

.REM @

IDENTIFICATION  
-----

PRODUCT CODE: AC-S830E-MC  
PRODUCT NAME: CZUDCEO UDA & DISK DRV DIAG  
PRODUCT DATE: 04-OCT-83  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: MATT TEDONE

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1981, 1982, 1983 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

.REM @

TABLE OF CONTENTS

	Page
1.0 GENERAL INFORMATION	3
1.1 PROGRAM ABSTRACT	3
1.2 SYSTEM REQUIREMENTS	4
2.0 OPERATING INSTRUCTIONS	5
2.1 COMMANDS	5
2.2 SWITCHES	6
2.3 FLAGS	7
2.4 HARDWARE QUESTIONS	7
2.5 SOFTWARE QUESTIONS	9
2.6 EXTENDED P-TABLE DIALOGUE	11
2.7 QUICK STARTUP PROCEDURE	13
3.0 ERROR INFORMATION	16
3.1 TYPES OF ERROR MESSAGES	16
3.2 SPECIFIC ERROR MESSAGES	18
3.2.1 HOST PROGRAM ERROR MESSAGES (00001 TO 00999)	18
3.2.2 TEST 1 ERROR MESSAGES (01000 TO 01999)	29
3.2.3 TEST 2 INFORMATIONAL MESSAGES	32
3.2.4 TEST 2 ERROR MESSAGES (02000 TO 02999)	33
3.2.5 TEST 3 INFORMATIONAL MESSAGES	43
3.2.6 TEST 3 ERROR MESSAGES (03000 TO 03999)	44
3.2.7 TEST 4 INFORMATIONAL MESSAGES	54
3.2.8 TEST 4 ERROR MESSAGES (04000 TO 04999)	55
3.2.9 SPECIAL DEVICE FATAL (05000)	79
3.3 TEST 4 RETRY/RECOVERY METHODS	81
3.4 DEC STANDARD 166 EXCERPTS	94
3.4.1 THE REPLACEMENT AND CACHING TABLES	94
3.4.2 FCT STRUCTURE	96
4.0 PERFORMANCE AND PROGRESS REPORTS	99
5.0 TEST SUMMARIES	101
5.1 TEST # 1 - UNIBUS ADDRESSING TEST	101
5.2 TEST # 2 - DISK RESIDENT DIAGNOSTIC TEST	103
5.3 TEST # 3 - DISK FUNCTION TEST	105
5.4 TEST # 4 - DISK EXERCISER	106

## 1.0 GENERAL INFORMATION

## 1.1 PROGRAM ABSTRACT

This is the only diagnostic program provided for testing the UDA-50 Unibus Disk Controller and the disk drives connected to it. There are four tests within this diagnostic:

- Test # 1 - Unibus Addressing Test. Runs the UDA-50 ROM resident diagnostics, then further tests the Unibus address interface.
- Test # 2 - Disk Resident Diagnostic Test. Executes the diagnostics in each disk drive.
- Test # 3 - Disk Function Test. Functionally tests each disk drive to ensure the disk can seek, read, write and format.
- Test # 4 - Disk Exerciser. Exercises the disk drives in a manner similar to normal operating systems. This test should be used to gain confidence in the reliability of the disk drive.

This program is designed to handle all future disk drives that are attached to the UDA-50 without modifying or rereleasing. This is possible because the disk drives are programmed to tell this diagnostic about all their characteristics that make them different from other drives, such as number of cylinders, sectors per cylinder, etc.

Two other VDP-11 diagnostic programs are provided for the UDA-50 disk subsystem:

CZUDEDO - UDA-50 Disk Drive Formatter.

CXUDFFD - UDA-50 Disk Drive Formatter Data File

DEC 111 - Unibus Exerciser can be run on the UDA-50 using the UDA-50 module DUBCO.

This diagnostic has been written for use with the Diagnostic Runtime Services Software (Supervisor). These services provide the interface to the operator and to the software environment. For a complete description of the Runtime Services, refer to the XXDP User's Manual. There is a brief description of the Runtime Services in section 2 of this document.

This diagnostic will test UDA-50's with modules M7485 and M7486. Whenever a fault is detected in a UDA-50 and the fault can be isolated to one of the two modules in the UDA-50, Replace that module.

## 1.2 SYSTEM REQUIREMENTS

-----

This program was designed using the PDP-11 Diagnostic Runtime Services revision C. Run time environments are determined by the Runtime Services and may change as new versions of the Services are developed. This program requires the following:

- PDP-11 Unibus processor
- 28K words of memory (minimum)
- Console terminal
- XXDP+ load media containing this program and the ZUDDE0.PAK data file
- One or more UDA50 subsystems. The subsystem controller must be type UDA50-A with microcode level 3 or greater.
- Line clock - either Type L or P

The line clock is used for all timed loops in the program. The diagnostic will run on a system with no clock but will hang whenever an event for which the program is waiting does not happen (i.e., a time-out error message will not result).

This diagnostic program requires that the data file ZUDDE0.PAK be on the XXDP+ system device. This data file is ordered under the name CZUDDE0. The XXDP+ system device must remain on-line during the execution of this diagnostic.

## 2.0 OPERATING INSTRUCTIONS

This section contains a brief description of the Runtime Services. For detailed information, refer to the XXDP+ User's Manual (CHQUS).

### 2.1 COMMANDS

There are eleven legal commands for the Diagnostic Runtime Services (Supervisor). This section lists the commands and gives a very brief description of them. The XXDP+ User's Manual has more details.

COMMAND	EFFECT
START	Start the diagnostic from an initial state
RESTART	Start the diagnostic without initializing
CONTINUE	Continue at test that was interrupted (after tC)
PROCEED	Continue from an error halt
EXIT	Return to XXDP+ Monitor (XXDP+ OPERATION ONLY!)
ADD	Activate a unit for testing (all units are considered to be active at start time)
DROP	Deactivate a unit
PRINT	Print statistical information (see section 4.0)
DISPLAY	Type a list of all device information
FLAGS	Type the state of all flags (see section 2.3)
ZFLAGS	Clear all flags (see section 2.3)

A command can be recognized by the first three characters. So you may, for example, type "STA" instead of "START".

2.2 SWITCHES  
 -----

There are several switches which are used to modify supervisor operation. These switches are appended to the legal commands. All of the legal switches are tabulated below with a brief description of each. In the descriptions below, a decimal number is designated by "DDDDD".

SWITCH	EFFECT
/TESTS:LIST	Execute only those tests specified in the list. List is a string of test numbers, for example - /TESTS:1:5:7-10. This list will cause tests 1,5,7,8,9,10 to be run. All other tests will not be run.
/PASS:DDDDD	Execute DDDDD passes (DDDDD = 1 to 64000)
/FLAGS:FLGS	Set specified flags. Flag- are described in section 2.3.
/EOP:DDDDD	Report end of pass message after every DDDDD passes only. (DDDDD = 1 to 64000)
/UNITS:LIST	TEST/ADD/DROP only those units specified in the list. List example - /UNITS:0:5:10-12 use units 0,5,10,11,12 (unit numbers = 0-63).

Example of switch usage:

START/TESTS:1-5/PASS:1000/EOP:100

The effect of this command will be: 1) tests 1 through 5 will be executed, 2) all units will tested 1000 times and 3) the end of pass messages will be printed after each 100 passes only. A switch can be recognized by the first three characters. You may, for example, type "/TES:1-5" instead of "/TESTS:1-5".

Below is a table that specifies which switches can be used by each command.

	TESTS	PASS	FLAGS	EOP	UNITS
START	X	X	X	X	X
RESTART	X	X	X	X	X
CONTINUE		X	X	X	
PROCEED			X		
DROP					X
ADD					X
PRINT					
DISPLAY					X
FLAGS					
ZFLAGS					
EXIT					

### 2.3 FLAGS

Flags are used to set up certain operational parameters such as looping on error. All flags are cleared at startup and remain cleared until explicitly set using the flags switch. Flags are also cleared after a START or RESTART command unless set using the flag switch. The ZFLAGS command may also be used to clear all flags. With the exception of the START, the RESTART and ZFLAGS commands, no commands affect the state of the flags; they remain set or cleared as specified by the last flag switch.

FLAG	EFFECT
HOE	Halt on error - control is returned to runtime services command mode
LOE	Loop on error
IER*	Inhibit all error reports
IBE*	Inhibit all error reports except first level (first level contains error type, number, PC, test and unit)
IXE*	Inhibit extended error reports (those called by PRINTX macro's)
PRI	Direct messages to line printer
PNT	Print test number as test executes
BOE	"BELL" on error
UAM	Unattended mode (no manual intervention)
ISR	Inhibit statistical reports
IDU	Inhibit program dropping of units
LOT	Loop on test

\*Error messages are described in section 3.1

See the XXDP\* User's Manual for more details on flags. You may specify more than one flag with the FLAG switch. For example, to cause the program to loop on error, inhibit error reports and type a "BELL" on error, you may use the following string:

```
/FLAGS:LOE:IER:BOE
```

### 2.4 HARDWARE QUESTIONS

When a diagnostic is STARTed, the Runtime Services will prompt the user for hardware information by typing "CHANGE HW (L) ?". When you answer this question with a "Y", the Runtime Services will ask for the number of units (in decimal). You will then be asked the following questions for each unit. When you answer this question with an "N", the Runtime Services will use the answers built into the program by the SETUP utility (see chapter 6 of the XXDP\* User's Manual). If you have never run the SETUP utility on this program file, the default values listed below (just before the question mark) will be used.

UNIBUS ADDRESS OF UDA (O) 172150 ?

Answer with the address of the UDAIP register of one UDA as addressed by the processor with memory management turned off (i.e., an even 16-bit address in the range of 160000 to 177774).

VECTOR (O) 154 ?

Answer with the interrupt vector address of the UDA. A vector address in the range of 4 to 774 may be specified. The UDA does not have a vector "hard wired" to it, so any vector not being used by this program and XXDF+ may be used.

RR LEVEL (D) 5 ?

Answer with the interrupt priority used by the UDA. Levels 4 to 7 are accepted. This level must match the level "hard wired" in the UDA by the priority plug.

UNIBUS BURST RATE (D) 63 ?

The UDA allows the ability to control the maximum number of words transferred across the UNIBUS each time the UDA becomes master. The default answer of 63 will allow for the fastest execution of this diagnostic program. You may answer with the value your operating system uses or use zero which will tell the UDA to supply a value that should work on any system. A decimal number in the range of 0 to 63 may be specified and all values should work on any system. A larger value will allow for a faster running program. The value will be passed directly to the UDA during initialization.

DRIVE NUMBER (D) 0 ?

Answer with the drive number of the drive you wish to test. This is the number which appears on the "unit plug" on the front of the disk drive. On a multi-unit drive, each sub-unit on the drive must be tested as a separate unit to completely test the drive. A maximum of eight logical drives may be tested on one UDA at a time (UDA configuration limit).

EXERCISE ON CUSTOMER DATA AREA IN TEST 4 (L) N ?

Answer "N" to have test 4 (drive exerciser) run on the diagnostic area of the disk. Answer "Y" to run on the customer data area. A "Y" answer will destroy any customer data that may be on the disk. A warning message will be printed before testing begins if this question is answered "Y".

CUSTOMER DATA WILL BE DESTROYED ON:

UNIT	UDA AT	DRIVE
xx	xxxxxx	xxx

Unless the diagnostic is being run in unattended mode (i.e., START/FLAG:UAM command), a confirmation will also be required as follows:

ARE YOU SURE CUSTOMER DATA CAN BE DESTROYED (L) ?

If the above question is answered "N", the entire diagnostic will stop and the Runtime Services prompt will be displayed. No default answer is provided for this question.

## 2.5 SOFTWARE QUESTIONS

-----

After you have answered the hardware questions or after a RESTART or CONTINUE command, the Runtime Services will ask for software parameters. You will be prompted by "CHANGE SW (L) ?" If you wish to change any parameters, answer by typing "Y". The software questions and the default values are described in the next paragraphs.

ENTER MANUAL INTERVENTION MODE FOR SPECIAL DIAGNOSIS (L) N ?

Test: 2 and 4 have manual intervention modes which allow additional parameters to be input to alter the normal testing of a disk drive. This question should normally be answered "N" when this diagnostic is first run. Then, depending on the errors detected, it may be desirable to change this answer to "Y" and alter the testing to further isolate the problem. If this question is answered "Y", and the UAM (unattended mode operation) flag is set, tests 2 and 4 will print a warning message that the mode cannot be entered and will proceed as if answered "N". See the description of the individual tests in section 5 for more information.

REMAINING SOFTWARE QUESTIONS APPLY TO TEST 4 ONLY

This informational message is printed to describe the use of the remaining questions. If test 4 is not being run, a "CONTROL Z" can be typed to bypass them.

ERROR LIMIT (D) 32 ?

Enter the number of hard errors allowed before a drive is dropped from exercise by test #4. A number in the range of 1 to 65535 will be accepted.

READ TRANSFER LIMIT IN MEGABYTES - 0 FOR NO LIMIT (D) 0 ?

When the specified number of bytes have been read from a drive by test #4, the drive will be dropped from testing. When all drives are dropped, an end of pass will be indicated and the selected tests will be run again. This is the method used to determine how long test #4 is to run. Answer with a zero to prevent test from ending. The only other way test #4 can end is to have all drives dropped by use the error limit on each is exceeded. Of course, the operator can always stop test #4 by typing a control-C.

SUPPRESS PRINTING SOFT ERRORS (L) Y?

When test #4 needs to perform retries, soft error reports will be printed to give as much information as possible. These actions are considered normal operation and are not error conditions until the retries fail. When the test is being run only to see how reliable the drive performs, this question should be answered "Y" so they are not confused with hard errors. The number of these soft errors is always reported in the statistical report. Answer "N" to see all the soft error reports.

DO INITIAL WRITE ON START (L) Y ?

If test #4 is to do data compares, the drive will need to be written with data patterns readable by the program.

If the diagnostic area is selected for testing, the initial write is always performed (regardless of how this question is answered).

If the customer data area is selected for testing, the initial write will be performed when all of the following are true:

1. This question is answered "Y".
2. This is the first time test #4 is being run after a START command.
3. The disk is write enabled.

Answering this question "N" when testing on the customer data area will normally result in data comparison errors if the disk was not previously written by this diagnostic or the formatter.

Note that write checks are not performed during the initial write.

ENABLE ERROR LOG (L) N ?

A "Y" answer will cause error messages in test #4 to be stored in a log buffer. Once the log buffer is full, additional error information is lost. The contents of the log buffer will be printed when test #4 is stopped and a statistical report requested. This log feature is intended to allow the Digital Diagnosis Center (DDC) to start test #4 then hang up from the system and let it run for some period of time. DDC can call the system back later, type control-C, then CONT and see the errors that have occurred (up to the limit of the log buffer). A message will be printed to indicate no errors have occurred if the log buffer is empty. Test #4 will not be allowed to end while the error log is enabled until the error log is printed. The log buffer will hold 30 error messages when one disk unit is being tested. The log buffer will decrease in size as more units are tested.

2.6 EXTENDED P-TABLE DIALOGUE  
-----

When you answer the hardware questions, you are building entries in a table that describes the devices under test. The simplest way to build this table is to answer all questions for each unit to be tested. If you have a multiplexed device such as a mass storage controller with several drives or a communication device with several lines, this becomes tedious since most of the answers are repetitious.

To illustrate a more efficient method, suppose you are testing a fictional device, the XY11. Suppose this device consists of a control module with eight units (sub-devices) attached to it. These units are described by the octal numbers 0 through 7. There is one hardware parameter that can vary among units called the Q-factor. This Q-factor may be 0 or 1. Below is a simple way to build a table for one XY11 with eight units.

# UNITS (0) ? 8<CR>

UNIT 1  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 0<CR>  
Q-FACTOR (0) 0 ? 1<CR>

UNIT 2  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 1<CR>  
Q-FACTOR (0) 1 ? 0<CR>

UNIT 3  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 2<CR>  
Q-FACTOR (0) 0 ? <CR>

UNIT 4  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 3<CR>  
Q-FACTOR (0) 0 ? <CR>

UNIT 5  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 4<CR>  
Q-FACTOR (0) 0 ? <CR>

UNIT 6  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 5<CR>  
Q-FACTOR (0) 0 ? <CR>

UNIT 7  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 6<CR>  
Q-FACTOR (0) 0 ? 1<CR>

```
UNIT 8  
CSR ADDRESS (0) 160000<CR>  
SUB-DEVICE # (0) ? 7<CR>  
Q-FACTOR (0) 1 ? <CR>
```

Notice that the default value for the Q-factor changes when a non-default response is given. Be careful when specifying multiple units!

As you can see from the above example, the hardware parameters do not vary significantly from unit to unit. The procedure shown is not very efficient.

The Runtime Services can take multiple unit specifications however. Let's build the same table using the multiple specification feature.

```
# UNITS (0) ? 8<CR>
```

```
UNIT 1  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 0,1<CR>  
Q-FACTOR (0) 0 ? 1,0<CR>
```

```
UNIT 3  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 2-5<CR>  
Q-FACTOR (0) 0 ? 0<CR>
```

```
UNIT 7  
CSR ADDRESS (0) ? 160000<CR>  
SUB-DEVICE # (0) ? 6,7<CR>  
Q-FACTOR (0) 0 ? 1<CR>
```

As you can see in the above dialogue, the runtime services will build as many entries as it can with the information given in any one pass through the questions. In the first pass, two entries are built since two sub-devices and q-factors were specified. The Services assume that the CSR address is 160000 for both since it was specified only once. In the second pass, four entries were built. This is because four sub-devices were specified. The "-" construct tells the Runtime Services to increment the data from the first number to the second. In this case, sub-devices 2, 3, 4 and 5 were specified. (If the sub-device were specified by addresses, the increment would be by 2 since addresses must be on an even boundary.) The CSR addresses and Q-factors for the four entries are assumed to be 160000 and 0 respectively since they were only specified once. The last two units are specified in the third pass.

The whole process could have been accomplished in one pass as shown below.

```
# UNITS (D) ? 8<CR>
UNI) 1
CSR ADDRESS (0) ? 160000<CR>
SUB-DEVICE # (0) ? 0-7<CR>
Q-FACTOR (0) 0 ? 0,1,0,,,,1,1<CR>
```

As you can see from this example, null replies (commas enclosing a null field) tell the Runtime Services to repeat the last reply.

## 2.7 QUICK START-UP PROCEDURE

-----

To start-up this program:

1. Boot XXDP+
2. Give the date and answer the LSI and 50HZ (if there is a clock) questions
3. Type "R ZUDCEO"
4. Type "START"
5. Answer the "CHANGE HW" question with "Y"
6. Answer all the hardware questions
7. Answer the "CHANGE SW" question with "N"

When you follow this procedure you will be using only the defaults for flags and software parameters. These defaults are described in sections 2.3 and 2.5.

Sample of terminal dialogue to test two disks on one UDA-50:

DR>STA/FLA:PNT

CHANGE HW (L) ? Y

\* UNITS (D) ? 2

UNIT 0

UNIBUS ADDRESS OF UDA (0) 172150 ?

VECTOR (0) 154 ?

BR LEVEL (D) 5 ?

UNIBUS BURST RATE (D) ?

DRIVE NUMBER (D) 0.1

EXERCISE ON CUSTOMER DATA AREA IN TEST 4 (L) N ?

CHANGE SW (L) ? N

TST: 001

TESTING INTERRUPT ABILITY OF UDA AT ADR 172150 VEC 154...COMPLETED

TST: 002

TST: 003

TST: 004

UNIT 0 UDA AT 172150 DRIVE 0   RUNTIME 0:02:43  
INITIAL WRITE COMPLETE

UNIT 1 UDA AT 172150 DRIVE 1   RUNTIME 0:05:31  
INITIAL WRITE COMPLETE

TEST 4 IN PROGRESS.    RUNTIME 0:15:00

UNIT	DRIVE	SERIAL-NUMBER	SEEKS X1000	MBYTES READ	MBYTES WRITTEN	HARD ERRORS	SOFT ERRORS	ECC
0	0	0	3	9	6	0	0	0
1	1	1	3	8	6	0	0	0

02

Sample of terminal dialogue going through software questions to specify transfer limit (one disk being tested).

DR>STA/FLA:PNT

CHANGE HW (L) ? N

CHANGE SW (L) ? Y

ENTER MANUAL INTERVENTION MODE FOR SPECIAL DIAGNOSIS (L) N ?

REMAINING SOFTWARE QUESTIONS APPLY TO TEST 4 ONLY

ERROR LIMIT (D) 32 ?

READ TRANSFER LIMIT IN MEGABYTES - 0 FOR NO LIMIT (D) 0 ? 5

SUPPRESS PRINTING SOFT ERRORS (L) Y ?

DO INITIAL WRITE ON START (L) Y ?

ENABLE ERROR LOG (L) N ?

TST: 001

TESTING INTERRUPT ABILITY OF UDA AT ADR 172150 VEC 154...COMPLETED

TST: 002

TST: 003

TST: 004

UNIT 0 UDA AT 172150 DRIVE 0 RUNTIME 0:02:43  
INITIAL WRITE COMPLETE

UNIT 0 UDA AT 172150 DRIVE 0 RUNTIME 0:09:41  
REACHED TRANSFER LIMIT - TESTING STOPPED

TEST 4 IN PROGRESS. RUNTIME 0:09:41

UNIT	DRIVE	SERIAL-NUMBER	SEEKS X1000	MBYTES READ	MBYTES WRITTEN	HARD ERRORS	SOFT ERRORS	ECC
0	0	0	2	5	4	0	0	0

CZUDC EOP 1

0 CUMULATIVE ERRORS

TST: 001

TESTING INTERRUPT ABILITY OF UDA AT ADR 172150 VEC 154...COMPLETED

TST: 002

.  
:  
.

### 3.0 ERROR INFORMATION

#### 3.1 TYPES OF ERROR MESSAGES

There are three levels of error messages that may be issued by a diagnostic: general, basic and extended. General error messages are always printed unless the "IER" flag is set (section 2.3). The general error message is of the form:

```
NAME TYPE NUMBER ON UNIT NUMBER TST NUMBER PC:XXXXXX
error message
```

where: NAME = diagnostic name  
 TYPE = error type (SYS FTL ERR, DEV FTL ERR, HRD ERR or SFT ERR)  
 NUMBER = error number  
 UNIT NUMBER = 0 - N (N is last unit in PTABLE)  
 TST NUMBER = test and subtest where error occurred  
 PC:XXXXXX = address of error message call

System fatal errors (SYS FTL ERR) are used to report errors that are fatal to the entire diagnostic program. The diagnostic stops and the Runtime Services prompt is printed.

Device fatal errors (DVC FTL ERR) are used to report errors that are fatal to the device (may be either a UDA-50 or disk drive). Testing stops on that device for the remainder of the current test.

Hard errors (HRD ERR) reports most of the errors detected. Testing will normally continue after the printing of the error.

Soft errors (SFT ERR) are used only in test 4. They present information about an error for which recovery will be attempted. These are printed only if the SUPPRESS PRINTING SOFT ERRORS software question is answered "N" and are used only to provide a greater detail of information. During the error recovery attempt, several soft errors may be printed. Unless the soft errors are followed by a hard error message, the error condition was corrected and testing proceeds.

Basic error messages are messages that contain some additional information about the error. These are always printed unless the "IER" or "IBE" flags are set (section 2.3). These messages are printed after the associated general message.

Extended error messages contain supplementary error information such as register contents or good/bad data. These are always printed unless the "IER", "IBE" or "IXE" flags are set (section 2.3). These messages are printed after the associated general error message and any associated basic error messages.

The general and basic error messages from this diagnostic are always one line each. The basic message defines what program detected the error, the drive being tested and the time of the error.

The PDP-11 program that is loaded into memory when you give the "R ZUDCEO" command to the XXDP+ monitor is only a small part of this diagnostic. A data file called ZUDCEO.PAK on the system load device (the same device from which the "R" command read the PDP-11 program) contains four programs which are read from the file and loaded into the UDA-50 for execution. These programs are called "diagnostic machine" or DM programs. The "diagnostic machine" is the facility in the UDA-50 which executes a PDP-11 like program. The large majority of the testing is done by these four "diagnostic machine" programs. Once the PDP-11 program has loaded and started the "diagnostic machine" program, all it does is respond to requests from that program. These requests include such things as telling the "diagnostic machine" which disks on that UDA-50 are to be tested, printing an error message and updating statistics which are printed in the statistical report (see section 4.0).

The basic message (the second line of every error message) will be one of the following:

HOST PROGRAM UDA AT xxxxxx RUNTIME hhh:mm:ss

The host program (PDP-11) detected the error. UDA AT xxxxxx identifies the address of the UDA-50 being tested. It may be omitted if the error is not specific to one UDA-50.

UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss

The "diagnostic machine" program loaded in test 1 detected the error. DM PC xxxx identifies the address in the "diagnostic machine" program where the error message is reported.

DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss

The "diagnostic machine" program loaded in test 2 detected the error. DM PC xxxx identifies the address in the "diagnostic machine" program where the error message is reported. DRIVE xxx identifies the drive number.

DISK FUNCTIONAL DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss

The "diagnostic machine" program loaded in test 3 detected the error.

DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss

The "diagnostic machine" program loaded in test 4 detected the error.

Sample error message:

```

CZUDC DVC FTL ERR 00021 ON UNIT 00 TST 001 SUB 003 PC: xxxxxx      - general message
HOST PROGRAM UDA AT 172150 RUNTIME 0:00:12                        - basic message
UDA RESIDEN. DIAGNOSTICS DETECTED FAILURE                          \
  UDASA CONTAINS 104041                                           }- extended message
REPLACE UDA MODULE M7485                                          /

```

Informational messages are also printed by this program. They are usually one or two lines in length. They are printed as extended messages and are always printed unless the "IER", "IBE" or "IXE" flags are set.

Sample informational message:

```
UNIT 0 UDA AT 172150 DRIVE 0 RUNTIME 0:02:43
INITIAL WRITE COMPLETE
```

### 3.2 SPECIFIC ERROR MESSAGES

-----

Following is a list of the error messages that may be printed by the diagnostic program. In the list, some of the numbers that may vary with execution or program version are shown as "xxx". These include program counters and runtime. Other numbers, such as unit number, drive number, UDA-50 address and data in registers are filled with sample numbers. Additional information about the error may follow the error message.

#### 3.2.1 HOST PROGRAM ERROR MESSAGES (00001 to 00999)

-----

```
00001 CZUDC SYS FTL ERR 00001 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx
I DON'T LIKE THE ANSWERS YOU GAVE TO THE HARDWARE QUESTIONS
UDA HAS MORE THAN ONE VECTOR, BR LEVEL OR BURST RATE
```

When the hardware questions were answered, two units were selected with the same UNIBUS address but with a different vector, BR level or burst rate. A single UDA-50 can have only one vector, BR level or burst rate. The program is aborted and returns to the Runtime Services prompt so that you can change the hardware questions.

```
00002 CZUDC SYS FTL ERR 00002 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx
I DON'T LIKE THE ANSWERS YOU GAVE TO THE HARDWARE QUESTIONS
TWO UNITS SELECT THE SAME DRIVE
```

The hardware questions for two units were exactly the same. The program is aborted and returns to the Runtime Services prompt so that you can change the hardware questions.

00003 CZUDC SYS FTL ERR 00003 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
I DON'T LIKE THE ANSWERS YOU GAVE TO THE HARDWARE QUESTIONS;  
MORE THAN EIGHT DRIVES SELECTED ON THIS UDA.

Up to four physical disk drives can be attached to a UDA-50 at one time. A physical disk drive may be from one to four logical disk drives. Each logical disk drive is considered one unit to the diagnostic program. Even though more than eight logical disk drives can be attached to one UDA-50, the UDA-50 only supports eight. The program is aborted and returns to the Runtime Services prompt so that you can change the hardware questions.

00004 CZUDC SYS FTL ERR 00004 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM RUNTIME x:xx:xx  
NOT ENOUGH ROOM IN MEMORY TO TEST THE UNITS SELECTED  
PLEASE START PROGRAM OVER AND TEST FEWER UNITS AT A TIME

This program does not limit the number of units that can be tested by specifying a maximum number. What limits the number is the amount of memory used to store data on each unit. You have exceeded the number of units that are testable at one time. Start program over and select fewer units.

00005 CZUDC SYS FTL ERR 00005 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM RUNTIME x:xx:xx  
CHECKSUM ERROR IN DM PROGRAM FILE

As a DM program is read from the load media, a checksum is calculated. If the checksum contained in the file does not match what is calculated, an error reading the data file is declared. Restore the data file ZUDCEO.PAK to your load media.

00006 CZUDC SYS FTL ERR 00006 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM RUNTIME x:xx:xx  
TABLE INCONSISTANCY ERROR. PLEASE RE-LOAD PROGRAM

When the host program is started, controller tables are set according to the P-tables. Error 00006 will occur if the tables were corrupted after restarting the diagnostic. Load and start your program again.

00007 CZUDC SYS FTL ERR 00007 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM RUNTIME x:xx:xx  
ERROR IN DM PROGRAM FILE. DM PROGRAM NOT FOUND

The host program was not able to read the DM program from the load media properly. Restore the data file ZUDCEO.PAK to your load media.

00008 CZUDC SYS FTL ERR 00008 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
I DON'T LIKE THE ANSWERS YOU GAVE TO THE HARDWARE QUESTIONS  
TWO UDA'S USE THE SAME VECTOR

The hardware questions for two units specified different UDA-50 Unibus addresses but identical vector addresses. The program is aborted and returns to the Runtime Services prompt so that you can change the hardware questions.

00010 CZUDC DVC FTL ERR 00010 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
WRONG APT DIAGNOSTIC IS BEING USED WITH THIS CONTROLLER  
USE CIUDx

The APT diagnostics are designed to run with one type of UDA-50 board set (either M7161-2 or M7485-6). For example, if the user is running CIUDA with a UDA-50 M7485-6 type, this error will occur. In that case the user will be told to use CIUDF. The following is a detailed description of which test is used with what configuration.

CIUDF - UDA-50 with M7485-6 modules runs tests 1-3  
CIUDG - UDA-50 with M7485-6 modules runs test 4  
CIUDH - UDA-50 with M7485-6 modules runs tests 1-3  
CIUDI - UDA-50 with M7485-6 modules runs test 4

00014 CZUDC DVC FTL ERR 00014 ON UNIT 00 TST xxx SUB xxx PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
UDA50 CONTROLLER IS AT A REVISION LEVEL NO LONGER SUPPORTED  
BY THIS DIAGNOSTIC PROGRAM. THIS PROGRAM REQUIRES A UDA50-A  
CONTROLLER (MODEL 6) WITH MICROCODE REVISION AT 3 OR GREATER.  
CONTROLLER REPORTED MODEL CODE xx AND MICROCODE VERSION xx

All UDA50-0's (modules M7161-2) are not supported by this diagnostic. The module set M7485-6 is the only one that can be tested by this diagnostic. If the controller is a UDA50-0 (M7161-2) it will not be tested. If the controller is a UDA50-A (M7485-6) and it has old microcode (the microcode version is less than 3) this message will be printed but testing will go on. If the controller consists of the M7161-2 modules, install one with M7485-6 modules. Do not intermix the two, it will not work!

