timed out trying to get data from the RAM buffer.

78. The hardware and software ECC polynomials do not compare.

79. Timed out trying to read or write into or from the RAM buffer.

80. Timed out trying to load the correctable error into the RAM buffer.

81. Timed out trying to write data with correctable errors (11 errors).

82. Error status in ECIB table after writing data with correctable errors.

83. Timed out trying to fill RAM buffer with zeros.

84. Timed out trying to read the corrected data.

85. The "Soft ECC Error" bit did not set after a soft ECC error occurred.

86. The correctable errors in the data did not get corrected, even though ECOS2 indicated a soft ECC error status.

87. Timed out trying to read data from the RAM buffer.

88. Timed out trying to load the RAM buffer with data containing an uncorrectable ECC error.

89. Timed out trying to read data from RAM buffer.

90. Timed out trying to load data with an uncorrectable error in it into the RAM.

91. Timed out trying to write a sector with an uncorrectable error in it.

92. Error status in the ECIB table after writing a sector with an uncorrectable error in it.

93. Timed out trying to read the sector with an uncorrectable error in it.

94. The "ECC Error" bit in ECOS2 did not set after reading data with an uncorrectable error in it.

95. Error status occurred while writing header and data.

108. Timed out trying to load header into RAM buffer.

109. Timed out trying to store data into RAM buffer.

110. Timed out trying to load header and data into the RAM buffer in order to set the bad block flag in the header.

111. Error status in the ECIB table after writing header and data to set the bad block flag in the header.

112. Timed out trying to load data into RAM buffer.

113. Timed out trying to write into a sector with the bad block flag set in the header.

114. Did not get "Bad Block" status in ECOS3 when attempting to write into a bad block (extended emulation).

115. Timed out trying to read a sector with the bad block flag set in the header.

116. Did not get "Bad Block" status in ECOS3 when attempting to read a bad block (extended emulation).

117. Timed out trying to check data in RAM buffer.

118. Timed out trying to load header into RAM buffer.

119. Timed out trying to write the header with the bad block flag removed.

120. Error status in ECIB table after writing header with the bad block flag cleared.

121. Timed out trying to read a sector after the bad block flag has been cleared.

122. The "Bad Block" bit in ECOS3 still sets after re-writing the header with the bad block flag cleared.

123. Timed out trying to check data in RAM buffer.

124. Was able to read the sector in spite of the bad block flag set in the header. The data in the sector was all zeros. The data in the RAM was '125252 before the read. The date in the RAM should have remained unchanged. Interpretation of the error report:

GD-BAD = '000252-'000000:	Read sector with bad block flag set.

125. Wrote into a sector with the bad block flag set. The test originally wrote zeros, then set the bad block flag. It then attempted to write ones in that sector. It removed the bad block flag by re-writing the header. Then, it filled the RAM with 125252 data, and read the sector back. If the RAM still contains 125252, then the read did not actually take place. If the RAM contains all ones, then the system wrote into the sector in spite of the bad block flag. Interpretation of the error summary:

GD-BAD = '000000-'000377	Data over written.
GD-BAD = '000000-'000252	Bad read.

126. Timed out trying to load header into RAM buffer.

127. Timed out trying to write header.

128. Error status in ECIB table after writing header.

129. Timed out trying to load data into RAM buffer.

130. Timed out trying to read header and CRC word.

131. Timed out trying to get header from RAM buffer.

132. Timed out trying to get header from RAM buffer.

133. Timed out trying to get CRC from RAM buffer.

134. Timed out trying to get CRC from RAM buffer.

135. Hardware and software CRC words do not match each other.

136. Timed out trying to load an erroneous CRC word into the RAM buffer.

137. Timed out trying to write a header with an erroneous CRC word.

138. Timed out trying to read a header with an erroneous CRC word.

139. The "CRC Error" bit in ECOS2 did not set after reading a header with an erroneous CRC word.

140. Timed out trying to load the cylinder most significant byte of the header into the RAM buffer.

141. Timed out trying to load the cylinder least significant byte of the header into the RAM buffer.

142. Timed out trying to load the head address of the header into the RAM buffer.

143. Timed out trying to load the sector address of the header into the RAM buffer.

144. Timed out trying to load the two key words of the header into the RAM buffer.

145. Error status occurred after trying to write random data to a random address with a random word count.

146. Error status occurred after trying to read random data from a random address with a random word count.

147. Status was OK after reading random data from a random address with a random word count, but one or more of the words read back did not compare with that written.