00021 CZUDC DVC FTL ERR 00021 ON UNIT 00 TST 001 SUB 003 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDA RESIDENT DIAGNOSTICS DETECTED FAILURE  
 UDASA CONTAINS 105154  
 REPLACE UDA MODULE M7486

The UDA Resident diagnostic detected a failure. The error is displayed in the UDASA. Here are the possible error values and their meaning:

104000 - Fatal sequencer error  
 104040 - D processor ALU error  
 104041 - D proc ROM parity error  
 105102 - D proc with no Board #2 or RAM parity error  
 105105 - D proc RAM buffer error  
 105152 - D proc SDI error  
 105153 - D proc write mode wrap SERDES error  
 105154 - D proc read mode SERDES, RSGEN, and ECC error  
 106040 - U proc ALU error  
 106041 - U proc Control Register error  
 106042 - U proc DFAIL/ROM parity error/Board #1 test count is wrong  
 106047 - U proc Constant ROM error with D proc running SDI test  
 106055 - Unexpected trap found, aborted diagnostic  
 106071 - U proc ROM error  
 106072 - U proc ROM parity error  
 106200 - Step 1 data error (MSB not set)  
 107103 - U proc RAM parity error  
 107107 - U proc RAM buffer error  
 107115 - Board #2 test count was wrong  
 112300 - Step 2 error  
 122240 - NPR error  
 122300 - Step 3 error  
 142300 - Step 4 error

Replace the board specified. M7485 is the Unibus interface board. M7486 is the SDI interface board.

00022 CZUDC DVC FTL ERR 00022 ON UNIT 00 TST 001 SUB 003 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 STEP BIT DID NOT SET IN UDASA REGISTER DURING INITIALIZATION  
 STEP BIT EXPECTED 004000  
 UDASA CONTAINS 000000  
 REPLACE UDA MODULE M7485

The UDA did not respond as expected during the initialization sequence which communicates using data in the UDASA register. A normal response from the UDA contains either a STEP bit or an ERROR bit defined as follows:

Bit 15 (100000)	Error bit
Bit 14 (040000)	Step 4 bit
Bit 13 (020000)	Step 3 bit
Bit 12 (010000)	Step 2 bit
bit 11 (004000)	Step 1 bit

The expected step bit nor the error bit set within the expected time.

00023 CZUDC DVC FTL ERR 00023 ON UNIT 00 TST 001 SUB 005 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDA DID NOT CLEAR RING STRUCTURE IN HOST MEMORY DURING INITIALIZATION  
 6 WORDS WERE TO BE CLEARED STARTING AT ADDRESS 040644  
 FIRST SEVERAL WORDS NOT CLEARED (UP TO 6):

ADDRESS	CONTENTS
040644	000010
040650	000010
040652	000010

REPLACE UDA MODULE M7485

The UDA is to clear the ring structure (a communications area used by the UDA to talk to the host) in host memory before Step 4 of initialization. If the UDA diagnostics did not clear memory and did not flag an error, then error message 00023 is displayed. The contents of each word in memory is set to 177777 before the test. Failure of the UDA to clear each word indicates a fault in the address interface to the Unibus.

00024 CZUDC DVC FTL ERR 00024 ON UNIT 00 TST 001 SUB 006 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDASA REGISTER DID NOT GO TO ZERO AFTER STEP 3 WRITE OF INITIALIZATION  
 PURGE/POLE DIAGNOSTICS WERE REQUESTED  
 UDASA CONTENTS 004400

For better testing, the host can test the PURGE and POLE mechanism of the UDA. To do so the host sets bit15 of the step 3 data and sends the data to the UDA. The UDA must go to zero and wait for the purge and pole. If the UDA never went to zero, then error message 00024 is displayed. The UDA may have a bad M7485 module or the UNIBUS maybe broken.

00025 CZUDC DVC FTL ERR 00025 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDA DID NOT RETURN CORRECT DATA IN UDASA REGISTER DURING INITIALIZATION  
 UDASA EXPECTED 004400  
 UDASA CONTAINS 004000  
 REPLACE UDA MODULE M7485

For each step of initialization, specific data is expected to be displayed in the UDASA. If the UDASA does not match the expected data, then error message 00025 is displayed. Replace UDA module M7485.

00026 CZUDC DVC FTL ERR 00026 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 DATA COMPARISON ERROR DURING DIAGNOSTIC PORT LOOP TEST  
 DATA SENT TO UDASA 000001  
 RECEIVED FROM UDASA 000000  
 REPLACE UDA MODULE M7485

The UDA can be put into a mode where the UDASA acts as a wrap port. While the UDA is in this mode, any data being sent to the UDASA will be displayed in the UDASA within a small period of time. If the data in the UDASA does not match the data that was sent to the UDASA, then error message 00026 is displayed. Replace UDA module M7485.

00027 CZUDC DVC FTL ERR 00027 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDASA REGISTER DID NOT CHANGE AFTER WRITING TO IT  
 IN PORT LOOP DIAGNOSTIC  
 UDASA CONTAINS 004400  
 REPLACE UDA MODULE M7485

The UDA can be put into a mode where the UDASA acts as a wrap port. While the UDA is in this mode, any data being sent to the UDASA will be displayed in the UDASA within a small period of time. After the host program sent data to it while it was in diagnostic wrap mode, the UDA did not change the contents of the UDASA. Error message 00027 is displayed. Replace UDA module M7485.

00028 CZUDC DVC FTL ERR 00028 ON UNIT 00 TST 001 SUB 004 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDA DID NOT INTERRUPT THE PDP-11  
 REPLACE UDA MODULE M7485

The host program timed out while waiting for an interrupt that had to occur. The UDA was told to use interrupts during the initialization process. The UDA then waited for the interrupt but it did not occur. Replace the UDA module M7485.

00029 CZUDC DVC FTL ERR 00029 ON UNIT 00 TST 001 SUB 004 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDA INTERRUPTED AT DIFFERENT BR LEVEL THAN SPECIFIED IN HARDWARE  
 QUESTIONS. INTERRUPT WAS AT BR LEVEL 5  
 CHECK PRIORITY PLUG ON UDA MODULE M7485  
 OR CHANGE HARDWARE QUESTIONS

The priority plug on the UDA and the BR LEVEL specified during the hardware questions do not match. Either change the plug number or reanswer the hardware question. If all these have been done and there is still a problem replace UDA module M7485.

00030 CZUDC DVC FTL ERR 00030 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
 HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
 UDA REPORTED FATAL ERROR IN UDASA REGISTER WHILE RUNNING DM PROGRAM  
 UDASA CONTAINS 100004

A message from the UDA firmware reports an unexpected failure. An error code is presented in the UDASA. Here is a list of the codes and their meanings:

- 004400 - UDA has been init'd by either a bus init or by writing into the UDATP.
- 100001 - UNIBUS envelope/pack read error (parity or timeout)
- 100002 - UNIBUS envelope/packet write error (parity or timeout)
- 100003 - UDA ROM and RAM parity error
- 100004 - UDA RAM parity error
- 100005 - UDA ROM parity error
- 100006 - UNIBUS ring read error
- 100007 - UNIBUS ring write error
- 100010 - UNIBUS interrupt master failure
- 100011 - Host access timeout error
- 100012 - Host exceeded credit limit
- 100013 - UDA SDI hardware fatal error
- 100014 - DM XFC fatal error
- 100015 - Hardware timeout of instruction loop
- 100016 - Invalid virtual circuit identifier
- 100017 - Interrupt write error on UNIBUS

00031 CZUDC DVC FTL ERR 00031 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
NO INTERRUPT RECEIVED FROM DM PROGRAM FOR 3 MINUTES  
ASSUME PROGRAM IS HUNG

All DM programs are required to communicate with the host program; so as to assure the host program that the DM program is not hung up or in an endless loop. If the DM program has not done so, the host program assumes the DM is hung and this message appears.

00032 CZUDC DVC FTL ERR 00032 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
MESSAGE BUFFER RECEIVED FROM DM PROGRAM WITH UNKNOWN REQUEST NUMBER  
MESSAGE BUFFER CONTAINS:

000001 000002 000003 000004 000005 000006 000007  
000008 000009 000010 000011 000012 000013 000014  
000015 000016 000017 000018 000019 000020 000021  
000022 000023 000024 000025 000026 000027 000028  
000029 000030 000031 000032 000033 000034 000035

The DM program and the host program communicate with each other using packets. Each packet must have a request number set up by the DM program and interpreted by the host program. This request number is not a known request number. The problem may be the UNIBUS or either one of the UDA modules or a corrupted DM program. Word 1 contains the DM request number, and word 2 typically contains the drive number. The rest of the buffer contains information specific to a DM request. The numbers in the example show the order in which words are displayed.

```

00033 CZUDC DVC FTL ERR 00033 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx
RESPONSE PACKET FROM UDA DOES NOT CONTAIN EXPECTED DATA
EITHER UDA RETURNED ERROR STATUS OR PACKET WAS NOT RECEIVED CORRECTLY
COMMAND PACKET SENT      RESPONSE PACKET RECEIVED
000000 000020          000000 000020
000000 000000          000000 000000
000000 000002          000000 000202
000000 014336          000000 014336
000000 034674          000000 034674
000000 00000C          000000 000000
000000 000000          000000 000000
000000 051232          000000 051232
000000 000000          000000 000000
000000 000000          000000 000000
000000 000000          000000 000000
000000 000000          000000 000000

```

The host program inspected the response packet which was given by to UDA. The response packet may have been in error with one of the following points:

- 1) The end code was not as expected.
- 2) The status code showed an error occurred with the last command.
- 3) The command reference numbers (the first word) did not match.

If 1 or 3 occurred, there may have been a transmission problem between the UDA and the host program. If 2 occurred, check the error code in the MSCP specification for further information. The packets are displayed two words per line, low order word and byte to the right (corresponding to the MSCP long-word entity).

```

00036 CZUDC DVC FTL ERR 00036 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx
NO INTERRUPT RECEIVED FROM UDA FOR 30 SECONDS
WHILE LOADING DM PROGRAM

```

After a DM program has been sent to the UDA, the host program expects an interrupt within 30 seconds. The interrupt is used to assure the host program that the DM program is gone. If no interrupt occurred, then error message 00036 is displayed and the DM program is assumed to be hung.

00037 CZUDC DVC FTL ERR 00037 ON UNIT 00 TST xxx SUB 000 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
UDA REPORTED FATAL ERROR IN UDASA REGISTER WHILE LOADING DM PROGRAM  
UDASA CONTAINS 100004  
REPLACE UDA MODULE M7485

While loading the DM program to the UDA, the UDASA became non-zero. When this occurs, it signifies that the UDA microcode has run across a fatal error. The displayed value is in octal. Check the error code with the list included with error number 00030.

00038 CZUDC DVC FTL ERR 00038 ON UNIT 00 TST 001 SUB 002 PC: xxxxxx  
HOST PROGRAM UDA AT 172150 RUNTIME x:xx:xx  
MEMORY ERROR TRYING TO READ UDA REGISTERS  
CHECK UNIBUS SELECTION SWITCHES ON UDA MODULE M7486  
OR UNIBUS  
OR REPLACE UDA MODULE M7485

A non-existent memory error occurred when the host program tried to access the UDAIP and UDASA registers while in subtest 2 of test 1. The UDA is at another address (check the UNIBUS selection switches) or module M7485 is broken or the UNIBUS is broken.

3.2.2 TEST 1 ERROR MESSAGES (01000 TO 01999)  
-----

01000 CZUDC HRD ERR 01000 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx  
UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss  
NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS.

	OCTAL	HEX
ADDRESS	000000	000C0

The host has given the DM routine the range of accessible host memory. While reading one location within the range, it appeared non-existent to the UDA. Since everything within the bounds were believed to be accessible this error message will be printed. The message prints the address in octal and hex.

01001 CZUDC HRD ERR 01001 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx  
UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss  
PARITY ERROR ON READ FROM UNIBUS.

	OCTAL	HEX
ADDRESS	000000	00000
DATA READ	000000	0000
DATA EXPECTED	000000	0000

The host has given the DM routine the range of accessible host memory. While reading one location within the range, the DM routine has found a location with bad parity. Every location was accessed by the host program. The host program filled a location with its address. The message prints the address, the data it actually received, and the expected data it should have received in octal and hex.

01002 CZUDC HRD ERR 01002 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx  
UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss  
UNIBUS ADDRESSING ERROR - INCORRECT DATA READ.  
MEMORY LOCATION SHOULD CONTAIN OWN ADDRESS.

	OCTAL	HEX
DATA READ	000000	0000
DATA EXPECTED	000000	0000

The host has given the DM routine the locations of accessible host memory. Every location was accessed by the host program. The host program filled a location with its address. The DM program read from one location and found that the data it read was not equal to its address. The message prints the address, the data it actually received, and the expected data it should have received in octal and hex.

E3

01003 CZUDC HRD ERR 01003 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx  
UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss  
NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS WITHIN BUFFER.  
                                  OCTAL      HEX  
STARTING ADDRESS OF BUFFER 123456      0A72E  
          BUFFER SIZE          001234      029C

After reading every accessible location of host memory, the DM routine breaks up memory into buffers. The DM routine writes and reads data patterns from each host buffer into its DM buffer. While reading one of these buffers, a non-existent memory error occurred. The message prints out the starting address of the buffer and the size of the buffer in octal(for PDP-11 users) and in hex(for VAX users) so the user can determine about where the non-existent memory location occurred.

01004 CZUDC HRD ERR 01004 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx  
UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss  
PARITY ERROR ON READ FROM UNIBUS WITHIN BUFFER.  
                                  OCTAL      HEX  
STARTING ADDRESS OF BUFFER 123456      0A72E  
          BUFFER SIZE          001234      029C

After reading every accessible location of host memory, the DM routine breaks up memory into buffers. The DM routine writes and reads data patterns from each host buffer into its DM buffer. While reading one of these buffers, a parity error occurred. The message prints out the starting address of the buffer and the size of the buffer in octal(for PDP-11 users) and in hex(for VAX users) so the user can determine about where the non-existent memory location occurred.

```

01005 CZUDC HRD ERR 01005 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx
UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss
DATA COMPARE FAILED AFTER WRITE THEN READ FROM UNIBUS.
BUFFER SIZE = 005302(O) 0AC2(X) 2754.(D)
STARTING ADDRESSES OF BUFFERS
OCTAL HEX
044232 0489A
057056 05E2E
071676 073BE
104512 0894A
CURRENT DATA PATTERN READ 0
LAST PATTERN WRITTEN 0
STARTING ADDRESS OF LAST BUFFER WRITTEN 104512(O) 0894A(X)
NUMBER OF ERRORS FOUND 2754.(D)
LOCATION DATA EXPECTED DATA RECEIVED
OCTAL HEX OCTAL HEX OCTAL HEX
057056 05E2E 111111 9249 002472 053A
057060 05E30 044444 4924 005302 0AC2
057062 05E32 022222 2492 000000 0000

```

After reading an entire buffer, the DM program checks each location. If any or all of the locations did not contain the expected data, this message appears. It contains the buffer size in octal, hex and decimal. The reason it appears in decimal is so the user can correlate this value with the number of errors which is printed in decimal. The starting addresses of the buffers are printed in octal and hex. There will always be at least two buffers and up to four buffers printed. The current data pattern read is printed. DM program will be testing the buffer with this data pattern. The last data pattern written by the DM program is printed. The address of the last buffer written is printed in octal and hex. As many as three errors are presented in the message. This portion presents the location of the error, the expected data and the actual data all in octal and hex.

01006 CZUDC HRD ERR 01006 ON UNIT 00 TST 001 SUB 007 PC: xxxxxx  
 UNIBUS ADDRESSING DM PC:xxxx UDA AT xxxxxx RUNTIME hhh:mm:ss  
 UNIBUS ADDRESSING ERROR, TWO ADDRESSES READ SAME LOCATION.

	OCTAL	HEX
KNOWN GOOD ADDRESS	625252	32AAA
ERROR ADDRESS	425252	22AAA
ADDRESS BIT IN ERROR	200000	10000

The UDA can only write to a small portion of memory because there is a PDP-11 program running in the memory. To verify it can address all of memory, it uses one location that it is permitted to write which it calls a "known good address". By changing only one bit in the address of this location it selects a "test address". Different patterns are written to the "known good address", each followed by a read of the "test address". If the data read from the "test address" matches the data written to the "known good address" each time, the address line is determined to be stuck. The "test address" is printed as the error address.

### 3.2.3 TEST 2 INFORMATIONAL MESSAGES

-----

UNIT x UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 INFORMATION SENT BACK FROM THE DRIVE IS BEING PRESENTED.  
 TEST NUMBER 0000  
 DRIVE TYPE 00  
 ERROR NUMBER 0000  
 data

There is not error, but it is a message. The disk drive wanted the let the host know what had happened when the drive's internal diagnostic was run. The format follows that of hard error 2021.

UNIT x UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 FOLLOWING REPORT HAS BEEN TRUNCATED DUE TO SIZE

This is a message that may appear if the disk drive gave too much data for the DM program to handle. This message may precede the previous message and hard error 2021.

3.2.4 TEST 2 ERROR MESSAGES (02000 TO 02999)  
-----

02000 CZUDC HRD ERR 02000 ON UNIT 00 TST 002 SUB 000 PC: xxxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
HOST SPECIFIED UNIT #0 THAT CAN'T BE FOUND.  
TEST2 RESTARTING

When test 2 starts executing out of the DM, it doesn't know if it had been started to execute drive diagnostics or restarted to down line load a diagnostic into the drive. If it had been restarted for the latter reason, the host must tell Test 2 which drive was to receive the diagnostic. If the drive specified by the host is not attached to the UDA or could not be located by Test 2, this error message will be printed.

02001 CZUDC HRD ERR 02001 ON UNIT 00 TST 002 SUB 000 PC: xxxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
CANNOT RECEIVE VALID DRIVE STATE FROM DRIVE AFTER DRIVE WAS INITED  
CHECK IF DRIVE IS POWERED ON.

This error message is presented if valid drive state was not received from the drive after the drive was inited. There are two types of invalid states: no clocks or 'hard' errors. If after getting state and no clocks occur, error 2001 is reported. There may be a bad transmitter on the drive side or a bad receiver on the UDA side or the SDI cable may have taken a hit.

02002 CZUDC HRD ERR 02002 ON UNIT 00 TST 002 SUB 000 PC: xxxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DRIVE STATE RECEIVED HAS BAD PARITY AFTER DRIVE WAS INITED

This error message is presented if bad parity was received from the drive after the drive was inited. There may be a bad transmitter on the drive side or a bad receiver on the UDA side or the SDI cable may have taken a hit.

02003 CZUDC HRD ERR 02003 ON UNIT 00 TST 002 SUB 000 PC: xxxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DRIVE IS NOT ASSERTING RECEIVER READY IN DRIVE STATE AFTER DRIVE WAS INITED

This error message is presented if receiver ready was not received from the drive after the drive was inited. There may be a bad transmitter on the drive side or a bad receiver on the UDA side or the SDI cable may have taken a hit.

02004 CZUDC HRD ERR 02004 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF ECHO COMMAND TO DRIVE  
ECHO DATA FF

This error message is presented if a send of the ECHO command timed out. This may be caused by receiver ready being deasserted. The echo data is presented in hex.

02005 CZUDC HRD ERR 02005 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF ECHO RESPONSE FROM DRIVE  
ECHO DATA FF

This error message is presented if a receive of an ECHO command was in error. The echo data is presented in hex. There may be a bad transmitter on the drive side or a bad receiver on the UDA side or the SDI cable may have taken a hit.

02006 CZUDC HRD ERR 02006 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ECHO COMMAND RESPONDED WITH DIFFERENT DATA  
ECHO DATA SENT 00FE  
ECHO DATA RECEIVED 00FF

This error message is presented if the data returning from an ECHO command did not match the data it was suppose to. The data presented is in hex.

02007 CZUDC HRD ERR 02007 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR BIT SET IN GET STATUS RESPONSE AFTER DRIVE CLEAR COMMAND  
GET STATUS RESPONSE  
REAL TIME STATE state  
STATUS (FROM R TO L): word6 word5 word4 word3 word2 word1 word0:

This error message is presented when an error bit is set in the status of a drive after the drive was cleared of all errors. The data displayed is the response from a GET STATUS command. The error bits in the response are in bit position 3, 5 and 6 of word2. For further description of the GET STATUS response, refer to the SDI Functional Spec v3.6 and the drive's functional spec.

REAL TIME STATE state: REAL TIME STATE 0003

The real time state is the real time drive state <<AFTER>> Test 2 detected the error. <<THIS VALUE IS DISPLAYED IN HEX>>. In this example, receiver ready and attention are both asserted.

The bit positions are defined as follows:

- 0001 - Receiver ready (Test 2 able to transmit to drive)
- 0002 - Attention (error occurred or online timeout expired)
- 0040 - Available (drive offline and usable)
- 1000 - Read/Write ready

The complete meaning of these bits is beyond the scope of this text, please refer to the operator documentation for the drive you are working on.

STATUS (R TO L): word6 word5 word4 word3 word2 word1 word0:  
The status is the response to the SDI GET STATUS command. These words are printed in HEX. <<NOTE THAT THE STATUS IS PRINTED OUT FROM RIGHT TO LEFT!!>>. The status' meaning is beyond the scope of this text, please refer to the operator documentation for the drive you are working on.

02008 CZUDC HRD ERR 02008 ON UNIT 00 1ST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF ONLINE COMMAND TO DRIVE

The ONLINE command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02009 CZUDC HRD ERR 02009 ON UNIT 00 1ST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF ONLINE RESPONSE FROM DRIVE  
explanation

This error message is presented if a receive of an ONLINE command was in error. An explanation of what the error was is also presented. These explanations are:

TIMEOUT ERROR OCCURED DURING RECEIVE XFC

- This error is a failure of the drive to respond to an SDI level 2 command (see the SDI specification) before the drive-supplied command timeout expires.

1ST WORD NOT START FRAME DURING RECEIVE XFC

- The first word received by the UDA from the drive was not a valid message start frame.

FRAMING ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC

- This is caused by one of the following conditions:  
1) Illegal frame code -- the frame is not a message start, continue, or end frame. 2) Illegal sequence of frames -- such as a message start frame without ever receiving a message end frame. This can be caused by the drive sending a response before the UDA asserts receiver ready, or a random hit on the SDI cable that garbles a frame or a bad drive transmitter or UDA receiver.

CHECKSUM ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC

- The checksum attached to a message end frame did not match the checksum computed over the level 0 command. This could be caused by a bad drive transmitter, bad UDA receiver, incorrectly computed checksum by the drive (unlikely) or a random hit on the SDI cable.

BUFFER SIZE SMALLER THEN RESPONSE DURING RECEIVE XFC

- A buffer size size set aside for the response was not large enough for the response received. This is caused by the drive sending a response that is incorrect for the request sent to the drive, or the drive sending some garbage with the response.

CODE FROM RECEIVE XFC WAS UNINTELLIGIBLE FROM SUBSYSTEM 0000

- The response from the drive was not anything that was expected. Possible UDA microcode change without test 2 update.

02010 CZUDC HRD ERR 02010 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ONLINE COMMAND WAS UNSUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The ONLINE command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status. The drive did not assert the RECEIVER READY signal over the SDI.

02011 CZUDC HRD ERR 02011 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ONLINE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
EXPECTED RESPONSE 7E  
ACTUAL RESPONSE 00

The ONLINE command did not return an expected response code. If there were at least an UNSUCCESSFUL response, test 2 will report the drive state and status. The expected response and actual response are in hex.

02012 CZUDC HRD ERR 02012 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF GET UNIT CHARACTERISTICS COMMAND TO DRIVE

The GET UNIT CHARACTERISTICS command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02013 CZUDC HRD ERR 02013 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF GET UNIT CHARACTERISTICS COMMAND FROM DRIVE  
explanation

This error message is presented if a receive of a GET UNIT CHARACTERISTICS command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02014 CZUDC HRD ERR 02014 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
GET UNIT CHARACTERISTICS COMMAND WAS UNSUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The GET UNIT CHARACTERISTICS command was not successful.  
The drive's status is displayed. See hard error 2007 for  
further information on the format of the status.

02015 CZUDC HRD ERR 02015 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
GET UNIT CHARACTERISTICS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
EXPECTED RESPONSE 78  
ACTUAL RESPONSE 00

The GET UNIT CHARACTERISTICS command did not return an expected  
response code. The expected response and actual response  
are in hex.

02016 CZUDC HRD ERR 02016 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
HOST PROGRAM GAVE DM CODE IMPROPER DATA  
EXPECTED VALUE SHOULD BE BETWEEN 0 AND 3  
ACTUAL VALUE WAS xx

The host tells the DM program what to do after the DM  
program is done testing the drive's diagnostic. If  
the value is not within the expected range, this error  
message is printed. There is no drive problem. The  
problem is between the host and the UDA.

02017 CZUDC HRD ERR 02017 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF DIAGNOSE COMMAND TO DRIVE

The DIAGNOSE command timed out while it was sent  
to the drive. The drive did not assert  
the RECEIVER READY signal over the SDI.

02018 CZUDC HRD ERR 02018 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF DIAGNOSE RESPONSE FROM DRIVE  
explanation

This error message is presented if a receive of a DIAGNOSE  
command was in error. An explanation of what the error was  
is also presented. These explanations are described in  
hard error 2009.

02019 CZUDC HRD ERR 02019 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DIAGNOSE COMMAND WAS UNSUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The DIAGNOSE command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02020 CZUDC HRD ERR 02020 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DIAGNOSE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
EXPECTED RESPONSE FC  
ACTUAL RESPONSE 00

The DIAGNOSE command did not return an expected response code. The expected response and actual response are in hex.

02021 CZUDC HRD ERR 02021 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DRIVE DIAGNOSTIC REPORTS A HARD ERROR  
TEST NUMBER 0000  
DRIVE TYPE 00  
ERROR NUMBER 0000  
data

The drive diagnostic found an error and is reporting the error back to the host. All values are in hex. TEST NUMBER shows what test was run. DRIVE TYPE shows what type of drive was being tested. ERROR NUMBER shows the result of the test. The drive may pass back data to the host. This data will be presented in a 32 bit hex format following the error message. More data may follow the 32 bit hex values. This data is printed in ascii format. For definitions of what these values mean, refer to the drive functional spec.

02022 CZUDC HRD ERR 02022 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
HOST PROGRAM DOWN LINE LOADED A DIAGNOSTIC WITH A ZERO BYTE COUNT

The host program was attempting to down line load a diagnostic of zero length. The DM program must have the byte count specified by the host.

02023 CZUDC HRD ERR 02023 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DIAGNOSTIC filnam REQUESTED BY THE DRIVE COULD NOT BE SUPPLIED BY HOST.

The host program could not supply the diagnostic 'filnam' to down line load to the drive.

02024 CZUDC HRD ERR 02024 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF MEMORY READ COMMAND TO DRIVE

The MEMORY READ command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02025 CZUDC HRD ERR 02025 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF MEMORY READ RESPONSE FROM DRIVE  
explanation

This error message is presented if a receive of a MEMORY READ command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02026 CZUDC HRD ERR 02026 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
MEMORY READ COMMAND WAS UNSUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The MEMORY READ command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02027 CZUDC HRD ERR 02027 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
MEMORY READ COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
EXPECTED RESPONSE 72  
ACTUAL RESPONSE 00

The MEMORY READ command did not return an expected response code. The expected response and actual response are in hex.

02028 CZUDC HRD ERR 02028 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF MEMORY WRITE COMMAND TO DRIVE

The MEMORY WRITE command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02029 CZUDC HRD ERR 02029 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF MEMORY WRITE RESPONSE FROM DRIVE  
explanation

This error message is presented if a receive of a MEMORY WRITE command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02030 CZUDC HRD ERR 02030 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
MEMORY WRITE COMMAND WAS UNSUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The MEMORY WRITE command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02031 CZUDC HRD ERR 02031 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
MEMORY WRITE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
EXPECTED RESPONSE 7E  
ACTUAL RESPONSE 00

The MEMORY WRITE command did not return an expected response code. The expected response and actual response are in hex.

02032 CZUDC HRD ERR 02032 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND OF RUN COMMAND TO DRIVE

The RUN command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02033 CZUDC HRD ERR 02033 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
ERROR DURING RECEIVE OF RUN RESPONSE FROM DRIVE  
explanation

This error message is presented if a receive of a RUN command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02034 CZUDC HRD ERR 02034 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
RUN COMMAND WAS UNSUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The RUN command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02035 CZUDC HRD ERR 02035 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 RUN COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
 EXPECTED RESPONSE 7E  
 ACTUAL RESPONSE 00

The RUN command did not return an expected response code. The expected response and actual response are in hex.

02036 CZUDC HRD ERR 02036 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 TIME-OUT ON SEND OF RECALIBRATE COMMAND TO DRIVE

The RECALIBRATE command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02037 CZUDC HRD ERR 02037 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 ERROR DURING RECEIVE OF RECALIBRATE RESPONSE FROM DRIVE  
 explanation:

This error message is presented if a receive of a RECALIBRATE command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02038 CZUDC HRD ERR 02038 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 RECALIBRATE COMMAND WAS UNSUCCESSFUL  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The RECALIBRATE command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02039 CZUDC HRD ERR 02039 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 RECALIBRATE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
 EXPECTED RESPONSE 7E  
 ACTUAL RESPONSE 00

The RECALIBRATE command did not return an expected response code. The expected response and actual response are in hex.

02040 CZUDC HRD ERR 02040 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 TIME-OUT ON SEND OF GET STATUS COMMAND TO DRIVE

The GET STATUS command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02041 CZUDC HRD ERR 02041 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 ERROR DURING RECEIVE OF GET STATUS RESPONSE FROM DRIVE  
 explanation

This error message is presented if a receive of a GET STATUS command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02042 CZUDC HRD ERR 02042 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 GET STATUS COMMAND WAS UNSUCCESSFUL  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0900 0706 0504 0302 0100

The GET STATUS command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02043 CZUDC HRD ERR 02043 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 GET STATUS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
 EXPECTED RESPONSE FE  
 ACTUAL RESPONSE 00

The GET STATUS command did not return an expected response code. The expected response and actual response are in hex.

02044 CZUDC HRD ERR 02044 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 TIME-OUT ON SEND OF DRIVE CLEAR COMMAND TO DRIVE

The DRIVE CLEAR command timed out while it was sent to the drive. The drive did not assert the RECEIVER READY signal over the SDI.

02045 CZUDC HRD ERR 02045 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 ERROR DURING RECEIVE OF DRIVE CLEAR RESPONSE FROM DRIVE  
 explanation

This error message is presented if a receive of a DRIVE CLEAR command was in error. An explanation of what the error was is also presented. These explanations are described in hard error 2009.

02046 CZUDC HRD ERR 02046 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 DRIVE CLEAR COMMAND WAS UNSUCCESSFUL  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The DRIVE CLEAR command was not successful. The drive's status is displayed. See hard error 2007 for further information on the format of the status.

02047 CZUDC HRD ERR 02047 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
 DISK RESIDENT DM PC:xxxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 DRIVE CLEAR COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
 EXPECTED RESPONSE 7E  
 ACTUAL RESPONSE 00

The DRIVE CLEAR command did not return an expected response code. The expected response and actual response are in hex.

### 3.2.5 TEST 3 INFORMATIONAL MESSAGES

-----

UNIT xx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 LOGGABLE INFORMATION AFTER RECAL  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After sending a RECALIBRATE command, the ATTENTION bit was set. Test 3 then sent a GET STATUS command and found the LOGGABLE INFORMATION bit was set. This is not an error, it is only some information being sent from the drive. Normal operation continues.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

3.2.6 TEST 3 ERROR MESSAGES (03000 TO 03999)  
-----

03001 CZUDC HRD ERR 03001 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT ON SEND  
COMMAND WAS command  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

If test 3 tries to send a level 2 command to the drive, and receiver ready is deasserted, error 3001 occurs.  
Where command is one of the following:

GET COMMON CHARACTERISTICS  
ONLINE  
DRIVE CLEAR  
DISCONNECT  
GET SUBMIT CHARACTERISTICS  
GET STATUS  
CHANGE MODE  
INITIATE RECALIBRATE  
SPIN UP

REAL TIME STATE state: REAL TIME STATE 0003

The real time state is the real time drive state <<AFTER>> Test 3 detected the error. <<THIS VALUE IS DISPLAYED IN HEX>>. In this example, receiver ready and attention are both asserted.

The bit positions are defined as follows:

0001 - Receiver ready (Test 3 able to transmit to drive)  
0002 - Attention (error occurred or online timeout expired)  
0040 - Available (drive offline and unusable)  
1000 - Read/Write ready

The complete meaning of these bits is beyond the scope of this text, please refer to the operator documentation for the drive you are working on.

STATUS (R TO L): word6 word5 word4 word3 word2 word1 word0:

The status is the response to the SDI GET STATUS command. These words are printed in HEX. <<NOTE THAT THE STATUS IS PRINTED OUT FROM RIGHT TO LEFT!>>. The status' meaning is beyond the scope of this text, please refer to the operator documentation for the drive you are working on.

03002 CZUDC HRD ERR 03002 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
TIME-OUT OF RECEIVE  
COMMAND WAS GET COMMON CHARACTERISTICS  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

This error is a failure of the drive to respond to an SDI level 2 command (see the SDI specification) before the drive-supplied command timeout expires.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03003 CZUDC HRD ERR 03003 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
FIRST WORD RECEIVED WAS NOT A START FRAME  
COMMAND WAS GET COMMON CHARACTERISTICS  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The first word received by the UDA from the drive was not a valid message start frame.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03004 CZUDC HRD ERR 03004 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
FRAMING ERROR ON LEVEL 0 RESPONSE  
COMMAND WAS GET COMMON CHARACTERISTICS  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Error 3004 is caused by one or more of the following conditions: 1) Illegal frame code -- the frame is not a message start, continue, or end frame. 2) Illegal sequence of frames -- such as a message start frame without ever receiving a message end frame. This can be caused by the drive sending a response before the UDA asserts receiver ready, or a random hit on the SDI cable that garbles a frame or a bad drive transmitter or UDA receiver.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03005 CZUDC HRD ERR 03005 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
CHECKSUM ERROR ON LEVEL 0 RESPONSE  
COMMAND WAS GET COMMON CHARACTERISTICS  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The checksum attached to a message end frame did not match the checksum computed over the level 2 command. This could be caused by a bad drive transmitter, bad UDA receiver, incorrectly computed checksum by the drive (unlikely) or a random hit on the SDI cable.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03006 CZUDC HRD ERR 03006 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
RESPONSE LONGER THAN EXPECTED  
COMMAND WAS GET COMMON CHARACTERISTICS  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The buffer size set aside for the response was not large enough for the response received. This is caused by the drive sending a response that is incorrect for the request sent to the drive, or the drive sending some garbage with

the response.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03007 CZUDC HRD ERR 03007 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxxx UDA AT xxxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
CODE FROM RECEIVE WAS UNINTELLIGIBLE FROM SUBSYSTEM = 0000  
COMMAND WAS GET COMMON CHARACTERISTICS  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The unknown error code occurs when the UDA returns an error code from an operation that test 3 does not recognize. Possible UDA microcode change without test 3 update.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03008 CZUDC HRD ERR 03008 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxxx UDA AT xxxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
COMMAND DID NOT RETURN EXPECTED RESPONSE CODE  
COMMAND WAS GET COMMON CHARACTERISTICS  
EXPECTED RESPONSE 7E  
ACTUAL RESPONSE 7D  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

This is caused by receiving an UNSUCCESSFUL response from the drive, or the drive sending some response other than the correct response for the request sent to the drive. See the contents of status for the unexpected response error (or reason).

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03009 CZUDC HRD ERR 03009 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxxx UDA AT xxxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DRIVE NOT ASSERTING RECEIVER READY IN DRIVE STATE  
REAL TIME STATE 0002  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Test 3 initiates the drive and checks the drive's real time state. If RECEIVER READY was not asserted after a period of time this error message is printed.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03011 CZUDC HRD ERR 03011 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxxx UDA AT xxxxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
NO VALID STATE FROM DRIVE  
NO DRIVE CLOCKS  
CHECK THAT DRIVE IS POWERED ON.

If test 3 attempts to get the drive state, and finds that there are no drive clocks on the port, the above message is occurs. This error usually means that the SDI cable is not connected, the drive is not powered on or the drive's port button that connects it to this UDA is not

depressed.

03012 CZUDC HRD ERR 03012 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
NO VALID STATE FROM DRIVE  
HARD PARITY OR PULSE ERROR FOR 1/2 A SECOND

If test 3 attempts to get the drive state, and gets pulse or parity errors for a full 1/2 second, the above message is printed. This error usually indicates a poor connection or grounding of the SDI cables, a bad drive transmitter, a bad UDA receiver or a broken SDI cable.

03014 CZUDC HRD ERR 03014 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
SUBUNIT CHARACTERISTICS SAY THERE ARE ZERO READ ONLY GROUPS  
IN THE DIAGNOSTIC AREA

After interrogating the subunit characteristics, test 3 finds out that the drive claims there are zero read only groups in the diagnostic area. There must be at least one for the test to run.

03015 CZUDC HRD ERR 03015 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
SUBUNIT CHARACTERISTICS SAY THERE ARE LESS THAN 1 READ/WRITE  
GROUPS IN THE DIAGNOSTIC AREA

After interrogating the subunit characteristics, test 3 finds out that the drive claims there are zero read/write groups in the diagnostic area. There must be at least one for the test to run.

03016 CZUDC HRD ERR 03016 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
NEITHER R/W READY NOR ATTENTION SET AFTER RECALIBRATE COMMAND  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After a RECALIBRATE command, R/W READY or ATTENTION did not set. Check the state for further information. This could be caused by a bad transmitter or receiver or by a hit on the SDI cable.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03017 CZUDC HRD ERR 03017 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
SUBUNIT CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER

After interrogating the subunit characteristics, test 3 finds out that the drive claims there are zero diagnostic cylinders. There must be at least one for the test to run.

03018 CZUDC HRD ERR 03018 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
READ/WRITE READY DROPPED BEFORE FORMAT OPERATION  
CYLINDER aa. GROUP bb. TRACK cc.

REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The R/W READY signal was deasserted by the drive before a format operation was going to be sent by the UDA. The drive may have gone off line or is not transmitting properly or the UDA may not be receiving properly or the SDI cable took a hit.

Where:

aaa is the cylinder value in decimal.  
bb is the group value in decimal.  
cc is the track value in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03019 CZUDC HRD ERR 03019 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
FORMAT OPERATION REPORTED TIME-OUT FAILURE  
CYLINDER aaa. GROUP bb. TRACK cc.  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The format operation sent by the UDA failed. The command timed out possibly due to receiver ready being dropped or communication problem (bad transmitter or receiver or hit on the SDI cable)

Where:

aaa is the cylinder value in decimal.  
bb is the group value in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03020 CZUDC HRD ERR 03020 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
AFTER RECAL, ERROR BITS WERE SET  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After sending a RECALIBRATE command, the ATTENTION bit was set. Test 3 then sent a GET STATUS command and found the error bits were set. For further information, check the state and the status.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03022 CZUDC HRD ERR 03022 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
READ/WRITE READY DROPPED BEFORE WRITE OPERATION  
CYLINDER aaa. GROUP bb. TRACK cc  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The R/W READY signal was deasserted by the drive before a write operation was going to be sent by the UDA.

The drive may have gone off line or is not transmitting properly or the UDA may not be receiving properly or the SDI cable took a hit.

Where:

aaa is the cylinder value in decimal.  
bb is the group value in decimal.  
cc is the track value in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03023 CZUDC HRD ERR 03023 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME nhh:mm:ss  
COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:  
WRITE OPERATION REPORTED FAILURE -- ERROR CODE aaa OCTAL.  
DBN bbb. CYLINDER ccc. GROUP dd. TRACK ee.  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After each track in the diagnostic space is formatted, at least one block must be able to have data written to it and read from it and the data must be correct. Not one block (DBN bbb.) from track (ee) was able to pass. The error code (aaa) gives the reason for the write operation failure.

Where:

aaa is the error code in octal.  
It may have one of the following values:  
2 = drive failure  
3 = requested LBN is a secondary revector.  
<<< NOTE >>> We are working with DBN's  
4 = header compare failure  
(desired header not found)  
153 = suspected positioner error  
213 = read/write ready failure  
253 = drive data or state clock timeout  
(indicates cable/transmitter/  
receiver broken)  
313 = receiver ready timeout  
413 = drive state receive error during write  
bbb is the DBN in decimal.  
ccc is the cylinder value in decimal.  
dd is the group value in decimal.  
ee is the track value in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03024 CZUDC HRD ERR 03024 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
READ/WRITE READY DROPPED BEFORE READ OPERATION  
CYLINDER aaa. GROUP bb. TRACK cc.  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The R/W READY signal was deasserted by the drive before

a read operation was going to be sent by the UDA.  
The drive may have gone off line or is not transmitting properly or the UDA may not be receiving properly or the SDI cable took a hit.

Where:

aaa is the cylinder value in decimal.  
bb is the group value in decimal.  
cc is the track value in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03025 CZUDC HRD ERR 03025 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:  
READ OPERATION REPORTED FAILURE -- ERROR CODE aaa OCTAL.  
CYLINDER ccc. GROUP dd. TRACK ee.  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After each track in the diagnostic space is formatted, at least one block must be able to have data written to it and read from it and the data must be correct. No block from track (ee) was able to pass. The error code (aaa) gives the reason for the read operation failure.

Where:

aaa is the error code in octal.  
It may have one of the following values:  
2 = drive failure  
3 = requested LBN is a secondary revector.  
<<< NOTE >>> We are working with DBN's  
4 = header compare failure  
(desired header not found)  
52 = SERDES overrun error  
150 = data sync timeout on read  
153 = suspected positioner error  
213 = read/write ready failure  
253 = drive data or state clock timeout  
(indicates cable/transmitter/  
receiver broken)  
313 = receiver ready timeout  
413 = drive state receive error during write  
ccc is the cylinder value in decimal.  
dd is the group value in decimal.  
ee is the track value in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03026 CZUDC HRD ERR 03026 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:  
DATA COMPARE FAILURE ON WORD aa.  
EXPECTED DATA bbbb  
ACTUAL DATA cccc  
CYLINDER ddd. GROUP ee. TRACK ff.

After each track in the diagnostic space is formatted, at least one block must be able to have data written to it and read from it and the data must be correct. Not one block (DBN bbb.) from track (ee) was able to pass. The data read did not match the data written.

Where:

aa is the offset in decimal into the buffer where the error occurred.  
 bbbb is the expected data in hex.  
 cccc is the actual data in hex.  
 ddd is the cylinder value in decimal.  
 ee is the group value in decimal.  
 ff is the track value in decimal.

03027 CZUDC HRD ERR 03027 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
 DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 SEEK COMPLETE TIME-OUT -- READ/WRITE READY DID NOT SET  
 SEEK WAS TO CYLINDER aaa. GROUP bb.  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After a SEEK command has been successfully sent from the UDA to the drive, the signal READ/WRITE READY must be set to indicate that the seek completed. If READ/WRITE READY never is asserted by the drive after the seek, the seek times out and error 3027 is presented.

Where:

aaa is the cylinder in decimal.  
 bb is the group in decimal.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03028 CZUDC HRD ERR 03028 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
 DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 NO BLOCK ON THIS TRACK CAN BE READ. LAST BLOCK TRIED:  
 aBN bbbb. CYLINDER ccc. GROUP dd. TRACK ee.

After a seek to a track, at least one block must be able to be read to assure that test 3 can read the header. If not one block was successful, error message 3028 appears.

Where:

a is 'L' for LBN, 'D' for DBN, or 'X' for XBN.  
 bbbb is the block number in decimal.  
 ccc is the cylinder in decimal.  
 dd is the group number in decimal.  
 ee is the track number in decimal.

03029 CZUDC HRD ERR 03029 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
 DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
 AVAILABLE WAS NOT ASSERTED AFTER DISCONNECT  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

After the DISCONNECT command was sent, the AVAILABLE flag should be asserted after a period of time. If it never was, then error 3029 appears. There maybe a problem with a transmitter or a receiver or the SDI cable at this point.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03030 CZUDC HRD ERR 03030 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
INVALID LEVEL 2 COMMAND OPCODE aaaa WAS SUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Some invalid level 2 commands are sent over the SDI. The drive should find these illegal commands and flag them as such. If the drive doesn't, then error 3030 will appear.

Where aaaa is the invalid command in hex.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03031 CZUDC HRD ERR 03031 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
COMMAND WITH type LENGTH = a WAS SUCCESSFUL  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

SDI level 2 commands with invalid lengths are sent to the drive to check if the drive can find them.

Where:

type could be 'COMMAND' or 'RESPONSE' for which  
field was affected  
a is the invalid length

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03032 CZUDC HRD ERR 03032 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
UNIT DID NOT REPORT TRANSMISSION ERROR  
WHEN reason  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Invalid level 1 sequences were sent to the drive. Several sequences are tried and the drive should find fault with everyone of them.

Where reason could be one of the following:

AN END FRAME WAS SENT AFTER A START FRAME TIMED OUT  
A CONTINUE OR END FRAME DID NOT FOLLOW A START FRAME  
AN END FRAME WAS SENT WITH NO START FRAME  
AN END FRAME WITH A BAD CHECKSUM WAS SENT  
A CONTINUE FRAME WAS SENT WITH NO START FRAME

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03033 CZUDC HRD ERR 03033 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
UNIT ACCEPTED AN INVALID GROUP NUMBER FROM GROUP SELECT LEVEL 1  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

A level 1 select group command with an illegal group number is sent to the drive. If the drive accepted it, then error 3033 will be displayed.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03035 CZUDC DVC FTL ERR 03035 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
SUCCESSFULLY WRTE ON DBN AREA WHEN DRIVE WAS WRITE PROTECTED  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

An attempt was made to write on a write protected drive. It should have resulted in an error response from the disk drive, but it didn't.

Check 03001 for explanation of 'REAL TIME STATE' and 'STATUS'

03036 CZUDC DVC FTL ERR 03036 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DRIVE IS NOT PROPERLY FORMATTED.  
UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION  
THIS DRIVE NEEDS TO BE FORMATTED.

Test 3 reads a copy of the FCI in the XBN area and determined that the FCI was corrupted. Any normal operating system (which uses the UDA as a controller) will spin down the drive, so the drive will need to be reformatted.

03037 CZUDC DVC FTL ERR 03037 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
DRIVE IS FORMATTED IN 576 BYTE MODE.  
TO RUN WITH A UDA, THIS DRIVE NEEDS TO BE FORMATTED IN 512 BYTE MODE.  
UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION  
THIS DRIVE NEEDS TO BE FORMATTED.

Test 3 reads a copy of the FCI from the XBN area and determined that the drive was formatted in 576 byte mode. Any normal operating system (which uses the UDA as a controller) will spin down the drive, so the drive will need to be reformatted.

03038 CZUDC DVC FTL ERR 03038 ON UNIT 00 TST 003 SUB 000 PC: xxxxxx  
DISK FUNCTION DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
NO COPY OF THE FCI COULD BE READ.  
UDA WILL SPIN DOWN THIS DRIVE IF USED IN NORMAL SYSTEM OPERATION  
THIS DRIVE NEEDS TO BE FORMATTED.

Test 3 attempted to read every copy of the FCI without success. Any normal operating system (which uses the UDA as a controller) will spin down the drive, so the drive will need to be reformatted

3.2.7 TEST 4 INFORMATIONAL MESSAGES  
-----

UNIT u UDA AT ccccc DRIVE n    RUNTIME hh:mm:ss  
A CORRECTABLE ECC ERROR EXISTS IN type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder

The above message occurs when Test 4 1) detects an ECC error and 2) is able to correct it, and 3) the corrections are less than the drive ECC threshold, (a SDI DRIVE CHARACTERISTIC) and 4) the EDC computed over the corrected sector matched the EDC read.

UNIT unit UDA AT udeadr DRIVE plug    RUNTIME hh:mm:ss  
INITIAL WRITE COMPLETE

Whenever Test 4 is STArted with initial write enabled, <<OR>> whenever it is STArted or REStArted and the diagnostic area is being tested on a drive not in read only mode, the disk will be initially written. The above message occurs when the initial write completes.

UNIT unit UDA AT udeadr DRIVE plug    RUNTIME hh:mm:ss  
READ ONLY DRIVE, INITIAL WRITE WILL NOT BE PERFORMED

If an initial write is to be performed (see above for conditions) and a unit or subunit is in read only mode, (can be set in the manual intervention questions) an initial write will not be performed, and this message will print to inform the operator.

NOTE: DATA COMPARE ERRORS RESULT IF THE DISK IS NOT INITIALLY WRITTEN!!

UNIT unit UDA AT udeadr DRIVE plug    RUNTIME hh:mm:ss  
THE PREVIOUS DEVICE FATAL WILL CAUSE THE FOLLOWING DRIVES  
TO BE DROPPED: plug, plug+1, plug+2, plug+3

plug:                    drive plug number -- each subunit's plug number is  
                          displayed. for a single subunit drive (such as  
                          and RA80) only one plug number is displayed.

If a device fatal error occurs and dropping is enabled, <<ALL>> subunits on the unit that the device fatal occurred must be dropped. To inform the operator, this message is printed after the device fatal error message.

NOTE: IF MORE THAN ONE UDA IS ON A SYSTEM, THIS MESSAGE MAY NOT IMMEDIATELY FOLLOW THE DEVICE FATAL IF AN ERROR HAPPENS AT THE SAME TIME ON ANOTHER UDA.

3.2.8 TEST 4 ERROR MESSAGES (04000 TO 04999)  
-----

04001 CZUDC SFT ERR 04001 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
ATTN ASSERTED DURING SEEK  
SEEK FROM GRP group CYL cylinder TO GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

This error occurs when the drive asserts the SDI ATTENTION signal without asserting the READ/WRITE READY signal, indicating the unsuccessful completion of a seek.

See retry/recovery section for recovery details.

04002 CZUDC SFT ERR 04002 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
ATTN ASSERTED UNEXPECTEDLY, ASYN DRIVE ERROR OR LOGGABLE  
INFORMATION  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

This is an asynchronous drive error. Asynchronous drive errors are those errors reported by the drive which are not related to a level 2 command. These errors are reported by the drive using the SDI ATTENTION signal. The operator must look at the status returned to determine the error that occurred.

See retry/recovery section for recovery details.

04003 CZUDC SFT ERR 04003 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
SEEK DID NOT COMPLETE, NEITHER ATTN OR R/W RDY WAS ASSERTED  
BEFORE TIMEOUT  
SEEK FROM GRP group CYL cylinder TO GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

This error occurs when the drive fails to assert READ/WRITE READY before the seek timeout, which indicates the successful completion of a seek.

See retry/recovery section for recovery details.

04004 CZUDC HRD ERR 04004 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
RCT AREA CORRUPTED, COULD NOT FIND REPLACEMENT FOR  
LBN THAT WAS REVECTORED  
ATTEMPTING TO READ RCT LBN bn  
SEARCHING FOR LBN bn

CZUDC HRD ERR 04004 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx

DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 RCT AREA CORRUPTED, COULD NOT FIND REPLACEMENT FOR  
 LBN WITH HEADER NOT FOUND  
 ATTEMPTING TO READ RCT LBN bn  
 SEARCHING FOR LBN bn

Error 4004 will occur only when Test 4 is running in the customer data area. It occurs when 1) A sector is either marked revectorred or the header can't be found in two revolutions of the disk (both cases should be revectorred) and 2) The replacement for that sector isn't found in the RCT and 3) a NULL entry isn't found at the end of the RCT (see DEC STANDARD 166, Replacement and Caching Table Format). In either case, the subunit should be reformatted, and the cause of the RCT corruption determined.

04005 CZUDC WRD ERR 04005 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 HEADER NOT FOUND DURING WRITE  
 DBN bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder

Error 4005 occurs only when test 4 is writing a DBN or RBN. This is because bad blocks in the diagnostic area are not revectorred, and RBN's are what LBN's are revectorred to, so they should never be bad. Test 4 reports this error if the header being searched for couldn't be found in two revolutions of the disk.

04006 CZUDC SFT ERR 04006 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 SELECT TRACK AND WRITE LEVEL 1 CMD NOT SENT  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Select track and read or write not executed occur when the UDA attempts to send the select track and read/write level 1 cmd, but receiver ready is deasserted or the state is invalid so it cannot send the command (the SERDES could also be broken so it's unable to send the command). The same error is generated if the UDA gets a header sync timeout, and when it looks at the drive's state, it is either invalid or receiver ready is deasserted (header sync timeout is <<NOT>> a error -- it's quite normal on a high-density disk).

See retry/recovery section for recovery details.

04007 CZUDC SFT ERR 04007 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss

ECC DETECTED ERROR  
 RETRY retry  
 ERROR RECOVERY LEVEL level  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder

Error 4007 occurs if an ECC error is detected but ECC correction is disabled.

See retry/recovery section for recovery details.

04008 CZUDC SFT ERR 04008 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 ECC DETECTED ERROR, BUT CORRECTION FAILED  
 RETRY retry  
 ERROR RECOVERY LEVEL level  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder

Error 4008 occurs if an ECC error is detected, but the correction algorithm is unable to correct the errors.

NOTE: THIS IS USUALLY (BUT NOT ALWAYS) INDICATIVE OF A BAD SPOT IN THE ECC RESIDUE AREA AFTER THE DATA AREA OF THE SECTOR.

See retry/recovery section for recovery details.

04009 CZUDC SFT ERR 04009 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 ECC CORRECTIONS EXCEED THRESHOLD  
 RETRY retry  
 ERROR RECOVERY LEVEL level  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder

Error 4009 occurs if an ECC error is detected, the correction algorithm succeeds in correcting the errors, but the number of bits that were corrected exceeds the correction threshold (a SDI DRIVE CHARACTERISTIC).

See retry/recovery section for recovery details.

04010 CZUDC SFT ERR 04010 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 ECC CORRECTION SUCCEEDED, BUT EDC DETECTS ERROR  
 RETRY retry  
 ERROR RECOVERY LEVEL level  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 EDC COMPUTED edc  
 EDC READ edc

edc:           The edc computed and read in octal.

Error 4010 could be caused by several problems:

1) A buffer with a few ECC errors that can be corrected, but the EDC was incorrectly computed or written, or 2) The ECC algorithm incorrectly corrected the buffer and/or the EDC value, (but corrections were less than the threshold) or 3) UDA buffer RAM problem.

See retry/recovery section for recovery details.

04011 CZUDC HRD ERR 04011 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
ERROR RECOVERY TRIED ALL LEVELS WITHOUT SUCCESS  
type bn  
GRP group CYL cylinder

Error 4011 occurs when retries are enabled, and Test 4 has tried all retries on all levels of error recovery. See ECC and EDC retries in the retry/recovery section.

04012 CZUDC HRD ERR 04012 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
DATA COMPARISON FAILED  
ECC OR EDC HAD DETECTED ERROR IN BUFFER  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
PATTERN NUMBER pattern  
OFFSET OF ERROR WITHIN BUFFER: buffer\_offset  
OFFSET OF ERROR WITHIN DISPLAYED LIST: list\_offset (1ST WORD OFFSET 0)  
data0 data1 data2 data3 data4 data5  
data6 data7 data8 data9 data10 data11

CZUDC HRD ERR 04012 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
DATA COMPARISON FAILED  
ECC OR EDC HAD <<NOT>> DETECTED ERROR IN BUFFER  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
PATTERN NUMBER pattern  
OFFSET OF ERROR WITHIN BUFFER: buffer\_offset  
OFFSET OF ERROR WITHIN DISPLAYED LIST: list\_offset (1ST WORD OFFSET 0)  
data0 data1 data2 data3 data4 data5  
data6 data7 data8 data9 data10 data11

pattern:       The pattern number (decimal) that failed the comparison.

buffer\_offset: The offset of the error (decimal) within the sector read, where the first word in the sector is offset 0

list\_offset:   The offset of the error (decimal) within the displayed list, where the first word in the list is offset 0

dataX:           Test 4 displays twelve data words read from the sector.  
                  They are displayed left to right, top to bottom.

Error 4012 occurs when a data compare detects a difference between the buffer read and a known data pattern. The operator is informed if the error was detected by the ECC or EDC. The first word of the sector which may or may not be printed, depending on the position of the error, is the pattern number replicated in each nibble of the word. If a disk is not initially written, it is likely that data comparison failures will occur in the first word of the sector. The following is the first word of the sector for the sixteen different patterns.

pattern	word 0	pattern	word 0
1	010421	9	114631
2	021042	10	125252
3	031463	11	135673
4	042104	12	146314
5	052525	13	156735
6	063146	14	167356
7	073567	15	177777
8	104210	16	000000

Note that pattern 16 is mapped to pattern 0.

04013 CZUDC DEV FTL ERR 04013 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
DRIVE NOT ONLINE TO UDA, AND NOT SPINABLE

If a drive drops offline while being tested (a normal occurrence during Test 4) and some event happens that makes the drive unspinnable (such as the operator popping out the run/stop switch) error 4013 will be printed. If the operator inhibits dropping units, Test 4 will go into error recovery and loop on error 4023, spindle dropped ready.

04014 CZUDC DEV FTL ERR 04014 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
UNABLE TO COMPLETE SEEK -- TRIED 3 TIMES  
type bn  
GRP group CYL cylinder

Once a seek has been attempted 3 times, and never successfully completed, error 4014 will be printed and the entire unit dropped. If the operator inhibits dropping units, the drive will be recalibrated, and the seek will be attempted again.

04015 CZUDC SFT ERR 04015 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
SEEK REQUIRED retries RETRIES BEFORE COMPLETING  
GRP group CYL cylinder

retries:           The number of times the seek was re-issued

If a seek required retries, error 4015 would print to notify the operator.

04016 CZUDC DEV FTL ERR 04016 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
ERRORS DURING DRIVE INITIALIZATION AND SETUP  
THIS UDA AND ALL DRIVES ATTACHED WILL BE REMOVED FROM TESTING

If any errors occur during drive and test initialization, DRIVES ATTACHED TO THE UDA THAT HAD THE DRIVE INITIALIZATION ERRORS WILL NOT BE TESTED. In this case, error 4016 will be printed to notify the operator. THIS ERROR DOES <<NOT>> REFER TO UDA INITIALIZATION. This error is unaffected by the operator inhibiting the dropping of units.

04017 CZUDC DEV FTL ERR 04017 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
NO VALID STATE FROM DRIVE  
NO DRIVE CLOCKS

CZUDC DEV FTL ERR 04017 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
NO VALID STATE FROM DRIVE  
HARD PARITY OR PULSE ERROR FOR 1/2 A SECOND

If Test 4 is <<EVER>> unable to get valid drive state, the drive is immediately dropped, and error 4017 is printed. There are two types of invalid state: no clocks or 'hard' errors. If Test 4 <<EVER>> detects no clocks, the driver is dropped IMMEDIATELY. Parity and pulse errors are normal, so Test 4 tolerates them, <<UNLESS THEY HAPPEN CONTINUOUSLY FOR 1/2 A SECOND>>. If they do occur for 1/2 a second, either the transmitter or receiver is bad, and the drive is dropped. If the operator has inhibited the dropping of units, Test 4 will retry the module that the error occurred on.

04018 CZUDC DEV FTL ERR 04018 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
ATTEMPT TO WRITE ON WRITE PROTECTED DRIVE  
ERROR CODE RETURNED FROM UDA: code  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

code:           The error (in octal) returned to Test 4 from the UDA when Test 4 attempted to write on the write protected drive.

The UDA error codes (in octal) are as follows:

code	error
------	-------

2	SELECT TRACK AND WRITE LEVEL 1 CMD NOT SENT
3	LBN IS REVECTORED
4	HEADER NOT FOUND
153	SEEK OR HEAD SELECT ERROR
213	R/W RDY DROPPED
253	DATA OR STATE CLOCK TIMEOUT
313	RCVR RDY DROPPED
413	REAL TIME STATE RECEIVE ERROR

If Test 4 attempts to write on a write protected drive, error 4018 is printed. Test 4 requires the drive to detect the attempt to write when write protected and return an error for this error to be printed. If the operator has inhibited the dropping of units, a seek will be issued and the write attempted again.

04019 CZUDC HRD ERR 04019 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
HEADER NOT FOUND DURING READ  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder

Error 4019 occurs only when Test 4 is reading a DBN or RBN. This is because bad blocks in the diagnostic area are not revectored, and RBN's are what LBN's are revectored to, so they should never be bad. Test 4 reports this error if the header being searched for couldn't be found in two revolutions of the disk.

04020 CZUDC SFT ERR 04020 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
SELECT TRACK AND READ LEVEL 1 CMD NOT SENT  
ATTEMPT attempt  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Select track and read or write not executed occurs when the UDA attempts to send the select track and read/write level 1 cmd, but receiver ready is deasserted or the state is invalid so it cannot send the command (the SERDES could also be broken so it's unable to send the command). The same error is generated if the UDA gets a header sync timeout, and when it looks at the drive's state, it is either invalid or receiver ready is deasserted (header sync timeout is <<NOT>> a error -- it's quite normal on a high-density disk).

See retry/recovery section for recovery details.

04021 CZUDC DEV FTL ERR 04021 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss

DRIVE NOT FORMATTED IN 512 BYTE MODE -- UNABLE TO TEST  
FCT BLOCK ZERO MODE WORD: mode

\*\*\* THIS PACK HAS AN INVALID FORMAT AND CANNOT BE USED \*\*\*

mode:                   The mode word found on the drive's FCT block zero.

Error 4021 occurs only when Test 4 Finds that the mode word found in FCT block zero is not the 512 byte mode word (126736 octal). See DEC STANDARD 166 "FCT Structure". Inhibiting the dropping of units has no effect on this error.

04022 CZUDC DEV FTL ERR 04022 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
COULD NOT READ FCT BLOCK ZERO

\*\*\* THIS PACK HAS AN INVALID FORMAT AND CANNOT BE USED \*\*\*

Error 4022 occurs when test 4 is unable to read any copy of FCT block zero. See DEC STANDARD 166 "FCT Structure". Inhibiting the dropping of units has no effect on this error.

04023 CZUDC DEV FTL ERR 04023 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
UNABLE TO CONTINUE TESTING  
PORT SWITCH OUT  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

If, during testing, the operator disables the port that Test 4 is using by popping out the port switch, Test 4 prints error 4023. CHANGING THE STATE OF THE PORT SWITCH FOR THE PORT THAT Test 4 IS <<NOT>> USING HAS NO EFFECT ON THE TEST. If dropping of units is inhibited, Test 4 will loop in error recovery, printing this error, until the error state is corrected (by some external action).

CZUDC DEV FTL ERR 04023 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
UNABLE TO CONTINUE TESTING  
RUN/STOP SWITCH OUT  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

If, during testing, the operator pops out the run/stop switch, Test 4 prints error 4023. If dropping of units is inhibited, Test 4 will loop in error recovery, printing this error, until the error state is corrected (by some external action).

CZUDC DEV FTL ERR 04023 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx

DISK EXERCISER DM PC:xxxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 UNABLE TO CONTINUE TESTING  
 SPINDLE DROPPED READY  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

If, during testing, the spindle drops from its ready state, error 4023 is printed. If dropping of units is inhibited, Test 4 will loop in error recovery, printing this error, until the error state is corrected (by some external action).

04024 CZUDC SFT ERR 04024 ON UNIT 00 TST 04 SUB 000 PC: xxxxxxx  
 DISK EXERCISER DM PC:xxxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 EDC DETECTED ERROR BUT ECC DID NOT  
 RETRY retry  
 ERROR RECOVERY LEVEL level  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 EDC COMPUTED edc  
 EDC READ edc

edc: The edc computed and read in octal.