201. Timed out trying to initialize the controller.

202. An error was detected in register RPCS1.

203. An error was detected in register RPWC.

204. An error was detected in register RPBA.

205. An error was detected in register RPDA.

206. An error was detected in register RPER1.

207. An error was detected in register RPAS.

208. An error was detected in register RPLA.

209. An error was detected in register RPDB.

210. An error was detected in register RPMR.

211. An error was detected in register RPDT.

212. An error was detected in register RPSN.

213. An error was detected in register RPOF.

214. An error was detected in register RPDC.

215. An error was detected in register RPCC.

216. An error was detected in register RPER2.

217. An error was detected in register RPER3.

218. An error was detected in register RPEC1.

219. An error was detected in register RPEC2.

220. An error was detected in register RHBAE.

221. An error was detected in register RPCS3.

222. "Medium On Line", "Data Ready", or "Volume Valid" bits in RPDS are stuck at zero.

223. "Drive Ready" bit in RPDS stuck at zero.

224. "Volume Valid" bit in RPDS stuck at zero.

225. "Attention Active" bit in RPCS1 stuck at one.

226. "Special Attention" or "Transfer Error" bit in RPCS1 stuck at one.

227. "Ready" bit in RPCS1 stuck at zero.

228. "Ready" bit in RPCS1 stuck at zero.

229. "Attention Active" bit in RPDS stuck at zero.

230. "Error" bit in RPDS stuck at one.

231. "Ready" bit in RPCS1 stuck at zero.

232. "Go" bit in RPCS1 stuck at zero.

233. "Go" bit in RPCS1 stuck at one.

234. "Program Error" bit in RPCS2 stuck at one.

235. "Program Error" bit in RPCS2 stuck at zero.

236. Timed out trying to initialize controller.

237. Timed out waiting for "Ready" bit in RPCS1.

238. "Special Condition" or "Transfer Error" bit in RPCS1 stuck at zero.

239. "Error" bit in RPDS stuck at zero.

240. Unexpected interrupt occurred at priority levels greater than six.

241. Unexpected interrupt occurred at priority levels greater than five.

242. No interrupt received when expected at priority levels greater than three.

243. Multiple interrupts occurring.

244. Microprocessor did not respond to interrupt.

245. Timed out trying to load the RAM buffer.

246. Half-write CPU failed.

247. Timed out trying to fill the RAM buffer.

248. RAM data buffer failure.

249. Ready bit did not set after attempted write to an illegal cylinder.

401. MPU did not respond to a recal command.

402. "On Cylinder" bit did not set after recal command.

403. Unexpected status after recal. Check ECUS1: "Dual Port Busy", "Seek Error", "Fault" bits.

404. Unexpected status after recal. Check ECUS1: "On Cylinder", "Unit Select", and "Attention" bits.

405. MPU did not respond to a seek command.

406. "On Cylinder" or "Seek Error" bit did not set after a seek.

407. MPU did not respond to recal command.

408. "On Cylinder" bit did not set after recal.

409. MPU did not respond to a seek command.

410. MPU did not respond to an RMC command.

411. "On Cylinder" bit did not clear during a long seek.

412. MPU does not respond to a WMC command.

413. The highest sector count detected does not match that given for this model.

414. Unexpected status after a recal command.

415. "On Cylinder" bit did not set after a recal.

416. Unexpected status after a seek command.

417. Unexpected status after a seek command.

418. Unexpected status after a seek command.

419. "On Cylinder" bit did not set after a seek.

420. Unexpected status after a seek command to lowest cylinder.

421. Unexpected status after a seek command to highest cylinder.

422. Unexpected status after a seek command from highest to lowest cylinder.

423. Unexpected status after a recal command.

424. "On Cylinder" bit did not set after a seek.

425. Unexpected status after a seek command to cylinder #1.

426. Unexpected status after a seek command to cylinder #0.

427. Unexpected status after a seek command to a higher cylinder.

428. Unexpected status after a recal command.

429. "On Cylinder" bit did not set after a recal.

430. Unexpected status after a seek to a random address.

431. MPU did not respond to a WMC command.

432. MPU did not respond to a half-write drive command.

433. "Ready" bit in RPCS1 did not set after write.

434. MPU did not respond to WMC command.

435. MPU did not respond to half-read drive command.

436. "Ready" bit in RPCS1 did not set after read.

437. Data from half-read drive does not compare with that written.

System Industries 1855 Barber Lane, Milpitas, California 95035, (408) 942-1212

SI Computer Systems, Ltd.
Guildford Road
Woking, Surrey
GU22 7QQ, England
Telephone: (44) 4862 5077
Telex: 859124

System Industries GmbH
Am Forsthaus Gravenbruch 9-11
D6078 Nue Isenburg 2
West Germany
Telephone: (49) 6102 5464/5
Telex: 417606

System Industries, Inc., Scandinavia AB
Grev Magnigatan 5
S-11455 Stockholm
Sweden
Telephone: (46) 0863 6274
Telex: 12697