Error 4024 could be caused by several problems. 1) A buffer with no ECC errors, but the EDC was incorrectly computed or written, or 2) UDA buffer RAM problem, or 3) The error is such that the ECC really doesn't detect an error... This is unlikely.

See retry/recovery section for recovery details.

04025 CZUDC HRD ERR 04025 ON UNIT 00 TST 04 SUB 000 PC: xxxxxxx  
 DISK EXERCISER DM PC:xxxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 WRITE ATTEMPTED MAXIMUM TIMES  
 type bn

If three I/O errors occur when attempting to write to the drive (one I/O error if retries are disabled) error 4025 is printed to inform the operator.

04026 CZUDC HRD ERR 04026 ON UNIT 00 TST 04 SUB 000 PC: xxxxxxx  
 DISK EXERCISER DM PC:xxxxx UDA AT xxxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 READ ATTEMPTED MAXIMUM TIMES  
 type bn

If three I/O errors occur when attempting to read from the drive (one I/O error if retries are disabled) error 4026 is printed to inform the operator.

04028 CZUDC DEV FTL ERR 04028 ON UNIT 00 TST 04 SUB 000 PC: xxxxxxx

DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 BOTH READ ONLY <AND> WRITE ONLY BITS SET -- HOST ERROR

Error 4028 prints ONLY IF THERE IS A HOST CODE ERROR -- THIS IS NOT  
 AN ERROR FROM A DRIVE. Inhibiting the dropping of units has no effect  
 on this error.

04034 CZUDC SFT ERR 04034 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 SERDES OVERRUN ERROR DURING READ  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The SERDES overrun error is detected on a read operation and is  
 indicative of a drive whose transfer rate is greater than 23 MHZ  
 or a broken SERDES.

See retry/recovery section for recovery details.

04035 CZUDC SFT ERR 04035 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 DATA OR STATE CLOCK TIMEOUT DURING READ  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The loss of drive clock occurs when the UDA is clocking data to or  
 from the drive through the SERDES. Failure of a word to be  
 clocked in during a 125 millisecond time period triggers a loss of  
 drive clock error.

See retry/recovery section for recovery details.

04036 CZUDC SFT ERR 04036 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 DATA SYNC TIMEOUT DURING READ  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

This error occurs on a read operation after the correct header has

been found and the UDA times out waiting for the data sync word.

See retry/recovery section for recovery details.

04037 CZUDC SFT ERR 04037 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 R/W RDY DROPPED BEFORE/DURING READ  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The loss of read/write ready error is detected either before an I/O has begun when trying to send out the real time command or at the end of an I/O operation when checking for errors.

See retry/recovery section for recovery details.

04038 CZUDC SFT ERR 04038 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 RCVR RDY DROPPED BEFORE/DURING READ  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The loss of drive receiver ready is detected when the UDA is trying to send out a real-time read or write command.

See retry/recovery section for recovery details.

04040 CZUDC HRD ERR 04040 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 ALL COPIES OF RCT READ WITH ERROR, SEARCHING FOR  
 LBN THAT WAS REVECTORED  
 LAST RCT LBN SEARCHED bn  
 SEARCHING FOR LBN bn

CZUDC HRD ERR 04040 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 ALL COPIES OF RCT READ WITH ERROR, SEARCHING FOR  
 LBN WITH HEADER NOT FOUND  
 LAST RCT LBN SEARCHED bn  
 SEARCHING FOR LBN bn

Error 4040 occurs when Test 4 is trying to find the RBN that replaces a LBN that was revectorred or whose header could not be found (both should

be revectored). Test 4 was unable to get a valid copy out of the M copies of the RCT due to I/O errors or ECC/EDC errors. M is a SDI DRIVE CHARACTERISTIC and is defined by the drive. This is indicative of either a bad pack (HDA) or that something wrote over the RCT incorrectly. Try to reformat the subunit.

04041 CZUDC H&D ERR 04041 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
COULD NOT FIND REPLACEMENT FOR  
LBN THAT WAS REVECTORED  
LBN TO REPLACE bn

CZUDC HRD ERR 04041 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
COULD NOT FIND REPLACEMENT FOR  
LBN WITH HEADER NOT FOUND  
LBN TO REPLACE bn

Error 4041 only occurs when Test 4 is running in the customer data area, and is trying to find the RBN that replaces a LBN that was revectored (must be in the RCT) or whose header could not be found (should be in the RCT, unless the media under the header has 'grown' a bad spot recently). In either case, Test 4 was unable to find an entry in the RCT for the the sector and the subunit should be reformatted. In the case of the revectored LBN, the cause of the RCT's corruption should be determined (even with the header not found, the RCT may have been corrupted because a header going bad without warning [eg. the formatter not being able to see it as a weak spot] is a very low probability occurrence).

04042 CZUDC DEV FTL ERR 04042 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
TIMEOUT WAITING FOR SECTOR OR INDEX PULSE  
GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Error 4042 occurs when the UDA microcode never detects a sector or index pulse from the drive before a read or write operation. If dropping of units is inhibited, a seek will be issued, and the write attempted again.

04044 CZUDC SFT ERR 04044 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
SEEK OR HEAD SELECT ERROR DETECTED DURING WRITE  
ATTEMPT attempt  
LBN bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

See error 4045 for description.

See retry/recovery section for recovery details.

04045 CZUDC SFT ERR 04045 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
SEEK OR HEAD SELECT ERROR DETECTED DURING READ  
ATTEMPT attempt  
LBN bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

Errors 4044 and 4045 occur when the header comparison routine determines that the drive is positioned at the wrong physical cylinder, or that the wrong head (which can be cylinders, groups or tracks, or any combination depending on the drive) had been selected. This error only occurs when the drive itself had not detected the misseek or incorrect head selected.

NOTE: These errors will only be detected when the operator is running Test 4 in the customer data area. This error will <<never>> appear when running in the diagnostic area.

See retry/recovery section for recovery details.

04047 CZUDC SFT ERR 04047 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
DATA OR STATE CLOCK TIMEOUT DURING WRITE  
ATTEMPT attempt  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The loss of drive clock occurs when the UDA is clocking data to or from the drive through the SERDES. Failure of a word to be clocked in during a 125 millisecond time period triggers a loss of drive clock error.

See retry/recovery section for recovery details.

04048 CZUDC SFT ERR 04048 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
R/W RDY DROPPED BEFORE/DURING WRITE  
ATTEMPT attempt  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The loss of read/write ready error is detected either before an I/O has begun when trying to send out the real time command or at

the end of an I/O operation when checking for errors.

See retry/recovery section for recovery details.

04049 CZUDC SFT ERR 04049 ON UNIT 00 1ST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
RCVR RDY DROPPED BEFORE/DURING WRITE  
ATTEMPT attempt  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The loss of drive receiver ready is detected when the UDA is trying to send out a real-time read or write command.

See retry/recovery section for recovery details.

04050 CZUDC DEV FTL EPR 04050 ON UNIT 00 1ST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
BEGIN/END SET STARTING BLOCK NUMBER GREATER THAN ENDING BLOCK NUMBER

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the BEGIN/END set questions. Inhibiting the dropping of units has no effect on this error.

04051 CZUDC DEV FTL ERR 04051 ON UNIT 00 1ST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
THE BEGIN/END SETS OVERLAP

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the BEGIN/END set questions. Inhibiting the dropping of units has no effect on this error.

04052 CZUDC DEV FTL ERR 04052 ON UNIT 00 1ST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
BEGIN/END SET ENDING BLOCK NUMBER EXCEEDS MAXIMUM  
MAXIMUM BLOCK NUMBER ON DEVICE IS maximum\_block\_number

max\_block\_number: This is the highest block number the operator can specify.

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the BEGIN/END set questions. Inhibiting the dropping of units has no effect on this error.

04053 CZUDC DEV FTL ERR 04053 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
DUPLICATE BAD BLOCKS

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the BAD BLOCK questions. Inhibiting the dropping of units has no effect on this error.

04054 CZUDC DEV FTL ERR 04054 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
BAD BLOCK NUMBER EXCEEDS MAXIMUM. MAXIMUM BLOCK NUMBER  
ON DEVICE IS maximum\_block\_number

maximum\_block\_number: This is the highest block number the operator can specify.

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the BAD BLOCK questions. Inhibiting the dropping of units has no effect on this error.

04055 CZUDC DEV FTL ERR 04055 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
STARTING CYLINDER GREATER THAN ENDING CYLINDER

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the STARTING AND ENDING CYLINDER questions. Inhibiting the dropping of units has no effect on this error.

04056 CZUDC DEV FTL ERR 04056 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
RANDOM AND SEQUENTIAL SEEKS CANNOT BE MIXED WITHIN A UNIT

Error 4056 is an operator error. The error occurs on a multiple subunit drive when one subunit is selected to run in random mode, and another is selected to run in sequential mode. This mix is not supported, so the above message is issued. Inhibiting the dropping of units has no effect on this error.

04057 CZUDC DEV FTL ERR 04057 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
OVERFLOW WHEN CALCULATING THE L/DBW FROM THE GIVEN CYLINDER  
CYLINDER TOO LARGE

This is a Test 4 initialization error due to an operator error.  
The operator entered a cylinder number, that when converted to a block  
number, the block number exceeded  $(2^{28}) - 1$ . Go back  
to the manual intervention questions and check the answers to the  
STARTING AND ENDING CYLINDER questions. Inhibiting the dropping of units  
has no effect on this error.

04058 CZUDC DEV FTL ERR 04058 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
TRACK EXCEEDS MAXIMUM FOR DEVICE. MAXIMUM IS maximum\_track

maximum\_track: This is the highest track number the operator can  
specify.

This is a Test 4 initialization error due to an operator error. Go back  
to the manual intervention questions and check the answers to the  
TRACK questions. Inhibiting the dropping of units has no effect  
on this error.

CZUDC DEV FTL ERR 04058 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
GROUP EXCEEDS MAXIMUM FOR DEVICE. MAXIMUM IS maximum\_group

maximum\_group: This is the highest group number the operator can  
specify.

This is a Test 4 initialization error due to an operator error. Go back  
to the manual intervention questions and check the answers to the  
GROUP questions. Inhibiting the dropping of units has no effect  
on this error.

04059 CZUDC DEV FTL ERR 04059 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
TWO IDENTICAL TRACKS

This is a Test 4 initialization error due to an operator error. Go back  
to the manual intervention questions and check the answers to the  
TRACK questions. Inhibiting the dropping of units has no effect  
on this error.

CZUDC DEV FTL ERR 04059 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss

OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
TWO IDENTICAL GROUPS

This is a Test 4 initialization error due to an operator error. Go back to the manual intervention questions and check the answers to the GROUP questions. Inhibiting the dropping of units has no effect on this error.

04062 CZUDC DEV FTL ERR 04062 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
DBN COMPUTED FROM END CYLINDER GIVEN EXCEEDS MAXIMUM DBN NUMBER ON  
DEVICE - CYLINDER TOO LARGE

This is a Test 4 initialization error.  
Note that though there may be writeable DBN's on the 'last' cylinder, the read only diagnostic area may start on that same cylinder, and Test 4 tries to write to the end of the cylinder that the operator specified. Therefore, specify the previous cylinder if cylinders must be specified. Inhibiting the dropping of units has no effect on this error.

CZUDC DEV FTL ERR 04062 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
OPERATOR ERROR IN ANSWERING MANUAL INTERVENTION QUESTIONS FOR THIS UNIT  
LBN COMPUTED FROM END CYLINDER GIVEN EXCEEDS MAXIMUM LBN NUMBER ON  
DEVICE - CYLINDER TOO LARGE

This is a Test 4 initialization error.  
Note that though there may be writeable LBN's on the 'last' cylinder, the RCT area may start on that same cylinder, and Test 4 tries to write to the end of the cylinder that the operator specified. Therefore, specify the previous cylinder if cylinders must be specified. Inhibiting the dropping of units has no effect on this error.

04063 CZUDC SFT ERR 04063 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
REAL TIME STATE RECEIVE ERROR DURING WRITE  
ATTEMPT attempt  
type bn  
SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
ORIGIN OF SEEK: GRP group CYL cylinder  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0976 0706 0504 0302 0100

The real time drive state receive error is detected at the end of an I/O operation and indicates that there was a pulse or parity error in the receipt of the drive's state during the I/O operation.

See retry/recovery section for recovery details.

04064 CZUDC SFT ERR 04064 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 REAL TIME STATE RECEIVE ERROR DURING READ  
 ATTEMPT attempt  
 type bn  
 SECTORS FROM INDEX sector TRK track GRP group CYL cylinder  
 ORIGIN OF SEEK: GRP group CYL cylinder  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

The real time drive state receive error is detected at the end of an I/O operation and indicates that there was a pulse or parity error in the receipt of the drive's state during the I/O operation.

See retry/recovery section for recovery details.

04068 CZUDC HRD ERR 04068 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 UNKNOWN ERROR CODE DURING WRITE  
 ERROR CODE RETURNED error\_code  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

error\_code: This is the error code returned to Test 4 by the UDA that Test 4 does not recognize.

The unknown error code occurs when the UDA returns an error code from an operation that Test 4 does not recognize. Possible UDA microcode change without Test 4 update.

See retry/recovery section for recovery details.

04069 CZUDC HRD ERR 04069 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
 UNKNOWN ERROR CODE DURING READ  
 ERROR CODE RETURNED error\_code  
 REAL TIME STATE 0003  
 STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

error\_code: This is the error code returned to Test 4 by the UDA that Test 4 does not recognize.

The unknown error code occurs when the UDA returns an error code from an operation that Test 4 does not recognize. Possible UDA microcode change without Test 4 update.

See retry/recovery section for recovery details.

04070 CZUDC SFT ERR 04070 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
 DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss

## TIMEOUT OF SEND

command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

If test 4 tries to send a level 2 command to the drive, and receiver ready is deasserted, error 4070 occurs.

See retry/recovery section for recovery details.

04071 CZUDC SFT ERR 04071 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
TIMEOUT OF RECEIVE  
command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

This error is a failure of the drive to respond to an SDI level 2 command (see the SDI specification) before the drive-supplied command timeout expires.

See retry/recovery section for recovery details.

04072 CZUDC SFT ERR 04072 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
FIRST WORD RECEIVED WAS NOT START FRAME  
command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4071 for a description

The first word received by the UDA from the drive was not a valid message start frame.

See retry/recovery section for recovery details.

04073 CZUDC SFT ERR 04073 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
FRAMING ERROR ON LEVEL 0 RECEIVE  
command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

Error 4073 is caused by one or more of the following conditions:

1) Illegal frame code -- the frame is not a message start, continue, or end frame. 2) Illegal sequence of frames -- such as a message start frame without ever receiving a message end frame. This can be caused by the drive sending a response before the UDA asserts receiver ready, or a random hit on the SDI cable that garbles a frame or a bad drive transmitter or UDA receiver.

See retry/recovery section for recovery details.

04074 CZUDC SFT ERR 04074 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
CHECKSUM ERROR ON LEVEL 0 RECEIVE  
command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

The checksum attached to a message end frame did not match the checksum computed over the level 2 command. This could be caused by a bad drive transmitter, bad UDA receiver, incorrectly computed checksum by the drive (unlikely) or a random hit on the SDI cable.

See retry/recovery section for recovery details.

04075 CZUDC SFT ERR 04075 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
BUFFER SIZE SMALLER THAN LEVEL 2 RESPONSE  
command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

The buffer size set aside for the response was not large enough for the response received. This is caused by the drive sending a response that is incorrect for the request sent to the drive, or the drive sending some garbage with the response.

See retry/recovery section for recovery details.

04076 CZUDC SFT ERR 04076 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
RESPONSE OF LEVEL 2 CMD NOT AS EXPECTED  
command\_type  
EXPECTED RESPONSE expected\_response  
RESPONSE RECEIVED response\_received  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

expected\_response: This is the correct response (HEX) for the command.  
response\_received: This is the response received from the drive, (HEX)  
where a 7D is an unsuccessful response. Any other  
than a 7D for this value indicates a <<VERY>> sick  
drive.

This is caused by receiving an UNSUCCESSFUL response from the drive, or  
the drive sending some response other than the correct response for the  
request sent to the drive. See the contents of status for the unexpected  
response error (or reason).

See retry/recovery section for recovery details.

04077 CZUDC HRD ERR 04077 ON UNIT 00 TST 04 SUB 000 PC: x<<.xxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
DRIVE NEVER DEASSERTED RECEIVER READY AFTER LEVEL 2 SEND  
command\_type  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

This is caused by the drive not seeing a command sent by  
the UDA. The drive must deassert receiver ready to acknowledge  
that it did see a command via the SDI. If the drive saw only  
part of the command, it would have marked the command as  
unsuccessful. But in this case, the drive did not see any  
of the command and is now waiting for a command to be sent  
from the UDA.

04078 CZUDC HRD ERR 04078 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
UNKNOWN ERROR CODE RETURNED FROM LEVEL 2 RECEIVE  
command\_type  
ERROR CODE RETURNED error\_code  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

command\_type: See section following error 4078 for a description

error\_code: This is the error code returned to Test 4 by the UDA  
that Test 4 does not recognize.

The unknown error code occurs when the UDA returns an error code from  
an operation that Test 4 does not recognize. Possible UDA microcode  
change without Test 4 update.

See retry/recovery section for recovery details.

NOTE: Errors 4070 - 4078 will become device fatals if attempted 3 times.  
If dropping of units are inhibited, error recovery is the same as

if the error was a soft error.

command\_type: in errors 4070-4078 command\_type is one of the following  
level 2 commands:

ATTEMPTING TO BRING DRIVE ONLINE  
ATTEMPTING TO ISSUE SEEK  
ATTEMPTING TO GET STATUS  
ATTEMPTING DRIVE CLEAR CMD  
ATTEMPTING TO BRING DRIVE ONLINE  
ATTEMPTING TO CHANGE MODE  
ATTEMPTING ERROR RECOVERY CMD  
ATTEMPTING TO ISSUE SEEK  
ATTEMPTING TO RECALIBRATE

The following commands\_types occur only during  
initialization, and will cause a device fatal if  
they occur. Inhibiting the dropping of units has no  
effect on these errors.

ATTEMPTING TO SPIN UP DRIVE  
ATTEMPTING TO GET COMMON CHAR  
ATTEMPTING TO GET SUBUNIT CHAR

If <<ANY>> error occurs during initialization, <<NO>> testing  
is done on <<ANY>> drive attached to the UDA that the  
initialization error occurred on. See error number 4016.

3.2.9 SPECIAL DEVICE FATAL (05000)  
-----

05000 CZUDC DVC FTL 05000 ON UNIT 00 TST 002 SUB 000 PC: xxxxxx  
DISK zzzzzzzz DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hhh:mm:ss  
UNABLE TO FIND REQUESTED DRIVE FOR TESTING  
THE FOLLOWING IS VISIBLE ON THE PORTS  
UDA PORT 0 -- description  
UDA PORT 1 -- description  
UDA PORT 2 -- description  
UDA PORT 3 -- description

Where zzzzzzzz is either 'RESIDENT', 'FUNCION' or 'EXERCISER'.  
This message is presented when the specified drive  
was not found by test 2, test 3 or test 4 on any of  
the ports. A description of what was each port follows.

NO DRIVE ATTACHED

- There is nothing on the port. If there is suppose to be a drive on this port, make sure there is an odd number of cables between the UDA and the drive and make sure the cables are properly attached.

RCVR RDY NEVER ASSERTED

- The device on the port did not assert RCVR RDY while trying to get state.

TIMEOUT OF SEND

- Sending an SDI command timed out. RCVR RDY is not asserted.

TIMEOUT OF RECEIVE

- Receiving an SDI command timed out. The drive failed to respond to an SDI level 2 command before a timeout expired.

FIRST WORD RECEIVED WAS NOT START FRAME

- The first word received by the UDA from the drive was not a valid message start frame.

FRAMING ERROR ON LEVEL 0 RECEIVE

- The device and the UDA are out of sync or an illegal frame code (the frame is not a message start, continue, or end frame) or illegal sequence of frames. This can be caused by the drive sending a response before the UDA asserts receiver ready, or a random hit on the SDI cable that garbles a frame or a bad drive transmitter or UDA receiver.

CHECKSUM ERROR ON LEVEL 0 RECEIVE

- The checksum attached to a message end frame did not match the checksum computed over the level 2 command. This could be caused by a bad drive transmitter, bad UDA receiver, incorrectly computed checksum by the drive (unlikely) or a random hit on the SDI cable.

## RESPONSE LONGER THAN EXPECTED FOR CMD

- The buffer size set aside for the response was not large enough for the response received. This is caused by the drive sending a response that is incorrect for the request sent to the drive, or the drive sending some garbage with the response.

DRIVE n[, consecutive drive numbers if subunit drive] [further explanation]  
A drive was found at the end of the cable. It may be a subunit drive, so all the subunit numbers are printed. A further explanation may be presented. These further explanations are:

## DRIVE NOT AVAILABLE TO THIS UDA

- The drive was found but is not available to this UDA. It may be dual ported and the drive is online to another controller.

## UNSPINABLE DRIVE

- The drive is unspinnable. The drive may be powered up but the RUN/STOP switch may be popped out.

### 3.3 TEST 4 RETRY/RECOVERY METHODS

-----

#### ECC Error on Disk Read

ECC DETECTED ERROR, BUT CORRECTION FAILED  
ECC CORRECTIONS EXCEED THRESHOLD  
ECC DETECTED ERROR (IF ECC correction disabled)

Retry/Recovery - The UDA or Test 4 will first re-read the sector with the erroneous ECC N times, then N times for each level of error recovery the drive supports. The value of N is an SDI drive characteristic. This retry mechanism will persist until either the recovery level reaches zero or the operation succeeds. It should be noted that the manual intervention questions can disable retries (in this case the recovery fails the first time) and disable error correction (i.e., no ECC correction will be performed). ECC correction and retries are <<ALWAYS>> enabled when the Test 4 is reading the RCT.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - Test 4 will issue a hard error for the sector. No soft errors will be counted.

#### Error Detecting Code (EDC) Error

EDC DETECTED ERROR BUT ECC DID NOT  
ECC CORRECTION SUCCEEDED, BUT EDC DETECTS ERROR

This error is indicative of a UDA hardware error, either a SERDES failure or an undetected RAM failure, or a sector that was written with an incorrectly computed EDC.

Retry/Recovery - The UDA or Test 4 will re-read the sector with the erroneous EDC N times, then N times for each level of error recovery the drive supports. The value of N is an SDI drive characteristic. This retry mechanism will persist until either the recovery level reaches zero or the operation succeeds. It should be noted that the manual intervention questions can disable retries (in this case the recovery fails the first time). Retries are <<ALWAYS>> enabled when the Test 4 is reading the RCT.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - Test 4 will issue a hard error for the sector. No soft errors will be counted.

## SDI Level 2 and Asynchronous Errors

The SDI level 2 errors are as follows:

- o Packet acknowledge failure
- o Level 2 command error response, "DE" bit set
- o Level 2 command error response, "PE" or "RE" bit set
- o Receipt of erroneous drive response
- o Seek complete timeout
- o Asynchronous drive errors

Level 2 errors are always retried, even if retries are disabled in the manual intervention questions.

In the following retry/recovery algorithms, the Test 4 'Generic error recovery' is the following steps:

1. Issue online command
  2. Get status
    - 2a. If the port, run or spindle ready (PS, RU or SR) bit is deasserted, an Immediate device fatal error is reported and the unit and all its subunits are dropped from testing.
    - 2b. If the recalibrate requested (RR) bit is set, Test 4 will issue a RECALIBRATE, then SEEK <<AFTER>> generic error recovery is complete.
    - 2c. If the drive error (DE) bit is set, Test 4 will issue a SEEK <<AFTER>> generic error recovery is complete.
  3. If no drive errors, go to 5
  4. Send DRIVE CLEAR command
  5. Change mode
- NOTE: If the drive's timeout expires once, so the drive asserts attention just to get Test 4 to issue a level 2, Test 4 will go through the above error recovery. However, since the timeout expiring is not an error, no error message is issued.

## Packet Acknowledge Failure

TIMEOUT OF SEND  
TIMEOUT OF RECEIVE

The timeout of send occurs when the UDA attempts to send a level 2 command to the drive, but the drive's receiver ready is not asserted. Timeout of receive is a failure of the drive to respond to an SDI level 2 command (see the SDI specification) before the drive-supplied command timeout expires. These errors are grouped together because their recoveries are the same.

Retry/Recovery - UDA - The steps listed below are performed.

1. The drive is initialized.
2. An SDI GET STATUS command is issued.
3. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR command.
4. An SDI SEEK command is issued.
5. The command is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. The drive is initialized
2. Test 4 Generic error recovery is performed
3. An SDI SEEK command is issued.
4. The command is retried.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence will be repeated two times and, if the failure persists, the Test 4 will issue a device fatal error and the drive and all its subunits will be dropped. It should be noted that the retry strategy for SDI level 2 errors involves issuing additional level 2 commands. The retry count is the sum of all retries on all SDI level 2 commands, including those commands issued in recovery attempts.

**Level 2 Command Error Response - "DE" Bit Set**

RESPONSE OF LEVEL 2 CMD NOT AS EXPECTED  
SEEK RECEIVED UNSUCCESSFUL RESPONSE

An UNSUCCESSFUL response to a level 2 command, with the "DE" bit set in the status response, notifies the Test 4 that a drive error was detected (or occurred) in connection with the execution of the SDI command.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. The drive error is cleared by an SDI DRIVE CLEAR command and a SEEK command is issued for the cylinder where the drive was positioned when the error was reported.
3. The command is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed  
Note that because the "DE" bit is set, Test 4 generic error recovery will issue a SEEK (see generic error recovery)
2. The command is retried

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if the failure persists, the Test 4 will issue a device fatal error and the drive and all its subunits will be dropped. Note that the retry strategy for SDI level 2 errors involves issuing additional level 2 commands. The retry count is the sum of all retries on all SDI level 2 commands, including those commands issued in recovery attempts.

**Level 2 Command Error Response - "PE" or "RE" Bit Set**

RESPONSE OF LEVEL 2 CMD NOT AS EXPECTED  
SEEK RECEIVED UNSUCCESSFUL RESPONSE

An UNSUCCESSFUL response to a level 2 command with the "PE" or "RE" bit set in the status response notifies the Test 4 that the command either was not appropriate for the state of the drive, or that the command contained invalid arguments.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued
2. The drive error is cleared by an SDI DRIVE CLEAR command.
3. The controller verifies the state of the drive and, if possible, retries the level 2 command. Otherwise, the UDA notifies the host and bypasses subsequent retries.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. The command is retried

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if the failure persists, the Test 4 will issue a device fatal error and the drive and all its subunits will be dropped.

Note that the retry strategy for SDI level 2 errors involves issuing additional level 2 commands. The retry count is the sum of all retries on all SDI level 2 commands, including those commands issued in recovery attempts.

**Receipt of an Erroneous Drive Response**

FIRST WORD RECEIVED WAS NOT START FRAME  
 FRAMING ERROR ON LEVEL 0 RECEIVE  
 CHECKSUM ERROR ON LEVEL 0 RECEIVE  
 BUFFER SIZE SMALLER THAN RESPONSE  
 UNKNOWN ERROR CODE RETURNED FROM LEVEL 2 RECEIVE (hard error)

The first word not start frame error is caused when the UDA does not see a valid message start frame as the first frame received from the drive. The framing error is caused by the UDA receiving an illegal frame code -- the frame is not a message start, continue, or end frame or Illegal sequence of frames -- such as a message start frame without ever receiving a message end frame. The checksum error occurs when a message end frame checksum did not match the checksum computed over the level 2 command. The buffer size smaller than response error occurs when the buffer set aside for the response was not large enough for the response received. The unknown error code is returned when the UDA returns an error code that the Test 4 does not recognize. These errors are grouped together because their recoveries are the same.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR command.
3. The command is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. The command is retried

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if the failure persists, the Test 4 will issue a device fatal error and the drive and all its subunits will be dropped.

Note that the retry strategy for SDI level 2 errors involves issuing additional level 2 commands. The retry count is the sum of all retries on all SDI level 2 commands, including those commands issued in recovery attempts.

**Seek Complete Timeout**

ATTN ASSERTED DURING SEEK  
SEEK DID NOT COMPLETE, NEITHER ATTN OR R/W RDY WAS ASSERTED

This error occurs when the drive fails to assert READ/WRITE READY, indicating the successful completion of a seek, or asserts the SDI ATTENTION signal without asserting the READ/WRITE READY signal, indicating the unsuccessful completion of a seek.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR command.
3. The SEEK is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. The SEEK is retried

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if the failure persists, the Test 4 will issue a device fatal error and the drive and all its subunits will be dropped. Note that the retry strategy for SDI level 2 errors involves issuing additional level 2 commands. The retry count is the sum of all retries on all SDI level 2 commands, including those commands issued in recovery attempts.

## Asynchronous Drive Errors

## ATTN ASSERTED UNEXPECTEDLY, ASYN DRIVE ERROR OR LOGGABLE INFORMATION

Asynchronous drive errors are those errors reported by the drive which are not related to a level 2 or command. These errors are reported by the drive using the SDI ATTENTION signal. Examples are OFF CYLINDER and HDA OVERTEMPERATURE errors. Drive errors are reported to the controller by the "DE" or "WE" bit being set in the error byte in the status response.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. The drive error is cleared by an SDI DRIVE CLEAR command and, if the error is not "WE", a SEEK command is issued for the cylinder where the drive was last positioned.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. A SEEK is issued

NOTE: A "WE" is a write on a write protected drive; Test 4 detects this in a different manner, so "WE" will never be set.

## Recovery Failure -

NOTE: There is a difference between the UDA in controller mode and the Test 4 for this type of error.

The UDA in controller mode will repeat the above sequence two times and, if the drive error persists, the drive would be marked as offline.

Test 4 will <<NOT>> drop the drive after two retries. Instead, the drive will be dropped due to a side effect of such an error: A seek never completing, (causing a device fatal error) or Spindle ready dropping (causing a device fatal error).

### Drive I/O Errors

The drive I/O errors occur either during the header compare process (i.e., before I/O actually begins) or during the I/O operation itself. They are as follows:

- o Header not found
- o Seek or head select error
- o Data sync timeout
- o Data or state clock timeout during operation (read/write)
- o Receiver ready dropped during operation (read/write)
- o Read/write ready dropped during operation (read/write)
- o SERDES overrun error
- o Drive failed to execute select track and (read/write)
- o Real time state receive error

### Header not found (header compare error)

#### HEADER NOT FOUND DURING (read/write)

This error occurs when the header compare routine fails to find the desired header (or a revectored version of the desired header) in two disk revolutions.

Retry/Recovery - UDA and Test 4 - Failure to find the desired header in two rotations of the disk will cause the Test 4 to search the Replacement and Caching table (RCT) to check if the logical block number has been replaced. If a match is found, the Test 4 will perform the desired operation on the revectored block. Enabling/disabling retries has no affect on this operation.

Recovery success - No error is reported or counted.

Recovery Failure - A hard error (header not found) is reported.

Seek or head select error (Positioner Error)

SEEK OR HEAD SELECT ERROR DETECTED DURING (read/write)

This error occurs when the header comparison routine determines that the drive is positioned at the wrong cylinder and that the drive has not detected a seek error.

NOTE: The header comparison routine is active <<ONLY>> in the customer data area. This error will never be detected in the diagnostic area.

Retry/Recovery - JDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR command.
3. An SDI RECALIBRATE command is issued.
4. An SDI SEEK command is issued.
5. The I/O operation is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. An SDI RECALIBRATE command is issued.
3. An SDI SEEK command is issued.
4. If retries are disabled, Immediate recovery failure. Retries are <<ALWAYS>> enabled when the Test 4 is reading the RCT.
5. The I/O operation is retried.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if a drive I/O error persists, a hard error is reported for the sector. No soft errors are counted.

Data Sync Timeout Error

DATA SYNC TIMEOUT DURING READ

This error occurs on a read operation after the correct header has been found and the UDA times out waiting for the data sync word.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR COMMAND.
3. An SDI SEEK command is issued.
4. The read operation is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. An SDI SEEK command is issued.
3. If retries are disabled, Immediate recovery failure. Retries are <<ALWAYS>> enabled when the Test 4 is reading the RCT.
4. The read operation is retried.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if a drive I/O error persists, a hard error is reported for the sector. No soft errors are counted.

Data or state clock timeout (Loss of Drive Clock)  
Receiver ready failure (Loss of Drive Receiver Ready)

DATA OR STATE CLOCK TIMEOUT DURING (read/write)  
RCVR RDY DROPPED DURING (read/write)  
COULD NOT SEND SELECT TRACK AND (read/write) CMD OR  
HEADER SYNC TIMEOUT WITH INVALID STATE

The loss of drive clock occurs when the UDA is clocking data to or from the drive through the SERDES. Failure of a word to be clocked in during a 125 millisecond time period triggers a loss of drive clock error. The loss of drive receiver ready is detected when the UDA is trying to send out a real-time read or write command. Unable to select track and read or write occurs when the UDA attempts to send the select track and read/write level 1 cmd, but receiver ready is deasserted or the state is invalid so it cannot send the command (the SERDES could also be broken so it's unable to send the command). The same error is generated if the UDA gets a header sync timeout, and when it looks at the drive's state, it is either invalid or receiver ready is deasserted (header sync timeout is <<NOT>> a error -- it's quite normal on a high-density disk). These errors are grouped together because their recoveries are the same.

Retry/Recovery - UDA - The steps listed below are performed.

1. The drive is initialized.
2. An SDI GET STATUS command is issued.
3. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR command.
4. An SDI SEEK command is issued.
5. The I/O operation is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. The drive is initialized
2. Test 4 Generic error recovery is performed
3. An SDI SEEK command is issued.
4. If retries are disabled, Immediate recovery failure. Retries are <<ALWAYS>> enabled when the Test 4 is reading the RCT.
5. The I/O operation is retried.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if a drive I/O error persists, a hard error is reported for the sector. No soft errors are counted.

Read/Write ready dropped (Loss of Drive Read/Write Ready)  
SERDES Overrun Error  
Real Time State Receive Error (Real Time Drive State Receive Error)

R/W RDY DROPPED DURING (read/write)  
SERDES OVERRUN ERROR DURING READ  
REAL TIME STATE RECEIVE ERROR DURING (read/write)  
UNKNOWN ERROR CODE DURING (read/write)

The loss of read/write ready error is detected either before an I/O has begun when trying to send out the real time command or at the end of an I/O operation when checking for errors. The SERDES overrun error is detected on a read operation and is indicative of a drive whose transfer rate is greater than 23 MHz or a broken SERDES. The real time drive state receive error is detected at the end of an I/O operation and indicates that there was a pulse or parity error in the receipt of the drive's state during the I/O operation. The unknown error code is returned when the UDA returns an error code that the Test 4 does not recognize. They are grouped together because their recoveries are the same.

Retry/Recovery - UDA - The steps listed below are performed.

1. An SDI GET STATUS command is issued.
2. If the status obtained in the previous step indicated error conditions, these error conditions are resolved and then cleared by an SDI DRIVE CLEAR command.
3. An SDI SEEK command is issued.
4. The I/O operation is retried.

Retry/Recovery - Test 4 - The steps listed below are performed.

1. Test 4 Generic error recovery is performed
2. An SDI SEEK command is issued.
3. If retries are disabled, Immediate recovery failure. Retries are <<ALWAYS>> enabled when the test 4 is reading the RCT.
4. The read operation is retried.

Recovery success - One soft error is counted for the entire operation including retries.

Recovery Failure - The above sequence is repeated two times and, if a drive I/O error persists, a hard error is reported for the sector. No soft errors are counted.

3.4 DEC STANDARD 166 EXCERPTS  
-----

3.4.1 THE REPLACEMENT AND CACHING TABLES  
-----

The Replacement and Caching Tables record the locations of all revectored LBN sectors and the status of each RBN on the unit. Each copy of the table is organized in ascending RBN order, with an entry for each RBN sector on the unit. There are "n" copies of the table on the unit, where "n" is a device characteristic. The tables are stored at the high address end of the LBN area of the unit. Table entries (and RBNs) are allocated via a hash algorithm described later.

Replacement And Caching Table Format -

Each entry in the Replacement and Caching Table represents an RBN on the unit. The table is ordered in ascending RBN order. Thus the first entry corresponds to the first RBN on the unit, etc. The size of each copy of the table may exceed that required to contain an entry for each RBN on the unit since additional entries may be required to align the table so that adjacent copies can begin on a track boundary. Entries that do not correspond to RBNs on the unit are called "null entries"; there is always at least one null entry at the end of the RCT. All other entries past this last null entry are undefined.

NOTE

The RCT pad area is controller specific and should never be accessed by the host.

The format of a replacement block descriptor in the Replacement and Caching Tables is:

```
!<-----16 bits----->!
!
!           LBN (low)
!-----!
! CODE !           LBN (high)
!-----!
! 4 bits!<-----12 bits----->!
```

Where:

LBN is the Logical Block Number of a revectored LBN sector.

CODE is one of the following octal values:

- 00 - Unallocated (empty) replacement block.
- 02 - Allocated replacement block - primary RBN.
- 03 - Allocated replacement block - non-primary RBN.
- 04 - Unusable replacement block.
- \* 05 - Alternate unusable replacement block
- 10 - Null entry - no corresponding RBN sector.

For codes 00, 04, and 10 the LBN field is always zero.

NOTE

\* This code is reserved. Programs should treat this code as if it were code 04.

Embedded-controllers with no distinction between primary and secondary RBN's must use:

1. Code 02 if the replacement block can be retrieved with little degradation of performance for all blocks.
2. Code 03 if accessing the replacement block has a large impact on performance for all blocks.

## 3.4.2 FCT Structure

Each copy of the FCT is composed of one volume information block, one 512 byte format table, one 576 byte format table, and one subsystem temporary storage area (distributed amongst the alignment pads). An FCT copy has the following format:

volume information block	SECTOR 0
128 bad block descriptors 512 mode	SECTOR 1
128 bad block descriptors 512 mode	SECTOR 2
:	
128 bad block descriptors 576 mode	SECTOR m
128 bad block descriptors 576 mode	SECTOR m+1
:	
128 bad block descriptors 576 mode	SECTOR p
subsystem scratch storage	SECTOR p+1
:	
subsystem scratch storage	SECTOR Fct-1

The XBN area itself is always formatted to contain 512 byte sectors. The calculations for m and p are:

$$m := (((Lc * g * t * r) + 1) / 2) + 127) / 128$$

$$p := 2 * m$$

Sector 0 contains various volume identification information. The format is:

media mode	WORD 0
formatting instance number	WORD 1
volume serial number least significant word	WORD 2
volume serial number	WORD 3
volume serial number	WORD 4
volume serial number most significant word	WORD 5
date that volume was first formatted (low)	WORD 6
date that volume was first formatted	WORD 7
date that volume was first formatted	WORD 8
date that volume was first formatted (high)	WORD 9
date of most recent volume formatting (low)	WORD 10
date of most recent volume formatting	WORD 11
date of most recent volume formatting	WORD 12
date of most recent volume formatting (high)	WORD 13
number of used entries in 512 table (low)	WORD 14

number of used entries in 512 table (high)	WORD 15
number of used entries in 576 table (low)	WORD 16
number of used entries in 576 table (high)	WORD 17
XBN of scratch area in this copy (low)	WORD 18
XBN of scratch area in this copy (high)	WORD 19
size of scratch area in this copy	WORD 20
zeros	
zeros	WORD 255

## Where:

WORD 0: "Media Mode" - is "126736" for a 512 byte format and "074161" for a 576 byte format. During formatting the media mode word is set to zero.

4.0 PERFORMANCE AND PROGRESS REPORTS  
-----

At the end of each pass, the pass count is given along with the total number of errors reported since the diagnostic was started. The "EOP" switch can be used to control how often the end of pass message is printed. Section 2.2 describes switches.

A statistical report will automatically be printed periodically (approximately every fifteen minutes) and at the end of test #4. It can be suppressed by setting the Inhibit Statistical Report flag (e.g. START/FLAGS:ISR). This is the same report that can be printed on demand with the PRINT command.

During tests 1, 2, and 3, the report will look like the following example:

TEST 1 IN PROGRESS      RUN TIME 2:24:10

During test #4, the report will contain statistics on each drive for the current pass of the test; for example:

TEST 4 IN PROGRESS      RUN TIME 2:24:10

UNIT	DRIVE	SERIAL-NUMBER	SEEKS X1000	MBYTES READ	MBYTES WRITTEN	HARD ERRORS	SOFT ERRORS	ECC
0	0	1002	12	36	22	0	0	1
1	4	7342102112	14	42	29	0	2	0

Explanation of each column:

UNIT	The unit number (number of HW P-table).
DRIVE	The drive number (the number which appears on the "unit plug" on the front of the disk drive).
SERIAL-NUMBER	The decimal serial number of the disk drive.
SEEKS X1000	The decimal number of seeks performed by this drive during this pass of test 4. Multiply value by 1000.
MBYTES READ	The number of mega-bytes (million bytes) read by this drive during this pass of test 4. It is this value that is used to optionally drop a drive by the READ TRANSFER LIMIT software question.
MBYTES WRITTEN	The number of mega-bytes written by this drive during this pass of test 4.
HARD ERRORS	The number of hard error reports printed for this drive during this pass of test 4. It is this value that is used to optionally drop a drive by the ERROR LIMIT software question.

## SOFT ERRORS

The number of soft errors reported for the drive during this pass of test 4. A soft error is any error condition that resulted in a retry operation that eventually succeeded in recovering from the error condition. One soft error is counted even though several retry attempts may be made and does not correspond to the number of soft error reports printed. To see the soft error reports, you must change the default answer to the SUPPRESS PRINTING SOFT ERRORS software question.

## ECC

The number of times data read from the drive was modified using the error correction code (ECC) and resulted in a matching error detection code (EDC).

5.0 TEST SUMMARIES  
-----

The UDA Host Resident Diagnostic consists of one PDP-11 diagnostic supervisor program that runs in the PDP-11 processor and four programs that run in the UDA's buffer memory through an interpreter called the "diagnostic machine" which resides in the UDA. The PDP-11 program mainly is responsible for downline loading the "diagnostic machine" programs into the UDA and starting their execution. The "diagnostic machine" program controls the testing from that point by requesting the PDP-11 processor to supply information, print error messages and update statistics. The "diagnostic machine" program informs the PDP-11 processor when a test is complete.

Four "diagnostic machine" programs are in the ZUDCEO.PAK data file which is read from the XXDP+ system device by the PDP-11 program. The data file comes with listings of each program.

5.1 TEST # 1 - UNIBUS ADDRESSING TEST  
-----

The purpose of test #1 is to complete the testing of the Unibus interface in the UDA. The UDA resident diagnostic is not able to completely test the Unibus interface because communication with the PDP-11 processor is necessary. Specifically, this test will:

1. Check that every address line on the Unibus can be driven to both one and zero states.
2. Check that the UDA can interrupt the PDP-11 processor at the proper priority level and vector.
3. Exercise the Unibus interface by transferring blocks of data to and from Unibus memory.

This test assumes that the following are being tested by the UDA Resident Diagnostic:

1. All data bits can be written and read correctly.
2. NPR cycles can be executed correctly.

Test 1 is divided into six subtests. One at a time, each UDA selected for testing will run each subtest.

Subtest 1 makes sure that the UDAIP and UDASA registers are existant and runs the first part of the UDA's resident diagnostics.

Subtest 2 initializes the UDA into diagnostic loop mode. In this mode any value written into the UDASA is echoed in the UDASA.

In subtest 3, the UDA is initialized with interrupts enabled. The vector address and priority level will be determined solely from the answers to the hardware questions. If the hardware vectors to the wrong address, it is impossible to determine the result. A descriptive error message of the problem will not occur (the program or processor may hang or an unrelated message may occur). Therefore, the message "TESTING INTERRUPT ABILITY OF UDA AT ADR xxxxxx VEC xxx..." is printed just before the UDA is requested to cause an interrupt and the word "COMPLETED" is printed (on the same line) when the interrupt test is completed. If the word "COMPLETED" does not follow the first message, it should be apparent that the interrupt caused the diagnostic or processor to go astray. The priority level of the interrupt request is also verified.

Subtest 4 and 5 initialize the UDA using different sizes of the host communications area. The different sizes of the host communications area are supplied to allow the UDA Resident Diagnostic to do the most Unibus address testing possible. Interrupts are disabled. Any UDA Resident Diagnostic errors will be reported. Subtest 4 initializes the UDA with the smallest ring buffer size possible. Subtest 5 initializes the UDA with a large ring buffer area.

Subtest 6 downline loads a "diagnostic machine" program into the UDA. The "diagnostic machine" program is downline loaded from the memory space included in the host communications area when the UDA was first initialized. The UDA Resident Diagnostic has already verified that it can access these memory addresses, so the downline load command should perform properly. The "diagnostic machine" program is then started.

The "diagnostic machine" program asks the PDP-11 program to fill free memory (that memory available to the PDP-11 program that is not being used by the program or the Runtime Services) with an addressing pattern and report the location and size of the free memory. Every location of free memory is read and the data checked. Then, one by one, each address line is tested as follows:

1. Determine a test address by taking the first address of free memory and complementing the address bit to be tested.
2. Read from the test address.
3. If a non-existent memory error occurs, the test is complete.
4. Write all ones to the first address of free memory then read from the test address. If data read is not all ones, then test is complete.
5. Write zeros to the first address of free memory then read from the test address. If data read is not zeros, then test is complete.
6. Report Unibus addressing error.

When all address bits have been tested, then block transfers to and from memory are tested with different data patterns. This data is transferred at the rate disk data is transferred to and from memory during normal UDA operation.

The next UDA selected for testing is then be tested in the same manner. When all UDAs have been tested, test #1 ends.

## 5.2 TEST # 2 - DISK RESIDENT DIAGNOSTIC TEST

-----

The purpose of test #2 is to execute the diagnostics that run in each disk drive. These diagnostic programs may be resident in the disk drive or require downline loading from the ZUDCEO.PAK data file. (There currently are no disk drives that require downline loading and no such files exist in the ZUDCEO.PAK file. This program is designed such that they can be easily added in a future release.) This UDA diagnostic program only knows the procedure to execute the disk resident diagnostics and how to determine whether a test passed or failed.

One at a time, each UDA selected for testing is initialized and a "diagnostic machine" program downline loaded. The "diagnostic machine" program asks what drives are to be tested, then issues several commands to the disk drive and check for the correct response from the drive. This should serve as a good indicator that the UDA and disk drive can communicate.

A DIAGNOSE command is then issued to the drive to request the drive run all of its diagnostics. If the disk drive requests a downline load of a drive diagnostic, the diagnostic program is read from the XXDP+ load device, downline loaded into the disk drive and started. There is no limit to the number of downline loads that can be requested by a drive.

If the "Manual Intervention Mode" software question was answered "N" (default) testing proceeds to the next drive. When all drives on the UDA have been tested, the next UDA selected for testing is tested in the same manner. When all UDA's have been tested, test #2 ends.

If the "Manual Intervention Mode" software question was answered "Y", an interactive mode is entered to allow the operator to perform diagnostic activities on the disk drive as desired. The Service Manual for the disk drive must be used to determine what diagnostic capabilities are available.

First, a brief description of available commands is printed as follows:

TEST #2 MANUAL INTERVENTION ON UNIT xx UDA AT xxxxxx DRIVE xxx  
TO WRITE AND READ MEMORY:  
W DATA REGION OFFSET  
R REGION OFFSET  
TO RUN A DIAGNOSTIC:  
D REGION  
TO EXIT QUESTIONING:  
E  
DATA, REGION AND OFFSET ARE HEX VALUES.  
?

Commands may be typed after the question mark prompt. Each command is processed as entered and results displayed immediately. The exit command will allow the diagnostic to proceed.

Read and write commands remember the region and offset values. Successive read and successive write commands automatically increment to the next offset if the region and offset values are not typed. If a region is typed but not an offset, offset zero is used.

Examples:

```
1.      W FF FFFC 4
2.      W 02
3.      R FFFC 4
          FFFC 0004/ FF
4.      R
          FFFC 0005/ 02
5.      W 21 FFFC
6.      R
          FFFC 0000/ 21
```

Command 1 writes one byte (FF) into region FFFC, offset 4. Command 2 writes one byte (02) into the next byte - region FFFC, offset 0005. Commands 3 and 4 read the bytes back. Command 5 writes one byte (21) into the first byte of region FFFC. Command 6 reads back that byte.

The diagnose command remembers the region from previous diagnose commands only, because the region containing the diagnostic is generally not the same region used to write parameters or read results. If the diagnostic returns any data, the data is printed immediately.

5.3 TEST # 3 - DISK FUNCTION TEST  
-----

The purpose of test #3 is to functionally test the disk drive. On a drive that is well diagnosed by its disk resident diagnostics (executed by test #2) these functional tests will have little value. On a drive that has no or minimal resident diagnostics, these functional tests will have more value.

Test #3 starts by initializing each UDA selected for testing and then downline loading a "diagnostic machine" program into each UDA. Once all UDAs have been started, the PDP-11 program responds to requests from all UDAs. When all the UDAs have indicated the end of testing, test #3 ends.

The "diagnostic machine" program performs the following functions on each drive:

1. Issue a DRIVE CLEAR command.
2. Issue RECALIBRATE command.
3. Issue a CHANGE MODE command to enable diagnostic cylinder access, set the drive to 512 byte sector size, and write protect.
4. Issue INITIATE SEEK command to last diagnostic cylinder.
5. Read all factory formatted sector headers. If no headers on a track can be read, report the error, otherwise continue.
6. Starting with cylinder 0, group 0 and incrementing through every cylinder on the disk, seek to a group, read a header on track 0 and then seek to the factory formatted diagnostic cylinder. Read from the diagnostic cylinder to verify disk positioned correctly.
7. Attempt to write on the first diagnostic cylinder while write protected.
8. Issue a CHANGE MODE command to enable formatting operations and disable write protect.
9. Format all writable DBNs in 512 byte format.
10. Write and read several data patterns to each writable DBN. Report an error if all DBNs on one track have an error.
11. Send invalid SDI level 2 and level 1 commands and check the results.
12. Go to the XBN area and read a copy of the FCT. Check to see if the drive has been properly formatted in 512 byte mode.
13. Issue a DISCONNECT command.

5.4 TEST # 4 - DISK EXERCISER  
-----

The purpose of test #4 is to exercise the disk drives in a manner similar to normal usage under standard operating systems. Execution of this test should give an indication of the performance of the disk drive. This test may be run for long or short periods of time, depending on how the software questions are answered.

These are two modes of operation for test #4:

1. Default operation on the entire area selected (customer or diagnostic) with all parameters selected for random operation as shown by default answers below.
2. Manual intervention mode where a number of questions are asked and operation is controlled by their answers.

Which mode is entirely determined by the answer to the first software question asking, "Enter manual intervention mode for special diagnosis?" This question would normally have been answered "N" (default) and testing will begin immediately. If answered "Y", the following series of questions will be asked for each unit selected for testing:

THE FOLLOWING QUESTIONS REFER TO UNIT xx UDA AT xxxxxx DRIVE xxx

This message will identify to which drive the questions are being asked. The entire series of questions will be asked for each drive, there is no short way to answer like in the hardware questions.

NUMBER OF BAD BLOCKS (D) 0 ?

An answer in the range of 1 to 16 will allow that many bad block numbers to be entered. The program will allow writes and reads to these blocks but no error messages will be printed for these blocks. Errors encountered on these blocks will not appear in the statistics. Answer zero to bypass entering bad blocks.

BAD BLOCK (A) ?

This question will be asked the number of times requested by the previous answer. Any decimal number that can be converted into a 28-bit binary value will be accepted. No other error checking will be made at this time to determine if the block number actually exists on the disk.

DO YOU WANT TO CHANGE TESTING PARAMETERS FOR THIS DRIVE (L) N ?

Answer "N" to bypass all further questioning on this drive.  
Answer "Y" to be asked the following questions.

READ ONLY (L) N ?

Answer "Y" to dictate read only and prevent test 04 from performing any writes to the disk.

WRITE ONLY (L) N ?

This question will only be asked if the previous question was answered "N". Answer "Y" to dictate write only.

CHECK ALL WRITES BY READING (L) N ?

Answer "Y" to cause all writes to be checked by reading the data immediately after the write operation.

RANDOMLY CHECK WRITES BY READING (L) Y ?

This question will only be asked if the previous question was answered "N". Answer "Y" for the write check to be performed randomly. Answer "N" if write checks are not desired. This question is asked no matter how previous questions were asked.

DATA PATTERN - 0 FOR RANDOM SELECTION (D) 0 ?

There are 16 data patterns available, selected as 1 to 16. Pattern number 0 will cause patterns 1 to 15 to be randomly selected for each write. If pattern number 16 is selected, the following set of questions will be asked for a pattern to be input.

ENABLE ECC DATA CORRECTION (L) Y ?

A "Y" answer will enable the use of ECC to correct data errors. If the number of corrections is within the drive's threshold, an informational message will be printed identifying the block number. These ECC corrections will also appear in the statistical report for the drive.

An "N" answer will prevent the use of ECC. All ECC errors will cause an error message to be printed and retries to be attempted.

COMPARE ALL DATA READ (L) N ?

Answer "Y" to cause a data compare after every read.

RANDOMLY COMPARE DATA READ (L) Y ?

This question will only be asked if the previous question was answered "N". Answer "Y" for the data compare to be performed on random records. Answer "N" if data compares are not desired.

09

USER DOCUMENTATION

ENABLE RETRIES (L) Y

A "Y" answer will enable retries to be performed on disk errors.

RANDOM ACCESS MODE (L) Y ?

Answer "Y" to cause block numbers to be chosen randomly.  
Answer "N" to cause block numbers to be selected sequentially up and down the disk surface.

DO YOU WISH TO:

- 0 - TEST ENTIRE AREA SELECTED
- 1 - SPECIFY BEGIN/END SETS TO TEST
- 2 - SPECIFY TRACKS AND CYLINDERS TO TEST
- 3 - SPECIFY GROUPS AND CYLINDERS TO TEST
- 4 - SPECIFY CYLINDERS TO TEST

(D) 0 ?

This question specifies the options available to limit testing to a portion of the selected area (customer or diagnostic) of the disk. A zero answer is the default which specifies to use the entire area for the test. Other answers will cause additional questions to be asked.

NUMBER OF BEGIN/END SETS (D) 1 ?

BEGIN BLOCK (A) 0 ?

END BLOCK (A) 0 ?

These questions are asked if begin/end sets were selected to limit the testing area (Answer 1). One to four sets may be specified. The BEGIN BLOCK and END BLOCK questions are asked as many times as needed.

NUMBER OF TRACKS TO TEST (D) 1 ?

TRACK (D) 0 ?

NUMBER OF GROUPS TO TEST (D) 1 ?

GROUP (D) 0 ?

One of these sets of questions is asked if either tracks and cylinders or groups and cylinders was specified to limit the testing area (Answers 2 or 3). Up to seven tracks or groups may be specified on which testing will be limited.

DO YOU WISH TO LIMIT THE CYLINDERS TESTED (L) N ?

This question is asked only after the tracks or groups have been specified above. If testing is to be further limited to a set of cylinders, answer "Y" and the following two questions will be asked:

STARTING CYLINDER (A) 0 ?  
 ENDING CYLINDER (A) 0 ?

These questions are asked if the question immediately above was answered "Y" or if cylinders were selected to limit the testing area (Answer 4). One set of cylinder numbers may be specified to limit the testing area.

After the above questions have been asked for all drives selected for testing, the following questions will be asked if data pattern 16 was selected for any drive:

NUMBER OF WORDS IN DATA PATTERN 16 (D) 1 ?  
 DATA WORD (O) 0 ?

Data pattern 16 can be input by these questions. A data pattern consists of a buffer of one to 16 words which is repeated throughout the data portion of the disk block. Enter the contents of the data pattern buffer. The DATA WORD question will be repeated as needed.

Test #4 will then initialize each UDA selected for testing and downline load a "diagnostic machine" program into each UDA. Because the "diagnostic machine" programs are too large to fit both copies in memory at the same time (as done in Tests 1 through 3), the program checks which type of UDA-50s are being tested. If all are of the same type, that program is read. If both types are selected for testing, the program for the UDA-50 with the M7485 and M7486 boards is read.

The "diagnostic machine" program asks what drives are to be tested and then for the parameters for each drive (the answers to the manual intervention questions or their defaults). Once all UDAs have been started, the PDP-11 program responds to requests from all UDAs.

The disks are then exercised according to the parameters. The exercise consists of selecting a disk sector, seeking to the proper cylinder, then reading or writing the sector. The parameters control how the disk sector is selected, whether the sector is written or read and whether a write is followed by a read (write check).

The "diagnostic machine" program periodically sends statistics to the PDP-11 program. These statistics include counts of reads, writes, seeks and errors on a per drive basis. The PDP-11 program accumulates the statistics from all the UDAs and watches for the transfer limit to be exceeded. As long as the error log is not enabled, the exceeding of the transfer limit will cause the end of test #4.

Each time an error occurs, the "diagnostic machine" tells the PDP-11 program. A message is printed (or stored in the log buffer) and then the error limit for the drive is checked. If the error limit has been reached, the drive is dropped from testing. If no more drives remain to be tested, test #4 will end (unless the error log is enabled).

When the end of test #4 occurs, the accumulated statistics for each drive is printed. This statistical report can be printed at any time during test #4 by typing control-C then the PRINT command.

The data patterns used by test #4 are indicated below. Each pattern is generated by writing the pattern number in each 4-bit nibble of the first word, then repeating the data pattern (sequence of one to 16 words) throughout the rest of the data buffer. Pattern number 16 writes nibbles of zeros. When pattern number zero is used, the actual pattern number written (1 to 15) is placed in the nibbles.

PATTERN 0 This pattern number is used to indicate any pattern number 1 to 15 chosen at random.

PATTERN 1 Words in pattern sequence - 1  
Sequence (Octal) 105613  
Sequence (Hex) 8B8B

PATTERN 2 Words in pattern sequence - 1  
Sequence (Octal) 031463  
Sequence (Hex) 3333

PATTERN 3 Words in pattern sequence - 1  
Sequence (Octal) 030221  
Sequence (Hex) 3091

PATTERN 4 Words in pattern sequence - 16 (Shifting ones)  
Sequence (Octal) 000001, 000003, 000007, 000017, 000037,  
000077, 000177, 000377, 000777, 001777,  
003777, 007777, 017777, 037777, 077777,  
177777  
Sequence (Hex) 0001, 0003, 0007, 000F, 001F, 003F,  
007F, 00FF, 01FF, 03FF, 07FF, 0FFF,  
1FFF, 3FFF, 7FFF, FFFF

PATTERN 5 Words in pattern sequence - 16 (Shifting zeros)  
 Sequence (Octal) 177776, 177774, 177770, 177760, 177740,  
 177700, 177600, 177400, 177000, 176000,  
 174000, 170000, 160000, 140000, 100000,  
 000000  
 Sequence (Hex) FFFE, FFFC, FFF8, FFF0, FFEO, FFCO,  
 FF80, FF00, FE00, FC00, F800, F000,  
 E000, C000, 8000, 0000

PATTERN 6 Words in pattern sequence - 16  
 Sequence (Octal) 000000, 000000, 000000, 177777, 177777,  
 177777, 000000, 000000, 177777, 177777,  
 000000, 177777, 000000, 177777, 000000,  
 177777  
 Sequence (Hex) 0000, 0000, 0000, FFFF, FFFF, FFFF,  
 0000, 0000, FFFF, FFFF, 0000, FFFF,  
 0000, FFFF, 0000, FFFF

PATTERN 7 Words in pattern sequence - (BINARY 1011011011011001)  
 Sequence (Octal) 133331  
 Sequence (Hex) 86D9

PATTERN 8 Words in pattern sequence - 16  
 Sequence (Octal) 052525, 052525, 052525, 125252, 125252,  
 125252, 052525, 052525, 125252, 125252,  
 052525, 125252, 052525, 125252, 052525,  
 125252  
 Sequence (Hex) 5555, 5555, 5555, AAAA, AAAA, AAAA,  
 5555, 5555, AAAA, AAAA, 5555, AAAA,  
 5555, AAAA, 5555, AAAA

PATTERN 9 Words in pattern sequence - 1 (BINARY 1101101101101100)  
 Sequence (Octal) 155554  
 Sequence (Hex) DB6C

PATTERN 10 Words in pattern sequence - 16  
 Sequence (Octal) 026455, 026455, 026455, 151322, 151322,  
 151322, 026455, 026455, 151322, 151322,  
 026455, 151322, 026455, 151322, 026455,  
 151322  
 Sequence (Hex) 2020, 2020, 2020, D202, D202, D202,  
 2020, 2020, D202, D202, 2020, D202,  
 2020, D202, 2020, D202

- PATTERN 11 Words in pattern sequence - 1 (BINARY 0110110110110110)  
Sequence (Octal) 066666  
Sequence (Hex) 6DD6
- PATTERN 12 Words in pattern sequence - 16 (Ripple one)  
Sequence (Octal) 000001, 000002, 000004, 000010, 000020,  
000040, 000100, 000200, 000400, 001000,  
002000, 004000, 010000, 020000, 040000,  
100000  
Sequence (Hex) 0001, 0002, 0004, 0008, 0010, 0020,  
0040, 0080, 0100, 0200, 0400, 0800,  
1000, 2000, 4000, 8000
- PATTERN 13 Words in pattern sequence - 16 (Ripple zero)  
Sequence (Octal) 177776, 177775, 177773, 177767, 177757,  
177737, 177677, 177577, 177377, 176777,  
175777, 173777, 167777, 157777, 137777,  
077777  
Sequence (Hex) FFFE, FFFD, FFFB, FFF7, FFEF, FFDF,  
FFBF, FF7F, FEFF, FDFF, FBFF, F7FF,  
EFFF, DFFF, BFFF, 7FFF
- PATTERN 14 Words in pattern sequence - 3  
Sequence (Octal) 155555, 133333, 155555  
Sequence (Hex) DB6D, B6DB, DB6D
- PATTERN 15 Words in pattern sequence - 16  
Sequence (Octal) 133331, 133331, 133331, 155554, 155554,  
155554, 133331, 133331, 155554, 155554,  
133331, 155554, 133331, 155554, 133331,  
155554  
Sequence (Hex) B6D9, B6D9, B6D9, DB6C, DB6C, DB6C,  
B6D9, B6D9, DB6C, DB6C, B6D9, DB6C,  
B6D9, DB6C, B6D9, DB6C
- PATTERN 16 This is the operator selectable pattern in manual  
intervention mode. Questions are asked when test #4 is  
started for the operator to input the number of words in  
the sequence and the contents of the words.

Sample of terminal dialogue going through manual intervention questions:

DR>STA/TEST:4

CHANGE HW (L) ? N

CHANGE SW (L) ? Y

ENTER MANUAL INTERVENTION MODE FOR SPECIAL DIAGNOSIS (L) N ? Y

REMAINING SOFTWARE QUESTIONS APPLY TO TEST 4 ONLY

ERROR LIMIT (D) 32 ?

READ TRANSFER LIMIT IN MEGABYTES - 0 FOR NO LIMIT (D) 0 ?

SUPPRESS PRINTING SOFT ERRORS (L) Y ? N

DO INITIAL WRITE ON START (L) Y ?

ENABLE ERROR LOG (L) N ?

THE FOLLOWING QUESTIONS REFER TO UNIT 0 UDA AT 172150 DRIVE 0

NUMBER OF BAD BLOCKS (D) 0 ? 2

BAD BLOCK (A) ? 234

BAD BLOCK (A) ? 8900

DO YOU WANT TO CHANGE TESTING PARAMETERS FOR THIS DRIVE (L) N ? Y

READ ONLY (L) N ?

WRITE ONLY (L) N ?

CHECK ALL WRITES BY READING (L) N ? Y

DATA PATTERN - 0 FOR RANDOM SELECTION (D) 0 ? 1

ENABLE ECC DATA CORRECTION (L) Y ?

COMPARE ALL DATA READ (L) N ? Y

ENABLE RETRIES (L) Y ?

RANDOM ACCESS MODE (L) Y ? N

DO YOU WISH TO:

0 - TEST ENTIRE AREA SELECTED

1 - SPECIFY BEGIN/END SETS TO TEST

2 - SPECIFY TRACKS AND CYLINDERS TO TEST

3 - SPECIFY GROUPS AND CYLINDERS TO TEST

4 - SPECIFY CYLINDERS TO TEST

(D) 0 ? 1

NUMBER OF BEGIN/END SETS (D) 1 ?

BEGIN BLOCK (A) 0 ?

END BLOCK (A) 0 ? 200

NUMBER OF WORDS IN DATA PATTERN 16 (D) 1 ?

DATA WORD (O) 0 ?

Ⓜ

1  
2  
358  
367  
368  
394  
396 000000  
397  
398 002000  
400  
402  
403  
404  
405  
406  
408  
425  
427 002000  
002000 103  
002001 132  
002002 125  
002003 104  
002004 103  
002005 000  
002006 000  
002007 000  
002010  
002010 105  
002011  
002011 060  
002012  
002012 000001  
002014  
002014 000000  
002016  
002016 113266  
002020  
002020 113526  
002022  
002022 064356  
002024  
002024 064374  
002026  
002026 114312  
002030  
002030 000000  
002032  
002032 000000  
002034  
002034 000001  
002036  
002036 000000  
002040  
002040 064344  
002042  
002042 000340  
002044

```

; *LAST REVISION 04-OCT-83
.TITLE CZUDCEO UDA & DISK DRV DIAG
.SBTTL PROGRAM HEADER
.ASECT
.ENABL AMA 2000
;
; **
; THE PROGRAM HEADER IS THE INTERFACE BETWEEN
; THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
; **

L$NAME:: ;DIAGNOSTIC NAME
.ASCLL /C/
.ASCLL /Z/
.ASCLL /U/
.ASCLL /D/
.ASCLL /C/
.BYTE 0
.BYTE 0
.BYTE 0

L$REV:: ;REVISION LEVEL
.ASCLL /E/

L$DEPO:: ;0
.ASCLL /0/

L$UNIT:: ;NUMBER OF UNITS
.WORD T$PTHV

L$TIML:: ;LONGEST TEST TIME
.WORD 0

L$HPCP:: ;POINTER TO H.W. QUES.
.WORD L$HARD

L$SPCP:: ;POINTER TO S.W. QUES.
.WORD L$SOFT

L$HPTP:: ;PTR. TO DEF. H.W. PTABLE
.WORD L$HW

L$SPTP:: ;PTR. TO S.W. PTABLE
.WORD L$SW

L$LADP:: ;DIAG. END ADDRESS
.WORD L$LAST

L$STA:: ;RESERVED FOR APT STATS
.WORD 0

L$CO::
.WORD 0

L$DTYF:: ;DIAGNOSTIC TYPE
.WORD 1

L$APT:: ;APT EXPANSION
.WORD 0

L$DTP:: ;PTR. TO DISPATCH TABLE
.WORD L$DISPATCH

L$PRIO:: ;DIAGNOSTIC RUN PRIORITY
.WORD PRIO?

L$ENVI:: ;FLAGS DESCRIBE HOW IT WAS SETUP

```

002044	000000		.WORD	0	
002046		L\$EXP1::	.WORD	0	;EXPANSION WORD
002046	000000		.WORD	0	
002050		L\$MREV::	.BYTE	C\$REVISION	;SVC REV AND EDIT #
002050	003		.BYTE	C\$EDIT	
002051	003		.BYTE	C\$EDIT	
002052		L\$EF::	.WORD	0	;DIAG. EVENT FLAGS
002052	000000		.WORD	0	
002054	000000		.WORD	0	
002056		L\$SPC::	.WORD	0	
002056	000000		.WORD	0	
002060		L\$DEVP::	.WORD	L\$DVTYP	; POINTER TO DEVICE TYPE LIST
002060	064700		.WORD	L\$DVTYP	
002062		L\$REPP::	.WORD	L\$RPT	;PTR. TO REPORT CODE
002062	106654		.WORD	L\$RPT	
002064		L\$EXP4::	.WORD	0	
002064	000000		.WORD	0	
002066		L\$EXP5::	.WORD	0	
002066	000000		.WORD	0	
002070		L\$AUT::	.WORD	0	;PTR. TO ADD UNIT CODE
002070	000000		.WORD	0	
002072		L\$DUT::	.WORD	0	;PTR. TO DROP UNIT CODE
002072	000000		.WORD	0	
002074		L\$LUN::	.WORD	0	;LUN FOR EXERCISERS TO FILL.
002074	000000		.WORD	0	
002076		L\$DESP::	.WORD	L\$DESC	;POINTER TO DIAG. DESCRIPTION
002076	064724		.WORD	L\$DESC	
002100		L\$LOAD::	EMT	E\$LOAD	;GENERATE SPECIAL AUTOLOAD EMT
002100	104035		EMT	E\$LOAD	
002102		L\$ETP::	.WORD	L\$ERRTBL	;POINTER TO ERRTBL
002102	064402		.WORD	L\$ERRTBL	
002104		L\$ICP::	.WORD	L\$INIT	;PTR. TO INIT CODE
002104	107636		.WORD	L\$INIT	
002106		L\$CCP::	.WORD	L\$CLEAN	;PTR. TO CLEAN-UP CODE
002106	111302		.WORD	L\$CLEAN	
002110		L\$ACP::	.WORD	L\$AUTO	;PTR. TO AUTO CODE
002110	111300		.WORD	L\$AUTO	
002112		L\$PRT::	.WORD	L\$PROT	;PTR. TO PROTECT TABLE
002112	107630		.WORD	L\$PROT	
002114		L\$TEST::	.WORD	0	;TEST NUMBER
002114	000000		.WORD	0	
002116		L\$DLY::	.WORD	0	;DELAY COUNT
002116	000000		.WORD	0	
002120		L\$HIME::	.WORD	0	;PTR. TO HIGH MEM
002120	000000		.WORD	0	

L9

```
1  
2 ;THIS LOCATION MUST BE AT THIS POSITION. SEPERATE CODE, STORED IN  
3 ;THE PAK FILE, WAS ASSEMBLED WITH THIS ADDRESS  
4  
5 062220 STOSIZ = 25000, - 236. ;STORAGE SIZE  
6  
8 002122 STORAG: .BLKB STOSIZ
```

DISPATCH TABLE

```
1          .SBTTL DISPATCH TABLE
2
3          ;++
4          ; THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
5          ; IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
6          ;--
7
8 064342    000004          .WORD    4
064344          L$DISPATCH:
064344    111344          .WORD    T1
064346    112420          .WORD    T2
064350    112516          .WORD    T3
064352    112554          .WORD    T4
11.
```

```

1          .SBTTL  DEFAULT HARDWARE P-TABLE
2
3          ;++
4          ; THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
5          ; THE TEST-DEVICE PARAMETERS  THE STRUCTURE OF THIS TABLE
6          ; IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
7          ; AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
8          ;--
9
10 064354 000006          .WORD  L10000-L$HW/2
    064356
    064356          L$HW::
11 064356 172150          DFPTBL::          .WORD  172150          ; UNIBUS ADDRESS
12 064360 000154          .WORD  154          ; VECTOR ADDRESS
13 064362 000005          .WORD  5.          ; BR LEVEL
14 064364 000077          .WORD  {3.          ; UNIBUS BURST RATE
15 064366 000000          .WORD  0.          ; LOGICAL DRIVE NUMBER
16 064370 000000          .WORD  0.          ; CUSTOMER DATA AREA
17
27 064372          L10000:

```

```

1          .SBTTL  SOFTWARE P-TABLE
2
3
4          ;***
5          ; THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
6          ; PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
7          ; SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
8          ; AT RUN TIME.
9          ;**
10         064372 000003          .WORD  L10001-L1SW/2
11         064374          ;L1SW::
12         064374 000040          ;SEPTBL::
13         064376 000000          .WORD  32.          ;ERROR LIMIT
14         064400 040400          .WORD  0.          ;DATA TRANSFER LIMIT (MEGABITS)
15         21          .WORD  +901000001000000000 ;SINGLE BIT QUESTIONS
16         29
17         30 064402          L10001:

```

12  
40  
50  
52  
53  
54  
55  
56  
57

.SBTTL GLOBAL EQUATES SECTION

\*\*\*  
; THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT  
; ARE USED IN MORE THAN ONE TEST.  
\*\*\*

; BIT DIFINITIONS

100000	BIT15** 100000
040000	BIT14** 40000
020000	BIT13** 20000
010000	BIT12** 10000
004000	BIT11** 4000
002000	BIT10** 2000
001000	BIT09** 1000
000400	BIT08** 400
000200	BIT07** 200
000100	BIT06** 100
000040	BIT05** 40
000020	BIT04** 20
000010	BIT03** 10
000004	BIT02** 4
000002	BIT01** 2
000001	BIT00** 1

; BIT9\*\* BIT09  
; BIT8\*\* BIT08  
; BIT7\*\* BIT07  
; BIT6\*\* BIT06  
; BIT5\*\* BIT05  
; BIT4\*\* BIT04  
; BIT3\*\* BIT03  
; BIT2\*\* BIT02  
; BIT1\*\* BIT01  
; BIT0\*\* BIT00

; EVENT FLAG DEFINITIONS  
; EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

000040	EF.START** 32.	; START COMMAND WAS ISSUED
000037	EF.RESTART** 31.	; RESTART COMMAND WAS ISSUED
000036	EF.CONTINUE** 30.	; CONTINUE COMMAND WAS ISSUED
000035	EF.NEW** 29.	; A NEW PASS HAS BEEN STARTED
000034	EF.PWR** 28.	; A POWER-FAIL/POWER-UP OCCURRED

; PRIORITY LEVEL DEFINITIONS

000340	PRI07** 340
000300	PRI06** 300
000240	PRI05** 240
000200	PRI04** 200
000140	PRI03** 140
000100	PRI02** 100

D10

000040	PRIC1** 40
000000	PRIO0** 0
	;
	OPERATOR FLAG BITS
	;
000004	EVL** 4
000010	LOT** 10
000020	ADR** 20
000040	IDU** 40
000100	ISR** 100
000200	UAM** 200
000400	BOE** 400
001000	PNT** 1000
002000	PRI** 2000
004000	IXF** 4000
010000	IBE** 10000
020000	IER** 20000
040000	LOE** 40000
100000	HOE** 100000

```

1          .SBTTL  UDA BIT DEFINITIONS
2
3
4          ;UDASA REGISTER UNIVERSAL READ BITS
5
6          100000  SA.ERR  = 100000      ;ERROR INDICATOR
7          040000  SA.S4   = 040000      ;STEP 4 STATUS BIT
8          020000  SA.S3   = 020000      ;STEP 3 STATUS BIT
9          010000  SA.S2   = 010000      ;STEP 2 STATUS BIT
10         004000  SA.S1   = 004000      ;STEP 1 STATUS BIT
11
12
13         ;UDASA REGISTER ERROR STATUS BITS
14
15         003777  SA.ERC  = 003777      ;ERROR CODE
16
17
18         ;UDASA REGISTER STEP 1 SEND BITS
19
20         000177  SA.VEC  = 000177      ; INTERRUPT VECTOR (DIVIDED BY 4)
21         000200  SA.INT  = 000200      ; INTERRUPT ENABLE DURING INITIALIZATION
22         003400  SA.MSG  = 003400      ; MESSAGE RING LENGTH
23         034000  SA.CMD  = 034000      ; COMMAND RING LENGTH
24         040000  SA.WRP  = 040000      ; WRAP BIT
25         100000  SA.STP  = 100000      ; STEP - MUST ALWAYS BE WRITTEN A ONE
26
27         000400  SA.MS1  = 000400      ; LSB OF MESSAGE RING LENGTH
28         004000  SA.CM1  = 004000      ; LSB OF COMMAND RING LENGTH
29
30
31         ;UDASA REGISTER STEP 1 RESPONSE BITS
32
33         002000  SA.NV   = 002000      ;NON SETTABLE INTERRUPT VECTOR.
34         001000  SA.A2   = 001000      ;22 BIT ADDRESS BUS
35         000400  SA.DI   = 000400      ;ENHANCED DIAGNOSTICS
36         ;          000377      ;ALL BITS RESERVED
37
38
39         ;UDASA REGISTER STEP 2 SEND BITS
40
41         000001  SA.PRG  = 000001      ;ENABLE VAX UNIBUS ADAPTER PURGE INTERRUPT
42         ;          177776      ;LOW ORDER MESSAGE RING BYTE ADDRESS
43
44
45         ;UDASA REGISTER STEP 2 RESPONSE BITS
46
47         000007  SA.MSE  = 000007      ;MESSAGE RING LENGTH ECHO
48         000070  SA.CME  = 000070      ;COMMAND RING LENGTH ECHO
49         ;          000100      ;RESERVED
50         000200  SA.STE  = 000200      ;STEP ECHO
51         003400  SA.CTP  = 003400      ;CONTROLLER TYPE
52
53
54         ;UDASA REGISTER STEP 3 SEND BITS
55
56         ;          077777      ;HIGH ORDER MESSAGE RING BYTE ADDRESS
57         100000  SA.TST  = 100000      ;PURGE POLE TEST ENABLE

```

```

58
59
60      ;UDASA REGISTER STEP 3 RESPONSE BITS
61
62      000177      SA.VCE   = 000177      ; INTERRUPT VECTOR ECHO
63      000200      SA.INE   = 000200      ; INTERRUPT ENABLE ECHO
64      000400      SA.NVE   = 000400      ; VECTOR NOT PROGRAMMABLE
65      ;           003000      ; RESERVED
66
67
68      ;UDASA REGISTER STEP 4 SEND BITS
69
70      000001      SA.GO    = 000001      ; GO BIT TO START UDA FIRMWARE
71      000002      SA.LFC   = 000002      ; LAST FAILURE CODE REQUEST
72      000374      SA.BST   = 000374      ; BURST LEVEL
73
74
75      ;UDASA REGISTER STEP 4 RESPONSE BITS
76
77      000017      SA.MCV   = 000017      ; UDA MICROCODE VERSION
78      000360      SA.CNT   = 000360      ; CONTROLLER TYPE
79      ;           003400      ; RESERVED

```

```

1          .SBTTL  HOST COMMUNICATION AREA DEFINIIONS
2
3          .COMMAND/MESSAGE RING BIT DEFINITIONS
4
5          100000      RG.OWN  = 100000      ;SET WHEN UDA OWNS RING
6          040000      RG.FLG  = 040000      ;FLAG BIT
7
8
9          ;VIRTUAL CIRCUIT IDENTIFIERS
10
11         000000      MSCP    = 0           ;MSCP CIRCUIT
12         000001      LOG     = 1           ;LOG CIRCUIT
13         177777      DIAG    = -1         ;DIAGNOSTIC CIRCUIT
14         001000      DUP     = 1000        ;DIAGNOSTIC AND UTILITIES PROTOCOL
15
16
17         ;OFFSETS INTO HOST COMMUNICATIONS AREA WITH ONE DESCRIPTOR TO EACH RING
18         ;AND TWO PACKET
19
20         000004      HC.ISZ  = 4.         ;SIZE OF INTERRUPT INDICATOR WORDS
21         000004      HC.RSZ  = 4.         ;SIZE OF RING IN BYTES
22         000004      HC.ESZ  = 4.         ;SIZE OF ENVELOPE WORDS BEFORE PACKET
23         000060      HC.PSZ  = 48.        ;SIZE OF COMMAND AND MESSAGE PACKETS
24         000106      HC.BSZ  = 70.        ;SIZE OF BUFFER
25
26         000000      HC.INT  = 0.         ;INTERRUPT INDICATOR WORDS START
27
28         000004      HC.MSG  = HC.INT+HC.ISZ ;MESSAGE RING START
29         000006      HC.MCT  = HC.MSG+2.   ;MESSAGE RING CONTROL WORD
30
31         000010      HC.CMD  = HC.MSG+HC.RSZ ;COMMAND RING START
32         000012      HC.CCT  = HC.CMD+2.   ;COMMAND RING CONTROL WORDS
33
34         000014      HC.MEV  = HC.CMD+HC.RSZ ;MESSAGE ENVELOPE START
35         000020      HC.MPK  = HC.MEV+HC.ESZ ;MESSAGE PACKET START
36
37         000014      HC.CEV  = HC.MEV      ;COMMAND ENVELOPE START
38         000020      HC.CPK  = HC.MPK      ;COMMAND PACKET START
39
40
41
42
43
44         000100      HC.BF1  = HC.CPK+HC.PSZ ;FIRST BUFFER
45         000206      HC.BF2  = HC.BF1+HC.BSZ ;SECOND BUFFER
46
47         000314      HC.SIZ  = HC.BF2+HC.BSZ ;TOTAL SIZE OF HOST COMMUNICATION AREA

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32

.SBTTL HOST COMMUNICATION AREA LAYOUT

HC.INT	INTERRUPT INDICATORS	4 BYTES
HC.MSG HC.MCT	MESSAGE (RESPONSE) RING	4 BYTES
HC.CMD HC.CCT	COMMAND RING	4 BYTES
HC.MEV & HC.CEV	MESSAGE & COMMAND ENVELOPE	4 BYTES
HC.MPK & HC.CPK	MESSAGE & COMMAND PACKET	48 BYTES
HC.BF1	BUFFER # 1 (RESPONSE TO DM PROGRAM)	70 BYTES
HC.BF2	BUFFER # 2 (REQUEST FROM DM PROGRAM)	70 BYTES

NOTE: BYTES ARE GIVEN IN DECIMAL

```

1      .SBTTL  COMMAND PACKET OPCODES DEFINITIONS
2
3      000001  OP.ABO  = 1          ;ABORT COMMAND
4      000020  OP.ACC  = 20         ;ACCESS COMMAND
5      000010  OP.AVL  = 10         ;AVAILABLE COMMAND
6      000021  OP.CCD  = 21         ;COMPARE CONTROLLER DATA COMMAND
7      000040  OP.CMP  = 40         ;COMPARE HOST DATA COMMAND
8      000022  OP.ERS  = 22         ;ERASE COMMAND
9      000023  OP.FLU  = 23         ;FLUSH COMMAND
10     000002  OP.GCS  = 2          ;GET COMMAND STATUS COMMAND
11     000003  OP.GUS  = 3          ;GET UNIT STATUS COMMAND
12     000011  OP.ONL  = 11         ;ONLINE COMMAND
13     000041  OP.RD   = 41         ;READ COMMAND
14     000024  OP.RPL  = 24         ;REPLACE COMMAND
15     000004  OP.SCC  = 4          ;SET CONTROLLER CHARACTERISTICS COMMAND
16     000012  OP.SUC  = 12        ;SET UNIT CHARACTERISTICS COMMAND
17     000042  OP.WR   = 42         ;WRITE COMMAND
18     000030  OP.MRD  = 30        ;MAINTENANCE READ COMMAND
19     000031  OP.MWR  = 31        ;MAINTENANCE WRITE COMMAND
20     000200  OP.END  = 200       ;END PACKET FLAG
21     000007  OP.SEX  = 7          ;SERIOUS EXCEPTION END PACKET
22     000100  OP.AVA  = 100       ;AVAILABLE ATTENTION MESSAGE
23     000101  OP.DUP  = 101      ;DUPLICATE UNIT NUMBER ATTENTION MESSAGE
24     000102  OP.SHC  = 102      ;SHADOW COPY COMPLETE ATTENTION MESSAGE
25     000103  OP.RLC  = 103      ;RESET COMMAND LIMIT ATTENTION MESSAGE
26
27     000001  OP.GSS  = 1          ;DUP GET DUST STATUS
28     000002  OP.ESP  = 2          ;DUP EXECUTE SUPPLIED PROGRAM
29     000003  OP.ELP  = 3          ;DUP EXECUTE LOCAL PROGRAM
30     000004  OP.SSD  = 4          ;DUP SEND DUST DATA
31     000005  OP.RSD  = 5          ;DUP RECEIVE DUST DATA
32
33     ;NOTE: END PACKET OPCODES (ALSO CALLED ENDCODES) ARE FORMED BY ADDING THE END
34     ;PACKET FLAG TO THE COMMAND OPCODE. FOR EXAMPLE, A READ COMMAND'S END PACKET
35     ;CONTAINS THE VALUE OP.RD+OP.END IN ITS OPCODE FIELD. THE INVALID COMMAND END
36     ;PACKET CONTAINS JUST THE END PACKET FLAG (I.E., OP.END) IN ITS OPCODE FIELD.
37     ;THE SERIOUS EXCEPTION END PACKET CONTAINS THE SUM OF THE END PACKET FLAG
38     ;PLUS THE SERIOUS EXCEPTION OPCODE SHOWN ABOVE (I.E., OP.SEX+OP.END) IN ITS
39     ;OPCODE FIELD.
40
41     ;
42     ;COMMAND OPCODE BITS 3 THROUGH 5 INDICATE THE COMMAND CLASS, WHICH IS ENCODED
43     ;AS FOLLOWS:
44     ; 000 IMMEDIATE COMMANDS
45     ; 001 SEQUENTIAL COMMANDS
46     ; 010 NON-SEQUENTIAL COMMANDS THAT DO NOT INCLUDE A BUFFER DESCRIPTOR
47     ; 100 NON-SEQUENTIAL COMMANDS THAT DO INCLUDE A BUFFER DESCRIPTOR

```

DRIVE NOT FORMATTED IN 512 BYTE MODE -- UNABLE TO TEST  
FCT BLOCK ZERO MODE WORD: mode

\*\*\* THIS PACK HAS AN INVALID FORMAT AND CANNOT BE USED \*\*\*

mode:                   The mode word found on the drive's FCT block zero.

Error 4021 occurs only when Test 4 Finds that the mode word found in FCT block zero is not the 512 byte mode word (126736 octal). See DEC STANDARD 166 "FCT Structure". Inhibiting the dropping of units has no effect on this error.

04022 CZUDC DEV FTL ERR 04022 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
COULD NOT READ FCT BLOCK ZERO

\*\*\* THIS PACK HAS AN INVALID FORMAT AND CANNOT BE USED \*\*\*

Error 4022 occurs when test 4 is unable to read any copy of FCT block zero. See DEC STANDARD 166 "FCT Structure". Inhibiting the dropping of units has no effect on this error.

04023 CZUDC DEV FTL ERR 04023 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
UNABLE TO CONTINUE TESTING  
PORT SWITCH OUT  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

If, during testing, the operator disables the port that Test 4 is using by popping out the port switch, Test 4 prints error 4023. CHANGING THE STATE OF THE PORT SWITCH FOR THE PORT THAT Test 4 IS <<NOT>> USING HAS NO EFFECT ON THE TEST. If dropping of units is inhibited, Test 4 will loop in error recovery, printing this error, until the error state is corrected (by some external action).

CZUDC DEV FTL ERR 04023 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx  
DISK EXERCISER DM PC:xxxx UDA AT xxxxxx DRIVE xxx RUNTIME hh:mm:ss  
UNABLE TO CONTINUE TESTING  
RUN/STOP SWITCH OUT  
REAL TIME STATE 0003  
STATUS (R TO L): 1312 1110 0908 0706 0504 0302 0100

If, during testing, the operator pops out the run/stop switch, Test 4 prints error 4023. If dropping of units is inhibited, Test 4 will loop in error recovery, printing this error, until the error state is corrected (by some external action).

CZUDC DEV FTL ERR 04023 ON UNIT 00 TST 04 SUB 000 PC: xxxxxx

```

1          .SBTTL  COMMAND PACKET OFFSETS
2
3
4          ;GENERIC COMMAND PACKET OFFSETS
5
6          000000      P.CRF      = 0.          ;COMMAND REFERENCE NUMBER
7          000004      P.UNIT     = 4.          ;UNIT NUMBER
8          000010      P.OPCD     = 8.          ;OPCODE
9          000012      P.MOD      = 10.         ;MODIFIERS
10         000014      P.BCNT     = 12.         ;BYTE COUNT
11         000020      P.BUFF     = 16.         ;BUFFER DESCRIPTOR
12         000020      P.UADR     = 16.         ;UNIBUS ADDRESS OF BUFFER DESCRIPTOR
13         000034      P.LBN      = 28.         ;LOGICAL BLOCK NUMBER
14
15
16         .ABORT AND GET COMMAND STATUS COMMAND PACKET OFFSETS
17
18         000014      P.OTRF     = 12.         ;OUTSTANDING REFERENCE NUMBER
19
20
21         ;ONLINE AND SET UNIT CHARACTERISTICS COMMAND PACKET OFFSETS
22
23         000016      P.UNFL     = 14.         ;UNIT FLAGS
24         000020      P.HSTI     = 16.         ;HOST IDENTIFIER / RESERVED
25         000034      P.ELGF     = 28.         ;ERROR LOG FLAGS
26         000040      P.SHUN     = 32.         ;SHADOW UNIT
27         000042      P.CPSP     = 34.         ;COPY SPEED
28
29
30         ;REPLACE COMMAND PACKET OFFSETS
31
32         000014      P.RBN      = 12.         ;REPLACEMENT BLOCK NUMBER
33
34
35         ;SET CONTROLLER CHARACTERISTICS COMMAND PACKET OFFSETS
36
37         000014      P.VRSN     = 12.         ;MSCP VERSION
38         000016      P.CNTF     = 14.         ;CONTROLLER FLAGS
39         000020      P.HTMO     = 16.         ;HOST TIMEOUT
40         000022      P.USEF     = 18.         ;USE FRACTION
41         000024      P.TIME     = 20.         ;QUAD-WORD TIME AND DATE
42
43
44         ;MAINTENANCE READ AND MAINTENANCE WRITE COMMAND PACKET OFFSETS
45
46         000034      P.RGID     = 28.         ;REGION ID
47         000040      P.RGOF     = 32.         ;REGION OFFSET
48
49
50         ;EXECUTE SUPPLIED PROGRAM COMMAND PACKET OFFSETS
51
52         000024      P.DMDT     = 20.         ;DMDT TERMINAL ADDRESS (MAINT WRITE ONLY)
53         000034      P.OVRL     = 28.         ;BUFFER DESCRIPTOR FOR OVERLAYS

```

```

1          .SBTTL  END PACKET OFFSETS
2
3
4          ;GENERIC END PACKET OFFSETS
5
6          000000      P.CRF      = 0.          ;COMMAND REFERENCE NUMBER
7          000004      P.UNIT     = 4.          ;UNIT NUMBER
8          000010      P.OPCD     = 8.          ;OPCODE (ALSO CALLED ENDCODE)
9          000011      P.FLGS     = 9.          ;END PACKET FLAGS
10         000012      P.STS      = 10.         ;STATUS
11         000014      P.BCNT     = 12.         ;BYTE COUNT
12         000034      P.FBBK     = 28.         ;FIRST BAD BLOCK
13
14
15         ;GET COMMAND STATUS END PACKET OFFSETS
16
17         000014      P.OTRF     = 12.         ;OUTSTANDING REFERENCE NUMBER
18         000020      P.CMST     = 16.         ;COMMAND STATUS
19
20
21         ;GET UNIT STATUS END PACKET OFFSETS
22
23         000014      P.MLUN     = 12.         ;MULTI-UNIT CODE
24         000016      P.UNFL     = 14.         ;UNIT FLAGS
25         000020      P.HSTI     = 16.         ;HOST IDENTIFIER
26         000024      P.UNTI     = 20.         ;UNIT IDENTIFIER
27         000034      P.MEDI     = 28.         ;MEDIA TYPE IDENTIFIER
28         000040      P.SHUN     = 32.         ;SHADOW UNIT
29         000042      P.SHST     = 34.         ;SHADOW STATUS
30         000044      P.TRKS     = 36.         ;TRACK SIZE
31         000046      P.CRPS     = 38.         ;GROUP SIZE
32         000050      P.CYLS     = 40.         ;CYLINDER SIZE
33         000054      P.RCTS     = 44.         ;RCT TABLE SIZE
34         000056      P.RBNS     = 46.         ;RBNS / TRACK
35         000057      P.RCTC     = 47.         ;RCT COPIES
36
37
38         ;ONLINE AND SET UNIT CHARACTERISTICS END PACKET AND AVAILABLE
39         ;ATTENTION MESSAGE OFFSETS
40
41         000014      P.MLUN     = 12.         ;MULTI-UNIT CODE
42         000016      P.UNFL     = 14.         ;UNIT FLAGS
43         000020      P.HSTI     = 16.         ;HOST IDENTIFIER
44         000024      P.UNTI     = 20.         ;UNIT IDENTIFIER
45         000034      P.MEDI     = 28.         ;MEDIA TYPE IDENTIFIER
46         000040      P.SHUN     = 32.         ;SHADOW UNIT
47         000042      P.SHST     = 34.         ;SHADOW STATUS
48         000044      P.UNSZ     = 36.         ;UNIT SIZE
49         000050      P.VSER     = 40.         ;VOLUME SERIAL NUMBER
50
51
52         ;SET CONTROLLER CHARACTERISTICS END PACKET OFFSETS
53
54         000014      P.VRSN     = 12.         ;MSCP VERSION
55         000016      P.CNTF     = 14.         ;CONTROLLER FLAGS
56         000020      P.CTMO     = 16.         ;CONTROLLER TIMEOUT
57         000022      P.CNCL     = 18.         ;CONTROLLER COMMAND LIMIT

```

M10

58	000024	P.CNTI = 20	;CONTROLLER ID
59			
60			
61		;GET DUST STATUS END PACKET OFFSETS	
62			
63	000014	P.DEXT = 12.	;EXTENSION FOR DOWNLINE LOADABLE PROGRAM
64	000017	P.DFLG = 15.	;FLAGS
65	000020	P.DPRG = 16.	;PROGRESS INDICATOR FOR REMOTE PROGRAM
66	000024	P.DTMO = 20.	;TIMEOUT

```

1          .SBTTL  STATUS AND EVENT CODE DEFINITIONS
2
3          000037  ST.MSK  = 37          ;STATUS / EVENT CODE MASK
4          000040  ST.SUB  = 40          ;SUB-CODE MULTIPLIER
5          000000  ST.SUC  = 0           ;SUCCESS
6          000001  ST.CMD  = 1           ;INVALID COMMAND
7          000002  ST.ABO  = 2           ;COMMAND ABORTED
8          000003  ST.OFL  = 3           ;UNIT-OFFLINE
9          000004  ST.AVL  = 4           ;UNIT-AVAILABLE
10         000005  ST.MFE  = 5           ;MEDIA FORMAT ERROR
11         000006  ST.WPR  = 6           ;WRITE PROTECTED
12         000007  ST.CMP  = 7           ;COMPARE ERROR
13         000010  ST.DAT  = 10          ;DATA ERROR
14         000011  ST.HST  = 11          ;HOST BUFFER ACCESS ERROR
15         000012  ST.CNT  = 12          ;CONTROLLER ERROR
16         000013  ST.DRV  = 13          ;DRIVE ERROR
17         000037  ST.DIA  = 37          ;MESSAGE FROM AN INTERNAL DIAGNOSTIC
18         000400  ST.AOL  = 400         ;ALREADY ON-LINE
19
20         ;DUP MESSAGE TYPES
21
22         010000  DU.QUE  = 10000       ;QUESTION
23         020000  DU.DFL  = 20000       ;DEFAULT QUESTION
24         030000  DU.INF  = 30000       ;INFORMATION
25         040000  DU.TER  = 40000       ;TERMINATOR
26         050000  DU.FTL  = 50000       ;FATAL ERROR
27         060000  DU.SPC  = 60000       ;SPECIAL

```

```

1          .SBTTL  CONTROLLER TABLE DEFINITIONS
2
3          ;ONE TABLE WILL BE SET UP BY INITIALIZE SECTION FOR EACH UDA SELECTED
4          ;FOR TESTING.  TABLES ARE CONTIGUOUS.  THE END OF THE TABLES IS
5          ;MARKED BY A WORD OF ZEROS.
6
7          ;
8          ;THE FIRST TABLE IS POINTED TO BY THE CONTENTS OF CTABS.
9          ;THE NUMBER OF TABLES IS CONTAINED IN CTRLRS.
10
11         000077      CT.UNT  = 000077      ;LOGICAL UNIT NUMBER MASK
12         000777      CT.VEC  = 000777      ;VECTOR ADDRESS MASK
13         007000      CT.BRL  = 007000      ;BR LEVEL MASK
14
15         100000      CT.AVL  = BIT15      ;SET WHEN NOT AVAILABLE FOR TESTING
16         000040      CT.US0  = BIT5       ;CONTROLLER IS UDA50 IF SET/UDA52 IF CLEARED
17         000020      CT.REQ  = BIT4       ;BUFFER HAS BEEN GIVEN TO UDA FOR REQUEST
18
19         000010      CT.MSG  = BIT3       ;SET WHENEVER READ STUD DATA COMMAND GIVEN TO UDA
20
21         000002      CT.RN   = BIT1       ;MESSAGE RESPONSE RECEIVED
22         000004      CT.CMD  = BIT2       ;WHENEVER THIS BIT IS SET, CT.CMD IS CLEARED
23
24         000000      C.UADR  = 0          ;DM PROGRAM RUNNING
25         000002      C.UNIT  = 2          ;COMMAND ISSUED, WAITING FOR RESPONSE
26         000004      C.VEC  = 4          ;UNIBUS ADDRESS OF UDAIP REGISTER
27         000006      C.BST  = 6          ;UNIT NUMBER TO TEST
28         000010      C.JSR  = 10         ;VECTOR ADDRESS/BR LEVEL
29         000012      C.JAD  = 12         ;BURST LEVEL
30         000014      C.FLG  = 14         ;INTERRUPT SERVICE ROUTINE FOR CONTROLLER
31         000016      C.HCOM = 16         ;THESE TWO WORDS LOADED WITH [JSR R0  UDASRV]
32         000020      C.DR0  = 20         ;FLAGS
33         000022      C.DR1  = 22         ;BEGINNING ADRS OF HOST COMM AREA IN MEMORY
34         000024      C.DR2  = 24         ;POINTER TO DRIVE TABLES
35         000026      C.DR3  = 26         ;IF ZERO, NO DRIVE TABLE EXISTS
36         000030      C.DR4  = 30         ;
37         000032      C.DR5  = 32         ;
38         000034      C.DR6  = 34         ;
39         000036      C.DR7  = 36         ;
40         000040      C.TO   = 40         ;TIMEOUT COUNTER
41         000042      C.TOM  = 42         ;(TWO WORDS)
42         000044      C.REF  = 44         ;COMMAND REFERENCE NUMBER
43
44         000046      C.SIZE = 46         ;SIZE OF CONTROLLER TABLE IN BYTES
45
46
47

```

```

1
2
3
4
5
6
7
8
9
10      000077      DT.UNT      * 000077      ; LOGICAL UNIT NUMBER OF DRIVE
11
12      100000      DT.AVL      * BIT15      ; SET WHEN NOT AVAILABLE FOR TESTING
13      040000      D.IW       * BIT14      ; INITIAL WRITE
14      020000      D.DCY      * BIT13      ; DIAGNOSTIC CYLINDERS
15      010000      D.ECC      * BIT12      ; ECC CORRECTION ENABLED
16      004000      D.RD       * BIT11      ; READ ONLY
17      002000      D.WO       * BIT10      ; WRITE ONLY
18      001000      D.RET      * BIT9       ; RETRIES ENABLED
19      000400      D.CYL      * BIT8       ; START/END CYLINDERS SPECIFIED
20      000100      D.SEG      * BIT6       ; SEQUENTIAL ACCESS
21      000040      D.BE       * BIT5       ; BEGIN/END BLOCKS USED
22      000020      D.TR       * BIT4       ; WHEN D.BE=0: 1 - TRACKS, 0 - GROUPS
23      000010      D.WC       * BIT3       ; WRITE CHECKS ENABLED
24      000004      D.WCA      * BIT2       ; ALWAYS WRITE CHECK
25      000002      D.DC       * BIT1       ; DATA COMPARES ENABLED
26      000001      D.DCA      * BIT0       ; ALWAYS DATA COMPARE
30      011012      DDEF      * D.ECC+D.WC+D.DC+D.RET ; DEFAULT D.PRM
32      140200      D.ZERO     * BIT15+BIT7+D.IW ; BITS TO BE CLEARED
33
35      000000      D.DRV      * 0          ; DRIVE NUMBER
36      000002      D.UNIT     * 2          ;
37      000004      D.PRM      * 4          ; HARDWARE QUESTION FLAGS
38      000006      D.PAT      * 6          ; DATA PATTERN NUMBER
39      000010      D.BB       * 10         ; BAD BLOCK COUNT
40      000012      D.BB01     * 12         ; BAD BLOCK 1
41      000016      D.BB02     * 16         ;
42      000022      D.BB03     * 22         ;
43      000026      D.BB04     * 26         ;
44      000032      D.BB05     * 32         ;
45      000036      D.BB06     * 36         ;
46      000042      D.BB07     * 42         ;
47      000046      D.BB08     * 46         ;
48      000052      D.BB09     * 52         ;
49      000056      D.BB10     * 56         ;
50      000062      D.BB11     * 62         ;
51      000066      D.BB12     * 66         ;
52      000072      D.BB13     * 72         ;
53      000076      D.BB14     * 76         ;
54      000102      D.BB15     * 102        ;
55      000106      D.BB16     * 106        ;
    
```

1			
2	000112	D.BEC	▪ 112
3	000114	D.BGN1	▪ 114
4	000120	D.END1	▪ 120
5	000124	D.BGN2	▪ 124
6	000130	D.END2	▪ 130
7	000134	D.BGN3	▪ 134
8	000140	D.END3	▪ 140
9	000144	D.BGN4	▪ 144
10	000150	D.END4	▪ 150
11	000154	D.BCYL	▪ 154
12	000160	D.ECYL	▪ 160
13	000164	D.XFRW	▪ 164
14	000166	D.XFRR	▪ 166
15	000170	D.HERR	▪ 170
16	000172	D.SERR	▪ 172
17	000174	D.SEEK	▪ 174
18	000176	D.ECCC	▪ 176
19	000200	D.SERN	▪ 200
24			
25	000206	D.SIZE	▪ 206
26			
27		;DM PROGRAM HEADER DEFINITIONS	
28			
29	000000	DMTRLN	▪ 0
30	000004	DMOVL	▪ 4
31	000040	DMMAIN	▪ 40
32	001000	DMFRST	▪ 1000

  

			;BEGIN/END SET COUNT
			;BEGIN BLOCK 1
			;END
			;BEGIN BLOCK 2
			;END
			;BEGIN BLOCK 3
			;END
			;BEGIN BLOCK 4
			;END
			;BEGIN CYLINDER
			;END CYLINDER
			;MEGABITS WRITTEN COUNT
			;MEGABITS READ COUNT
			;HARD ERROR COUNTER
			;SOFT ERROR COUNTER
			;NUMBER OF SEEKS X1000
			;ECC COUNTER
			;DRIVE SERIAL NUMBER
			;SIZE OF DRIVE TABLE IN BYTES
			;OFFSET TO SIZE OF PROGRAM NEEDING DOWNLINE LOAD
			;OFFSET TO SIZE OF OVERLAY
			;OFFSET TO FIRST WORD OF MAIN PROGRAM
			;ADDRESS IN DM FILE CONTAINING FIRST BYTE OF HEADER

```

1          .SBTTL GLOBAL DATA SECTION
2
3
4          ;**
5          ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
6          ; IN MORE THAN ONE TEST.
7          ;--
8 064402   LERRTBL::
064402   000000   ERR1YP::          .WORD 0
064404   000000   ERRNBR::         .WORD 0
064406   000000   ERRMSG::         .WORD 0
064410   000000   ERRBLK::         .WORD 0
9
10 064412   FFREE::  .BLKW 1          ; FIRST FREE WORD IN MEMORY
11 064414   FSIZE::  .BLKW 1          ; SIZE OF FREE MEMORY IN WORDS
12 064416   FMEM:   .BLKW 1          ; COPY OF FFREE AT END OF INIT SECTION
13 064420   FMEMS:  .BLKW 1          ; COPY OF FSIZE AT END OF INIT SECTION
14 064422   DTABS::  .BLKW 1          ; START OF DRIVE TABLE STORAGE
15 064424   CTABS::  .BLKW 1          ; START OF CONTROLLER TABLE STORAGE
16 064426   CTRLRS: .BLKW 1          ; COUNT OF UDA CONTROLLERS IN PTABLES
17 064430   TSTTAB: .BLKW 1          ; POINTER TO 1ST CONTROLLER TABLE UNDER TEST
18 064432   DMPRG:  .BLKW 1          ; START ADDRESS OF UDA52 DM PROGRAM
19
20 064434   KTBASA:  .BLKW 1          ; HIGH TWO BYTES OF BASE ADDRESS FOR KT ACCESS
21 064436   KTBASO:  .BLKW 1          ; LOW BYTE OF ADDRESS FOR KT ACCESS
22
23 064440   IFLAGS:: .BLKW 1          ; FLAGS FROM INIT CODE FOR TEST 4
24
25          000002   ICONT   == BIT1   ; CONTINUE EVENT FLAG
26          000004   IREST   == BIT2   ; RESTART FLAG
27          000010   ISTRT   == BIT3   ; START FLAG
28          000020   ISTRTH  == BIT4   ; START FLAG HOLD FOR T4UPRM ROUTINE
29
30 064442   FNUM:    .WORD 0          ; FILE # IN PAK FILE THAT IS CURRENTLY LOADED
31 064444   TNUM:    .WORD 0          ; NUMBER OF TEST EXECUTING
32 064446   URUN:    .BLKW 1          ; NUMBER OF UNITS TO RUN AT ONE TIME
33 064450   URNING:  .BLKW 1          ; NUMBER OF UNITS STILL RUNNING
34 064452   UCNT:    .BLKW 1          ; COUNTER OF UNITS UNDER TEST
35 064454   INTRCV:  .BLKW 1          ; INTERRUPT RECEIVED FLAG FOR INT TESTING

```

```

1
5
7 064456      132      125      104  FNAME:  .ASCIZ  \ZUDCEO.PAK\      ;NAME OF DATA FILE
8                                     .EVEN
9
10 064472 000000      FDATA:  .WORD 0
11 064474 000000      FILOPN: .WORD 0      ;FILE OPEN WHEN NON-ZERO
12 064476      TEMP:   .BLKW 12.      ;TEMPORARY STORAGE FOR GMANI RESPONSES
13
14 064526      125      065      062  U52EXT: .ASCII  "U52"
15                                     .EVEN
19
20 064532 000000      TYPCNT: .WORD 0      ; TYPE OF CONTROLLER WORD
21
22          000002      TY.U50  = BIT1
23          000001      TY.U52  = BIT0
24
25 064534 000000      IPADRS: .WORD 0      ; EIGHT ENTRIES
26 064536 000000      .WORD 0
27 064540 000000      .WORD 0
28 064542 000000      .WORD 0
29 064544 000000      .WORD 0
30 064546 000000      .WORD 0
31 064550 000000      .WORD 0
32 064552 000000      .WORD 0
33
34 064554 000001      PAT16C: .WORD 1      ;COUNT OF WORDS IN DATA PATTERN 16
35 064556 000000      PAT16W: .WORD 0      ;WORD SEQUENCE FOR DATA PATTERN 16
36 064560 000000      .WORD 0
37 064562 000000      .WORD 0
38 064564 000000      .WORD 0
39 064566 000000      .WORD 0
40 064570 000000      .WORD 0
41 064572 000000      .WORD 0
42 064574 000000      .WORD 0
43 064576 000000      .WORD 0
44 064600 000000      .WORD 0
45 064602 000000      .WORD 0
46 064604 000000      .WORD 0
47 064606 000000      .WORD 0
48 064610 000000      .WORD 0
49 064612 000000      .WORD 0
50 064614 000000      .WORD 0
    
```

```

1
2
3      ;KW11 CLOCK CONTROL
4 064616 000000      KW.CSR: .WORD 0      ;CSR OF CLOCK
5 064620              KW.BRL: .BLKW 1      ;BR LEVEL
6 064622              KW.VEC: .BLKW 1      ;VECTOR
7 064624              KW.HZ: .BLKW 1      ;HERTZ (50. OR 60.)
8 064626              KW.EL: .BLKW 2      ;ELAPSED TIME
9 064632              STIME: .BLKW 2      ;STATISTICAL REPORT TIMER
10
11 064636              NXMAD: .BLKW 1      ;SET TO ALL ONES BY NON-EXISTANT ADDRESS
12 064640 177777      KTMEM: .WORD -1     ;SET TO ALL ONES IF NO KT EXISTS
13
14 064642              T2WRR: .BLKW 1      ;WRITE/READ REGION
15 064644              T2WRO: .BLKW 1      ;WRITE/READ OFFSET
16 064646              T2DR: .BLKW 1      ;DIAGNOSE REGION
17
18
19      ;ERROR LOG CONTROL WORDS
20
21 064650              LBUFS: .BLKW 1      ;START ADDRESS OF LOG/ZERO IF NONE
22 064652              LBUFN: .BLKW 1      ;ADDRESS FOR MORE DATA FOR LOG
23 064654              LBUFE: .BLKW 1      ;LAST ADDRESS AVAILABLE FOR LOG DATA
24
25      ;DISK DIAGNOSTIC DLL CONTROL WORDS
26
27 064656              DLL: .BLKW 1      ;DOWNLINE LOAD RESPONSE CODE = 0 - NO DATA,
28                                ;1 - PROGRAM PROVIDED, 2- PROGRAM NOT FOUND
29 064660              DLLDR: .BLKW 1      ;DRIVE NUMBER REQUESTING PROGRAM
30 064662              DLLV: .BLKW 1      ;A VALUE FROM DM PROGRAM TO BE RETURNED
31 064664              DLLR: .BLKW 1      ;REGION
32 064666              DLLADR: .BLKW 2     ;ADDRESS WHERE PROGRAM STORED
33 064672              DLLSIZ: .BLKW 1     ;SIZE OF PROGRAM IN BYTES
34 064674              DLLNAM: .BLKW 2     ;NAME OF PROGRAM IN RAD50

```

```

1      .SBTTL  GLOBAL TEXT SECTION
2
3
4      ;**
5      ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
6      ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
7      ; MORE THAN ONE TEST.
8      ;**
9
10     ;NAMES OF DEVICES SUPPORTED BY PROGRAM
11     ;
12     ;L$DVTYP:
13     ;
14     064700      114      117      107      .ASCIZ  /LOGICAL DISK DRIVE/
15     064700      .EVEN
16
17     ; TEST DESCRIPTION
18     ;
19     ;L$DESC:
20     ;
21     064724      103      132      125      .ASCIZ  /CZUDCEO UDA & DISK DRV DIAG/
22     064724      .EVEN
23
24     ; UNFORMATTED MESSAGES
25     ;
26     064760      040      040      000      T4OPT7: .ASCIZ  \ \ \
27     064763      101      122      105      INITWC: .ASCIZ  \ARE YOU SURE CUSTOMER DATA CAN BE DESTROYED\
28
29     ; FORMAT STATEMENTS USED IN PRINT CALLS
30     ;
31     065037      045      124      000      FRMTT:  .ASCIZ  \*T\
32     065042      045      116      000      CRLF:   .ASCIZ  \*N\
33     065045      042      040      040      RNTIM:  .ASCIZ  \ "  RUNTIME "D16": "\
34     065070      104      071      042      RNTIM1: .ASCIZ  \D9": "\
35     065076      104      071      000      RNTIM2: .ASCIZ  \D9\
36     065101      042      040      040      ERRME1: .ASCIZ  \ " * * * ERROR PROCESSING MESSAGE STRING * * * "\
37     065170      116      042      122      MXFERP: .ASCIZ  \N"REACHED TRANSFER LIMIT - TESTING STOPPED"N\
38     065245      116      042      125      ERRLIM: .ASCIZ  \N"UNIT "D6" REACHED ERROR LIMIT - WILL NO LONGER BE TESTED"N\
39     065342      116      042      124      INTSTO: .ASCIZ  \N"TESTING INTERRUPT ABILITY OF UDA AT ADR "016" VEC "09"..."\
40     065437      042      103      117      INTST1: .ASCIZ  \ "COMPLETED"N\
41     065454      116      042      103      INITWA: .ASCIZ  \N"CUSTOMER DATA WILL BE DESTROYED ON:"NS5"UNIT"S5"UDA AT"S3"DRIVE"N\
42     065560      045      123      066      INITWB: .ASCIZ  \S6#D2#S6#06#S4#D3#N\
43     065605      116      042      115      T4WARN: .ASCIZ  \N"MANUAL INTERVENTION NOT ALLOWED. TEST 4 USING DEFAULT PARAMETERS"N\
44     065713      116      042      125      MESSG:  .ASCIZ  \N"UNIT "D6" UDA AT "016" DRIVE "D9S\
45     065757      116      042      115      T2WARN: .ASCIZ  \N"MANUAL INTERVENTION NOT ALLOWED. TEST 2 RUNNING UNATTENDED"N\
46     066056      116      042      124      T2CMS1: .ASCII   \N"TEST #2 MANUAL INTERVENTION ON UNIT "D8" UDA AT "016" DRIVE "D9N\
47     066160      042      124      117      .ASCII   \ "TO WRITE AND READ MEMORY: "N\
48     066214      042      040      040      .ASCII   \ " W DATA REGION OFFSET"N\
49     066245      042      040      040      .ASCII   \ " R REGION OFFSET"N\
50     066271      042      124      117      .ASCII   \ "TO RUN A DIAGNOSTIC: "N\
51     056320      042      040      040      .ASCII   \ " D REGION"N\
52     066335      042      124      117      .ASCII   \ "TO EXIT QUESTIONING: "N\
53     066364      042      040      040      .ASCII   \ " E"N\
54     066372      042      104      101      .ASCIZ   \ "DATA, REGION AND OFFSET ARE HEX VALUES. "N\
55     066445      042      077      040      T2CMS5: .ASCIZ   \ "? INPUT ERROR"N\
    
```

## GLOBAL TEXT SECTION

79	066466	042	116	117	NOCLOCK: .ASCIZ	\ "NO LINE CLOCK AVAILABLE FOR TIMING EVENTS:"\	
80	066543	116	042	103	LOGM1: .ASCIZ	\ "CONTENTS OF ERROR LOG:"\	
81	066575	116	042	105	LOGM2: .ASCIZ	\ "END OF ERROR LOG"\	
82	066622	116	042	105	LOGM3: .ASCIZ	\ "ERROR LOG IS EMPTY"\	
83							
84	066651	042	110	117	BASNO: .ASCIZ	\ "HOST PROGRAM"\	
86	066670	042	125	116	BASN1: .ASCIZ	\ "UNIBUS ADDRESSING"\	
87	066714	042	104	111	BASN2: .ASCIZ	\ "DISK RESIDENT"\	
88	066734	042	104	111	BASN3: .ASCIZ	\ "DISK FUNCTION"\	
91	066754	042	104	111	BASN4: .ASCIZ	\ "DISK EXERCISER"\	
93	066775	042	040	040	BASL1: .ASCIZ	\ " DM PC:"012\	
94	067013	042	040	040	BASL2: .ASCIZ	\ " UDA AT "016\	
95	067032	042	040	040	BASL3: .ASCIZ	\ " DRIVE "D9\	
96	067047	000			BAS: .BYTE 0		;NULL TO PRINT NOTHING
97							
98	067050	122	066	122	BASLN: .ASCIZ	\R6R6R6R6\	;USED TO PRINT BASIC LINE OF ERROR MESSAGE

## GLOBAL TEXT SECTION

1	067061				X1A:	
2	067061				X2A:	
3	067061				X3A:	
4	057061	042	111	040	X8A:	.ASCIZ \ "I DON'T LIKE THE ANSWERS YOU GAVE TO THE HARDWARE QUESTIONS" N\
5	067160	122	065	122	X1:	.ASCIZ \R5R6"UDA HAS MORE THAN ONE VECTOR, BR LEVEL OR BUPST RATE" N\
6	067254	122	065	122	X2:	.ASCIZ \R5R6"TWO UNITS SELECT THE SAME DRIVE" N\
7	067323	122	065	122	X3:	.ASCIZ \R5R6"MORE THAN EIGHT DRIVES SELECTED ON THIS UDA" N\
8	067406	122	064	042	X4:	.ASCII \R4"NOT ENOUGH ROOM IN MEMORY TO TEST THE UNITS SELECTED" N\
9	067477	042	120	114		.ASCIZ \ "PLEASE START PROGRAM OVER AND TEST FEWER UNITS AT A TIME" N\
10	067573	122	064	042	X6:	.ASCIZ \R4"TABLE CONSISTANCY ERROR, PLEASE RE-LOAD PROGRAM" N\
11	067660	122	065	122	X8:	.ASCIZ \R5R6"TWO UDA'S USE THE SAME VECTOR" N\
13	067725	122	064	042	X5:	.ASCIZ \R4"CHECKSUM ERROR IN DM PROGRAM FILE " N\
14	067775	122	064	042	X7:	.ASCIZ \R4"ERROR IN DM PROGRAM FILE, DM PROGRAM NOT FOUND" N\
16	070061	122	064	042	X14:	.ASCII \R4"UDA50 CONTROLLER IS AT A REVISION LEVEL NO LONGER SUPPORTED" N\
17	070161	042	102	131		.ASCII \ "BY THIS DIAGNOSTIC PROGRAM, THIS PROGRAM REQUIRES A UDA50-A" N\
18	070260	042	103	117		.ASCII \ "CONTROLLER (MODEL 6) WITH MICROCODE REVISION AT 3 OR GREATER, " N\
19	070360	116	042	103		.ASCIZ \N"CONTROLLER REPORTED MODE CODE "04" AND MICROCODE VERSION "04N\
35	070460	122	065	042	X38:	.ASCII \R5"MEMORY ERROR TRYING TO READ UDA REGISTERS" N\
36	070531	042	103	110		.ASCII \ "CHECK UNIBUS SELECTION SWITCHES ON UDA MODULE M7485" N\
37	070624	042	117	122		.ASCII \ "OR UNIBUS" N\
38	070640	042	117	122		.ASCIZ \ "OR "R7\
39	070650	122	065	042	X21:	.ASCII \R5"UDA RESIDENT DIAGNOSTICS DETECTED FAILURE" NR8\
40	070730	042	122	105		.ASCIZ \ "REPLACE UDA MODULE M748"03N\
41	070765	122	065	042	X22:	.ASCII \R5"STEP BIT DID NOT SET IN UDASA REGISTER DURING INITIALIZATION" N\
42	071066	042	123	124		.ASCIZ \ "STEP BIT EXPECTED "016NR8R7\
43	071123	122	065	042	X23A:	.ASCII \R5"UDA DID NOT CLEAR RING STRUCTURE IN HOST MEMORY DURING INITIALIZATION" N\
44	071235	104	071	042		.ASCII \09" WORDS WERE TO BE CLEARED STARTING AT ADDRESS "016N\
45	071323	042	106	111		.ASCII \ "FIRST SEVERAL WORDS NOT CLEARED (UP TO 6):" N\
46	071400	123	066	042		.ASCIZ \S6"ADDRESS" S4" CONTENTS" N\
47	071431	123	067	117	X23B:	.ASCIZ \S7016S5016N\
48	071445	122	065	042	X24:	.ASCII \R5"UDASA REGISTER DID NOT GO TO ZERO AFTER STEP 3 WRITE OF INITIALIZATION" N\
49	071560	042	120	125		.ASCIZ \ "PURGE/POLE DIAGNOSTICS WERE REQUESTED" NR8R7\
50	071635	122	065	042	X25:	.ASCII \R5"UDA DID NOT RETURN CORRECT DATA IN UDASA REGISTER DURING INITIALIZATION" N\
51	071751	042	040	040		.ASCIZ \ " UDASA EXPECTED "016NR8R7\
53	072006	122	065	042	X26:	.ASCII \R5"DATA COMPARISON ERROR DURING DIAGNOSTIC PORT LOOP TEST" N\
54	072101	042	040	040		.ASCII \ " DATA SENT TO UDASA "016N\
55	072135	042	040	040		.ASCIZ \ " RECEIVED FROM UDASA "016NR7\
56	072174	122	065	042	X27:	.ASCII \R5"UDASA REGISTER DID NOT CHANGE AFTER WRITING TO IT" N\
57	072262	042	111	116		.ASCIZ \ "IN PORT LOOP DIAGNOSTIC" NR8R7\
58	072321	122	065	042	X28:	.ASCIZ \R5"UDA DID NOT INTERRUPT THE PDP-11" NR7\
59	072371	122	065	042	X29:	.ASCII \R5"UDA INTERRUPTED AT DIFFERENT BR LEVEL THAN SPECIFIED IN HARDWARE" N\
60	072476	042	121	125		.ASCII \ "QUESTIONS, INTERRUPT WAS AT BR LEVEL "03N\
61	072550	042	103	110		.ASCII \ "CHECK PRIORITY PLUG ON UDA MODULE M7485" N\
62	072622	042	117	122		.ASCIZ \ "OR CHANGE HARDWARE QUESTIONS" N\
64	072662	122	065	042	X30:	.ASCIZ \R5"UDA REPORTED FATAL ERROR IN UDASA REGISTER WHILE RUNNING DM PROGRAM" NR8\
65	072775	122	065	042	X31:	.ASCII \R5"NO INTERRUPT RECEIVED FROM DM PROGRAM FOR 3 MINUTES" N\
66	073065	042	101	123		.ASCIZ \ "ASSUME PROGRAM IS HUNG" N\
67	073117	122	065	042	X32:	.ASCIZ \R5"MESSAGE BUFFER RECEIVED FROM DM PROGRAM WITH UNKNOWN REQUEST NUMBER" N\
68	073230	122	065	042	X35:	.ASCIZ \R5"DM PROGRAM ASKED FOR DATA ON UNKNOWN DRIVE" N\
69	073310	122	065	042	X36:	.ASCII \R5"NO INTERRUPT RECEIVED FROM UDA FOR 30 SECONDS" N\
70	073372	042	127	110		.ASCIZ \ "WHILE LOADING DM PROGRAM" N\
71	073426	122	065	042	X37:	.ASCIZ \R5"UDA REPORTED FATAL ERROR IN UDASA REGISTER WHILE LOADING DM PROGRAM" NR8R

GLOBAL TEXT SECTION

```

1 073543 042 115 105 XMSG1: .ASCIZ \MESSAGE BUFFER CONTAINS:"N\
2 073577 123 063 117 XMSG2: .ASCIZ \S3016S1016S1016S1016S1016S1016N\
3 073644 122 065 042 XPKT1: .ASCII \R5"RESPONSE PACKET FROM UDA DOES NOT CONTAIN EXPECTED DATA"N\
4 073740 042 105 111 .ASCII \EITHER UDA RETURNED ERROR STATUS OR PACKET WAS NOT RECEIVED CORRECTLY"N\
5 074050 123 063 042 .ASCIZ \S3"COMMAND PACKET SENT"S6"RESPONSE PACKET RECEIVED"N\
6 074135 123 066 117 XPKT2: .ASCIZ \S6016S1016S14016S1016N\
7 074164 042 040 040 XSA: .ASCIZ \ UDASA CONTAINS "016N\
8 074215 042 122 105 XFRU: .ASCIZ \REPLACE UDA MODULE M7485"N\
12 .EVEN

```

```

1          .SBTTL  GLOBAL ERROR REPORT SECTION
2
3          ;++
4          ; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS
5          ; USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION.  PRINTB
6          ; (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
7          ;--
25
26 074252          ERR001::
27 074252 012746 067061      MOV     #X1A,-(SP)          ;PUSH #X1A ON STACK
   074256 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE
   074262 067160              .WORD   X1              ;ADDRESS OF ASCIZ STRING
   074264 000002              .WORD   ARG.CT         ;ARGUMENT COUNT * 2
28 074266          L10002:
   074266 104423          TRAP   C$MSG
29
30 074270          ERR002::
31 074270 012746 067061      MOV     #X2A,-(SP)          ;PUSH #X2A ON STACK
   074274 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE
   074300 067254              .WORD   X2              ;ADDRESS OF ASCIZ STRING
   074302 000002              .WORD   ARG.CT         ;ARGUMENT COUNT * 2
32 074304          L10003:
   074304 104423          TRAP   C$MSG
33
34 074306          ERR003::
35 074306 012746 067061      MOV     #X3A,-(SP)          ;PUSH #X3A ON STACK
   074312 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE
   074316 067323              .WORD   X3              ;ADDRESS OF ASCIZ STRING
   074320 000002              .WORD   ARG.CT         ;ARGUMENT COUNT * 2
36 074322          L10004:
   074322 104423          TRAP   C$MSG
37
38 074324          ERR004::
39 074324 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE
   074330 067406              .WORD   X4              ;ADDRESS OF ASCIZ STRING
   074332 000000              .WORD   ARG.CT         ;ARGUMENT COUNT * 2
40 074334          L10005:
   074334 104423          TRAP   C$MSG
41
42 074336          ERR005::
43 074336 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE
   074342 067725              .WORD   X5              ;ADDRESS OF ASCIZ STRING
   074344 000000              .WORD   ARG.CT         ;ARGUMENT COUNT * 2
44 074346          L10006:
   074346 104423          TRAP   C$MSG
45
46 074350          ERR007::
47 074350 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE
   074354 067775              .WORD   X7              ;ADDRESS OF ASCIZ STRING
   074356 000000              .WORD   ARG.CT         ;ARGUMENT COUNT * 2
48 074360          L10007:
   074360 104423          TRAP   C$MSG
49
50
51 074362          ERR014::
52 074362 010146      MOV     R1,-(SP)          ;PUSH R1 ON STACK
53 074362 010346      MOV     R3,-(SP)          ;PUSH R3 ON STACK
   074366 004137 075674      JSR     R1,LPNTB          ;CALL  LPNTB PRINT ROUTINE

```



074544	010246		MOV	R2, -(SP)	;PUSH R2 ON STACK
074546	004137	075674	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
074552	071431		.WORD	X23B	;ADDRESS OF ASCIZ STRING
074554	000004		.WORD	ARG.CT	;ARGUMENT COUNT * 2
97	074556	005304	DEC	R4	
98	074560	001403	BEQ	ERR23C	
99	074562	005722	ERR23B:	TST (R2)+	
100	074564	005303	DEC	R3	
101	074566	001363	BNE	ERR23A	
102	074570		ERR23C:		
	074570	004137	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
	074574	074215	.WORD	XFRU	;ADDRESS OF ASCIZ STRING
	074576	000000	.WORD	ARG.CT	;ARGUMENT COUNT * 2
103	074600		L10015:		
	074600	104423	TRAP	C\$MSG	
104					
105	074602		ERR024::		
106	074602	010246	MOV	R2, -(SP)	;PUSH R2 ON STACK
	074604	004137	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
	074610	071445	.WORD	X24	;ADDRESS OF ASCIZ STRING
	074612	000002	.WORD	ARG.CT	;ARGUMENT COUNT * 2
107	074614		L10016:		
	074614	104423	TRAP	C\$MSG	
108					
109	074616		ERR025::		
110	074616	010246	MOV	R2, -(SP)	;PUSH R2 ON STACK
	074620	010146	MOV	R1, -(SP)	;PUSH R1 ON STACK
	074622	004137	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
	074626	071635	.WORD	X25	;ADDRESS OF ASCIZ STRING
	074630	000004	.WORD	ARG.CT	;ARGUMENT COUNT * 2
111	074632		L10017:		
	074632	104423	TRAP	C\$MSG	
112					
114	074634		ERR026::		
115	074634	016446	MOV	2(R4), -(SP)	;PUSH 2(R4) ON STACK
	074640	010246	MOV	R2, -(SP)	;PUSH R2 ON STACK
	074642	004137	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
	074646	072006	.WORD	X26	;ADDRESS OF ASCIZ STRING
	074650	000004	.WORD	ARG.CT	;ARGUMENT COUNT * 2
116	074652		L10020:		
	074652	104423	TRAP	C\$MSG	
117					
118	074654		ERR027::		
119	074654	016446	MOV	2(R4), -(SP)	;PUSH 2(R4) ON STACK
	074660	004137	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
	074664	072174	.WORD	X27	;ADDRESS OF ASCIZ STRING
	074666	000002	.WORD	ARG.CT	;ARGUMENT COUNT * 2
120	074670		L10021:		
	074670	104423	TRAP	C\$MSG	
121					
122	074672		ERR028::		
123	074572	004137	JSR	R1, LPNTB	;CALL LPNTB PRINT ROUTINE
	074576	072321	.WORD	X28	;ADDRESS OF ASCIZ STRING
	074700	000000	.WORD	ARG.CT	;ARGUMENT COUNT * 2
124	074702		L10022:		
	074702	104423	TRAP	C\$MSG	
125					

126	074704			ERR029::		
127	074704	010146		MOV	R1, -(SP)	; PUSH R1 ON STACK
	074706	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	074712	072371		.WORD	X29	; ADDRESS OF ASCIZ STRING
	074714	000002		.WORD	ARG, CT	; ARGUMENT COUNT * 2
128	074716			L10023:		
	074716	104423		TRAP	C MSG	
130						
131	074720			ERR030::		
132	074720	010146		MOV	R1, -(SP)	; PUSH R1 ON STACK
	074722	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	074726	072662		.WORD	X30	; ADDRESS OF ASCIZ STRING
	074730	000002		.WORD	ARG, CT	; ARGUMENT COUNT * 2
133	074732			L10024:		
	074732	104423		TRAP	C MSG	
134						
135	074734			ERR031::		
136	074734	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	074740	072775		.WORD	X31	; ADDRESS OF ASCIZ STRING
	074742	000000		.WORD	ARG, CT	; ARGUMENT COUNT * 2
137	074744			L10025:		
	074744	104423		TRAP	C MSG	
138						
139	074746			ERR032::		
140	074746	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	074752	073117		.WORD	X32	; ADDRESS OF ASCIZ STRING
	074754	000000		.WORD	ARG, CT	; ARGUMENT COUNT * 2
141	074756	004737	075150	CALL	MSGPKT	
142	074762			L10026:		
	074762	104423		TRAP	C MSG	
143						
144	074764			ERR033::		
145	074764	004737	075056	CALL	PNTPKT	
146	074770			L10027:		
	074770	104423		TRAP	C MSG	
147						
148	074772			ERR034::		
149	074772	004737	075056	CALL	PNTPKT	
150	074776			L10030:		
	074776	104423		TRAP	C MSG	
151						
152	075000			ERR035::		
153	075000	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075004	073230		.WORD	X35	; ADDRESS OF ASCIZ STRING
	075006	000000		.WORD	ARG, CT	; ARGUMENT COUNT * 2
154	075010	004737	075150	CALL	MSGPKT	
155	075014			L10031:		
	075014	104423		TRAP	C MSG	
156						
157	075016			ERR036::		
158	075016	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075022	073310		.WORD	X36	; ADDRESS OF ASCIZ STRING
	075024	000000		.WORD	ARG, CT	; ARGUMENT COUNT * 2
159	075026			L10032:		
	075026	104423		TRAP	C MSG	
160						
161	075030			ERR037::		

162	075030	010146		MOV	R1, -(SP)	; PUSH R1 ON STACK
	075032	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075036	073426		.WORD	X37	; ADDRESS OF ASCIZ STRING
	075040	000002		.WORD	ARG.CT	; ARGUMENT COUNT * 2
163	075042			L10033:		
	075042	104423		TRAP	C1MSG	
164						
165	075044			ERRO38:		
166	075044	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075050	070460		.WORD	X38	; ADDRESS OF ASCIZ STRING
	075052	000000		.WORD	ARG.CT	; ARGUMENT COUNT * 2
167	075054			L10034:		
	075054	104423		TRAP	C1MSG	
168						
169	075056			PNTPKT:		
	075056	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075062	073644		.WORD	XPKT1	; ADDRESS OF ASCIZ STRING
	075064	000000		.WORD	ARG.CT	; ARGUMENT COUNT * 2
170	075066	010401		MOV	R4, R1	
171	075070	062701	000020	ADD	#HC.CPK, R1	
172	075074	010402		MOV	R4, R2	
173	075076	062702	000020	ADD	#HC.MPK, R2	
174	075102	012703	000014	MOV	#12., R3	
175						
176	075106			PNTPKL:		
	075106	011246		MOV	(R2), -(SP)	; PUSH (R2) ON STACK
	075110	016246	000002	MOV	2(R2), -(SP)	; PUSH 2(R2) ON STACK
	075114	011146		MOV	(R1), -(SP)	; PUSH (R1) ON STACK
	075116	016146	000002	MOV	2(R1), -(SP)	; PUSH 2(R1) ON STACK
	075122	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075126	074135		.WORD	XPKT2	; ADDRESS OF ASCIZ STRING
	075130	000010		.WORD	ARG.CT	; ARGUMENT COUNT * 2
177	075132	062701	000004	ADD	#4, R1	
178	075136	062702	000004	ADD	#4, R2	
179	075142	005303		DEC	R3	
180	075144	001360		BNE	PNTPKL	
181	075146	000207		RETURN		
182						
183	075150			MSGPKT:		
	075150	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075154	073543		.WORD	XMSG1	; ADDRESS OF ASCIZ STRING
	075156	000000		.WORD	ARG.CT	; ARGUMENT COUNT * 2
184	075160	016504	000016	MOV	C.HCOM(R5), R4	
185	075164	062704	000206	ADD	#HC.BF2, R4	
186	075170	012703	000005	MOV	#5, R3	
187	075174			MSGPKL:		
	075174	016446	000014	MOV	12.(R4), -(SP)	; PUSH 12.(R4) ON STACK
	075200	016446	000012	MOV	10.(R4), -(SP)	; PUSH 10.(R4) ON STACK
	075204	016446	000010	MOV	8.(R4), -(SP)	; PUSH 8.(R4) ON STACK
	075210	016446	000006	MOV	6.(R4), -(SP)	; PUSH 6.(R4) ON STACK
	075214	016446	000004	MOV	4.(R4), -(SP)	; PUSH 4.(R4) ON STACK
	075220	016446	000002	MOV	2.(R4), -(SP)	; PUSH 2.(R4) ON STACK
	075224	011446		MOV	(R4), (SP)	; PUSH (R4) ON STACK
	075226	004137	075674	JSR	R1, LPNTB	; CALL LPNTB PRINT ROUTINE
	075232	073577		.WORD	XMSG2	; ADDRESS OF ASCIZ STRING
	075234	000016		.WORD	ARG.CT	; ARGUMENT COUNT * 2
188	075236	062704	000016	ADD	#14., R4	

189	075242	005303		DEC	R3	
190	075244	001353		BNE	MSGPKL	
191	075246	000207		RETURN		
192						
193	075250					
194	075250	013702	064444	ERR, TN: :	MOV	TNUM, R2 ;GET TEST NUMBER
195	075254	006302			ASL	R2 ;DOUBLE
196	075256	012703	067032		MOV	#BASL3, R3 ;GET ADDRESS OF DRIVE PRINT LINE
197	075262	005764	000004		TST	4(R4) ;CHECK IF DRIVE NUMBER GIVEN
198	075266	100002			BPL	1 ;BRANCH IF SO
199	075270	012703	067047		MOV	#BAS, R3
200	075274			1\$:		
	075274	016446	000004		MOV	4(R4), -(SP) ;PUSH 4(R4) ON STACK
	075300	010346			MOV	R3, -(SP) ;PUSH R3 ON STACK
	075302	011546			MOV	(R5), -(SP) ;PUSH (R5) ON STACK
	075304	012746	067013		MOV	#BASL2, -(SP) ;PUSH #BASL2 ON STACK
	075310	011446			MOV	(R4), -(SP) ;PUSH (R4) ON STACK
	075312	012746	066775		MOV	#BASL1, -(SP) ;PUSH #BASL1 ON STACK
	075316	016246	076272		MOV	TNAMES-2(R2), -(SP) ;PUSH TNAMES-2(R2) ON STACK
	075322	004137	075674		JSR	R1, LPNTB ;CALL LPNTB PRINT ROUTINE
	07532C	067050			.WORD	BASLN ;ADDRESS OF ASCIZ STRING
	075330	000016			.WORD	ARG, CT ;ARGUMENT COUNT * 2
202	075332	004137	106362		CALL	RNTIME ;GET RUNTIME PARAMETERS
203	075336	112700	000015		MOVB	#CR, R0 ;STORE #CR IN R0 AND
	075342	004737	075506		JSR	PC, PRINTC ;PRINT THE CHARACTER.
204	075346	062704	000006		ADD	#6, R4 ;INCREASE R4 TO POINT TO MESSAGE POINTER
205	075352	012402			MOV	(R4)+, R2 ;GET MESSAGE POINTER
206	075354	006302			ASL	R2 ;DOUBLE TO MAKE BYTE OFFSET
207	075356	063702	064432		ADD	DMPROG, R2 ;ADD TO START OF MESSAGE STRINGS
208	075362	067702	167044		ADD	BDMPROG, R2 ;ADD SIZE OF MAIN PROGRAM
209	075366	105712			TSTB	(R2) ;CHECK FIRST BYTE
210	075370	001001			BNE	NCON ;IF ZERO
211	075372	005202			INC	R2 ;INCREMENT TO NEXT BYTE
212	075374	012737	075612	075770	MOV	#PX, PTYPE ;CHANGE TO EXTENDED OUTPUT
213	075402	004737	075772		CALL	OSTRNG ;OUTPUT ACCORDING TO STRING
214	075406			L10035:		
	075406	104423			TRAP	C#MSG
215						

```
1      .SBTTL  GLOBAL SUBROUTINES SECTION
2
3      ;MEMORY ALLOCATION ERROR
4      ;
5      ;THIS ROUTINE PRINTS A SYSTEM FATAL ERROR AND EXITS THE TEST
6
7      075410
8      075410  104454
9      075412  000004
10     075414  000000
11     075416  074324
12
13     3
14     9 075420  104444
15
16     FMERR:
17           TRAP      C$ERSF
18           .WORD     4
19           .WORD     0
20           .WORD     ERRO04
21
22           ;DO CLEAN-UP TRAP
23           TRAP      C$DCLN
```

GLOBAL SUBROUTINES SECTION

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17 075422
18 075426 013746 064412
19 075432 002766
20 075434 060101
21 075436 060137 064412
22 075442 012601
23 075444 000207

;ALOCM
;
;ALLOCATE A BLOCK OF FREE MEMORY. REPORT ERROR IF MEMORY EXHAUSTED.
;
;INPUTS:
; R1 - NUMBER OF WORDS TO ALLOCATE
; FFREE - FIRST FREE WORD IN MEMORY
; FSIZE - SIZE OF FREE MEMORY AVAILABLE IN WORDS
;OUTPUTS:
; R1 - ADDRESS OF FIRST WORD OF ALLOCATED MEMORY
; FFREE - NEW FIRST FREE WORD IN MEMORY
; FSIZE - SIZE OF FREE MEMORY LEFT AFTER ALLOCATION
;SYSTEM FATAL ERROR WILL BE REPORTED IF NOT ENOUGH MEMORY AVAILABLE
;AND ENTIRE PROGRAM WILL BE STOPPED.

ALOCM:  MOV    FFREE, -(SP)           ;;PUSH FFREE ON STACK
        SUB    R1, FSIZE           ;;REDUCE SIZE OF FREE MEMORY
        BLT    FMERR               ;;REPORT ERROR IF NOT ENOUGH MEMORY
        ADD    R1, R1              ;;CHANGE WORDS TO BYTES
        ADD    R1, FFREE           ;;CALCULATE NEW START OF FREE MEMORY
        MOV    (SP)+, R1           ;;POP STACK INTO R1
        RTS    PC

```

GLOBAL SUBROUTINES SECTION

```

1
2      ;HCOMM
3      ;
4      ;ALLOCATES MEMORY FOR HOST COMMUNICATION AREA AND PACKET BUFFERS WITH ONE
5      ;DESCRIPTOR IN EACH RING. TO BE CALLED AFTER INITIALIZING
6      ;A CONTROLLER WITH SA.MSG=0 AND SA.CMD=0.
7      ;
8      ;INPUTS:
9      ;   R5 - ADDRESS OF CONTROLLER TABLE
10     ;OUTPUTS:
11     ;   CONTROLLER TABLE POINTING TO HOST COMMUNICATION AREA,
12     ;   RING POINTERS TO PACKETS,
13     ;   R4 - ADDRESS OF HOST COMMUNICATION AREA
14
15 075446 012701 000146      HCOMM:  MOV     #<HC.SIZ>/2,R1      ;GET SIZE OF AREA TO ALLOCATE
16 075452 004737 075422      JSR     PC,ALOCM                ;ALLOCATE THE MEMORY
17 075456 010104              MOV     R1,R4                    ;GET ADDRESS OF HOST COMM AREA
18 075460 010465 000016      MOV     R4,C.HCOM(R5)           ;PLACE BEGINNING ADRS OF HOST COMM AREA IN THE
19                                ;CONTROLLER TABLE
20 075464 062701 000020      ADD     #HC.MPK,R1              ;COMPUTE START OF MESSAGE PACKET
21 075470 010164 000004      MOV     R1,HC.MSG(R4)           ;PLACE IN RING
22 075474 062701 000000      ADD     #<HC.CPK-HC.MPK>,R1     ;COMPUTE START OF COMMAND PACKET
23 075500 010164 000010      MOV     R1,HC.CMD(R4)          ;PLACE IN RING
24 075504 000207              RTS     PC

```

```

1
2
3
4
5
6
7
8 075506 110037 075662 PRINTC: MOV      RO,TTYOUT
9 075512 010146          MOV      R1,-(SP)
10 075514 012701 065037 MOV      #FRMT,R1
11 075520 120027 000015 CMPB     RO,#CR
12 075524 001002          BNE     1$
13 075526 012701 065042 MOV      #CRLF,R1
14
15 075532 004777 000232 1$:      JSR      PC,#PTYPE
16 075536 012601          MOV      (SP)+,R1
17 075540 000207          RTS      PC
18 075542          PF:
19 075542 012746 075662      MOV      #TTYOUT,-(SP)
20 075546 010146          MOV      R1,-(SP)
21 075550 012746 000002      MOV      #2,-(SP)
22 075554 010600          MOV      SP,RO
23 075556 104417          TRAP    C#PNTF
24 075560 062706 000006      ADD     #6,SP
25 075564 000207          RTS      PC
26 075566          PB:
27 075566 012746 075662      MOV      #TTYOUT,-(SP)
28 075572 010146          MOV      R1,-(SP)
29 075574 012746 000002      MOV      #2,-(SP)
30 075600 010600          MOV      SP,RO
31 075602 104414          TRAP    C#PNTB
32 075604 062706 000006      ADD     #6,SP
33 075610 000207          RTS      PC
34 075612          PX:
35 075612 012746 075662      MOV      #TTYOUT,-(SP)
36 075616 010146          MOV      R1,-(SP)
37 075620 012746 000002      MOV      #2,-(SP)
38 075624 010600          MOV      SP,RO
39 075626 104415          TRAP    C#PNTX
40 075630 062706 000006      ADD     #6,SP
41 075634 000207          RTS      PC
42 075636          PS:
43 075636 012746 075662      MOV      #TTYOUT,-(SP)
44 075642 010146          MOV      R1,-(SP)
45 075644 012746 000002      MOV      #2,-(SP)
46 075650 010600          MOV      SP,RO
47 075652 104416          TRAP    C#PNTS
48 075654 062706 000006      ADD     #6,SP
49 075660 000207          RTS      PC
50
51 075662          TTYOUT: .BYTE 0
52 075663          .BYTE 0
53
54
55
56
57
58
59

```

```

;SAVE CHARACTER FOR TTY OUTPUT
;;PUSH R1 ON STACK
;PICKUP FORMATTED ASCIZ STRING STATEMENT
;IF NOT A CARRIAGE RETURN, THEN
;PRINT SOME OTHER CHARACTER, ELSE
;PICKUP FORMATTED ASCIZ STRING STATEMENT
;GO PRINT CR-LF.
;PRINT THE ASCIZ STRING,
;;POP STACK INTO R1

```

```

;TTY OUTPUT BUFFER
;TERMINATOR FOR ASCIZ STRING

```

```

1
2
3          ;PRINT FORMATTED MESSAGE
4          ;CALL WITH MACRO PNT, PNTF, PNTB, PNTX, OR PNTS
5
6 075664 012737 075542 075770 LPNTF:  MOV    @PF,PTYPE
7 075672 000413                BR      LPNT
8
9 075674 012737 075566 075770 LPNTB:  MOV    @PB,PTYPE
10 075702 000407                BR      LPNT
11
12 075704 012737 075612 075770 LPNTX:  MOV    @PX,PTYPE
13 075712 000403                BR      LPNT
14
15 075714 012737 075636 075770 LPNTS:  MOV    @PS,PTYPE
16
17 075722                LPNT:
18 075722 010246                MOV    R2,-(SP)          ;;PUSH R2 ON STACK
19 075724 010346                MOV    R3,-(SP)          ;;PUSH R3 ON STACK
20 075726 010446                MOV    R4,-(SP)          ;;PUSH R4 ON STACK
21 075730 010546                MOV    R5,-(SP)          ;;PUSH R5 ON STACK
22 075732 012102                MOV    (R1)+,R2          ;;GET ADDRESS OF ASCIZ STRING
23 075734 010604                MOV    SP,R4            ;;COMPUTE ADDRESS OF 1ST ARGUMENT AND
24 075736 062704 000012        ADD    @12,R4            ;;SAVE IT IN R4.
25 075742 010146                MOV    R1,-(SP)          ;;PUSH R1 ON STACK
26 075744 004737 075772        JSR    PC,OSTRNG        ;;PRINT THE FORMATTED MESSAGE
27 075750 012600                MOV    (SP)+,R0          ;;POP STACK INTO R0
28 075752 012605                MOV    (SP)+,R5          ;;POP STACK INTO R5
29 075754 012604                MOV    (SP)+,R4          ;;POP STACK INTO R4
30 075756 012603                MOV    (SP)+,R3          ;;POP STACK INTO R3
31 075760 012602                MOV    (SP)+,R2          ;;POP STACK INTO R2
32 075762 012601                MOV    (SP)+,R1          ;;POP STACK INTO R1
33 075764 062006                ADD    (R0)+,SP          ;;ADJUST STACK POINTER OVER ARGUMENTS
34 075766 000110                JMP    @R0              ;;RETURN
35
36
37 075770 075542                PTYPE:  .WORD    PF          ;;PRINT TYPE

```

```

4
5 ;OSTRNG
6 ;
7 ;OUTPUT A MESSAGE ACCORDING TO A FORMAT STRING
8 ;FORMAT OF THE ASCIZ STRING IS AS FOLLOWS:
9 ;
10 ;CHARACTERS ENCLOSED IN QUOTES ARE TO BE PRINTED AS THEY ARE.
11 ;
12 ;OTHERWISE CODE IS A SINGLE LETTER FOLLOWED BY AN OPTIONAL DECIMAL
13 ;NUMBER:
14 ; On - PRINT OCTAL NUMBER. n REPRESENTS SIZE OF BINARY NUMBER PASSED
15 ; IN PARAMETER IN BITS. MAY BE IN RANGE 1 TO 32. IF n>16, TWO PARAMETER
16 ; WORDS ARE USED, OTHERWISE ONLY ONE WORD. LEADING ZEROS ARE PRINTED.
17 ; n IS ALWAYS SPECIFIED.
18 ; Dn - PRINT UNSIGNED DECIMAL NUMBER FROM n BIT PARAMETER. LEADING ZEROS
19 ; ARE NOT PRINTED. A 16 BIT NUMBER EQUAL TO ZERO WILL PRINT "0".
20 ; Hn - PRINT HEX NUMBER FROM PARAMETER OF n BITS. IF n>16 TWO PARAMETERS
21 ; ARE USED, OTHERWISE ONLY ONE PARAMETER. LEADING ZEROS ARE PRINTED.
22 ; Sn - PRINT n SPACES. n ASSUMED TO BE 1.
23 ; Nn - START NEW LINE (CR-LF SEQUENCE). n ASSUMED TO BE 1.
24 ; An - PRINT n ASCII CHARACTERS FROM PARAMETERS. n ASSUMED TO BE 1.
25 ; n/2 PARAMETER WORDS USED.
26 ; Rn - EXECUTE ROUTINE #n. n MUST BE GIVEN AND DEFINED IN HOST PROGRAM.
27 ;
28 ;A NULL CHARACTER MEANS END OF MESSAGE. A NULL AS FIRST CHARACTER IN STRING
29 ;MUST BE IGNORED.
30 ;
31 ;INPUTS:
32 ; R2 - ADDRESS OF START OF FORMAT STRING
33 ; R4 - ADDRESS OF PARAMETERS
34 ;OUTPUTS:
35 ; R2 AND R4 UPDATED TO END OF STRING AND PARAMETERS
39
40 075772 112201 OSTRNG: MOVB (R2)+,R1 ;SEE IF TERMINATOR IN ASCIZ STRING,
41 075774 001421 BEQ OSTRE ;EXIT
42 075776 012700 076304 MOV #ERRC,R0 ;GET POINTER TO CHARACTER TABLE
43 076002 120110 NCUNS: CMPB R1,(R0) ;COMPARE CHARACTER WITH TABLE ENTRY
44 076004 001407 BEQ NCONF ;BRANCH IF MATCH FOUND
45 076006 105720 TSTB (R0)+ ;INCREMENT POINTER
46 076010 001374 BNE NCONS ;CONTINUE SEARCH IF NOT END OF TABLE
47 076012 004137 075664 JSR R1,LPNTF ;CALL LPNTF PRINT ROUTINE
48 076016 065101 .WORD ERRME1 ;ADDRESS OF ASCIZ STRING
49 076020 000000 .WORD ARG.CT ;ARGUMENT COUNT * 2
50 076022 000406 BR OSTRE
51 076024 162700 076304 NCONF: SUB #ERRC,R0 ;GET INCREMENT INTO TABLE
52 076030 006300 ASL R0 ;DOUBLE TO WORD COUNT
53 076032 004770 076316 JSR PC,#ERRD(R0) ;DISPATCH TO PRINT ROUTINE
54 076036 000755 BR OSTRNG ;GET NEXT
55 076040 000207 OSTRE: RTS PC

```

```

1
2
3           ;CONTROL CHARACTER WAS A QUOTE, SO PRINT ALL CHARACTERS TO
4           ;THE NEXT QUOTE.
5 076042 112200
6 076044 120027 000042 CON.QU: MOVB (R2)+,R0 ;GET CHARACTER
7 076050 001403 CMPB RO,#'" ;CHECK IF ENDING QUOTE
8 076052 004737 075506 BEQ CON.QX ;IF SO, GO GET NEXT CONTROL CHARACTER
9 076056 000771 JSR PC,PRINTC ;PRINT THE CHARACTER.
10 076060 000207 BR CON.QU ;CONTINUE PRINTING
11
12           ;CONTROL CHARACTER WAS AN 'A', SO PRINT ASCII CHARACTERS FROM
13           ;PARAMETERS.
14
15 076062 004737 102364 CON.A: JSR PC,GETCNT ;GET COUNT OF CHARACTERS
16 076066 112400 CON.A1: MOVB (R4)+,R0 ;STORE (R4)+ IN R0 AND
17 076070 004737 075506 JSR PC,PRINTC ;PRINT THE CHARACTER.
18 076074 005301 DEC R1 ;COUNT THE CHARACTERS
19 076076 001373 BNE CON.A1 ;PRINT UNTIL COUNT REACHES ZERO
20 076100 032704 000001 BIT #1,R4 ;CHECK IF R4 NOW ODD
21 076106 005204 BEQ CON.A2
22 076110 000207 INC R4 ;IF SO, INCREMENT TO NEXT EVEN ADDRESS
23           ;NOW GET NEXT CONTROL CHARACTER
24
25           ;CONTROL CHARACTER WAS A 'D', SO PRINT A DECIMAL NUMBER.
26 076112 012701 000012 CON.D: MOV #10.,R1 ;LOAD RADIX
27 076116 004737 102442 JSR PC,PNTNUM ;PRINT NUMBER
28 076122 000207 RTS PC ;NOW GET NEXT CONTROL CHARACTER
29
30           ;CONTROL CHARACTER WAS AN 'H', SO PRINT A HEX NUMBER.
31
32 076124 012701 000020 CON.H: MOV #16.,R1 ;LOAD RADIX
33 076130 004737 102442 JSR PC,PNTNUM ;PRINT NUMBER
34 076134 000207 RTS PC ;NOW GET NEXT CONTROL CHARACTER

```

```

1
2
3           ;CONTROL CHARACTER WAS AN 'O', SO PRINT AN OCTAL NUMBER.
4 076136 012701 000010      CON.O:  MOV    #8, R1           ;LOAD RADIX
5 076142 004737 102442      JSR    PC, PNTNUM        ;PRINT NUMBER
6 076146 000207              RTS    PC                 ;NOW GET NEXT CONTROL CHARACTER
7
8           ;CONTROL CHARACTER WAS AN 'N', SO PRINT A CARRIAGE RETURN-LINE FEED.
9
10 076150 004737 102364     CON.N:  JSR    PC, GETCNT        ;GET COUNT
11 076154              CON.N1:  MOVB   #CR, R0           ;STORE #CR IN R0 AND
12 076160 112700 000015      JSR    PC, PRINTC        ;PRINT THE CHARACTER.
13 076164 004737 075506      DEC    R1                ;COUNT THE SEQUENCES
14 076166 005301              BNE    CON.N1
15 076170 000207              RTS    PC                 ;NOW GET NEXT CONTROL CHARACTER
16
17           ;CONTROL CHARACTER WAS AN 'R', SO CALL ONE OF THE PRE-PROGRAMMED
18           ;ROUTINE.
19 076172 004737 102364     CON.R:  JSR    PC, GETCNT        ;GET ROUTINE NUMBER
20 076176 020127 000011      CMP    R1, #ERR.SZ      ;CHECK IF DEFINED ROUTINE NUMBER
21 076202 101004              BHI    CON.R1
22 076204 060101              PDD    P1, R1           ;DOUBLE COUNT TO GET WORD INDEX
23 076206 004771 076250      JSR    PC, #ERR.TB-2(R1) ;CALL ROUTINE
24 076212 000207              RTS    PC                 ;NOW GET NEXT CONTROL CHARACTER
25
26 076214              CON.R1:  JSR    R1, LPNTF          ;CALL LPNTF PRINT ROUTINE
27 076220 004137 075664      .WORD  ERRME1          ;ADDRESS OF ASCIZ STRING
28 076222 065101              .WORD  ARG.CT          ;ARGUMENT COUNT * 2
29 076224 012601              MOV    (SP)+, R1        ;POP STACK INTO R1
30 076226 000207              RTS    PC
31
32           ;CONTROL CHARACTER WAS AN 'S', SO PRINT SOME NUMBER OF SPACES.
33 076230 004737 102364     CON.S:  JSR    PC, GETCNT        ;GET COUNT
34 076234              CON.S1:  MOVB   #' , R0           ;STORE #' IN R0 AND
35 076240 112700 000040      JSR    PC, PRINTC        ;PRINT THE CHARACTER.
36 076244 004737 075506      DEC    R1                ;COUNT THE SPACES
37 076246 005301              BNE    CON.S1
38 076250 000207              RTS    PC                 ;NOW GET NEXT CONTROL CHARACTER

```

```

1
2
3
4 076252 101726
5 076254 101754
6 076256 102052
7 076260 102066
8 076262 102142
9 076264 102220
10 076266 102234
11 076270 102252
12 076272 102270
13
14          000011
15
16 076274
18 076274 066670
19 076276 066714
20 076300 066734
23 076302 066754
25

```

```

;PRE-PROGRAMMED ERROR ROUTINE DISPATCH TABLE

ERR.TB: .WORD CALR1          ;CALL ALTERNATE PRINT STRING IN DM MEMORY IMAGE
        .WORD CALR2          ;PRINT AN SDI DIAGNOSE RESPONSE
        .WORD CALR3          ;DECIDE WHETHER TO PRINT RBN
        .WORD CALR4          ;PRINT BASIC LINE WITHOUT JDA ADDRESS
        .WORD CALR5          ;PRINT BASIC LINE WITH UDA ADDRESS
        .WORD CALR6          ;CALL ALTERNATE PRINT STRING IN PDP-11 MEMORY
        .WORD CALR7          ;PRINT "REPLACE UDA MODULE M7161"
        .WORD CALR8          ;PRINT " UDASA CONTAINS XXXXXX"
        .WORD CALR9          ;REPRINT LAST NUMBER

ERR.SZ  = <.-ERR.TB>/2

TNAMEs:
        .WORD BASN1
        .WORD BASN2
        .WORD BASN3
        .WORD BASN4

```

```

1      ;BUILD TWO TABLES
2      ;      FIRST CONTAINING CONTROL CHARACTERS
3      ;      SECOND CONTAINING ROUTINE ADDRESSES
4
5      ;HERE IS FIRST TABLE
6
16     076304      042      ERRC:      .BYTE      ' "
      076304      101      .BYTE      'A
      076305      101      .BYTE      'A
      076306      104      .BYTE      'D
      076307      110      .BYTE      'H
      076310      117      .BYTE      'O
      076311      116      .BYTE      'N
      076312      122      .BYTE      'R
      076313      123      .BYTE      'S
17     076314      000      .BYTE      0      ;FOLLOW WITH A NULL BYTE
18
19
20
21
31     076316      076042      ERRD:      .WORD      CON.QU
      076316      076062      .WORD      CON.A
      076320      076062      .WORD      CON.A
      076322      076112      .WORD      CON.D
      076324      076124      .WORD      CON.H
      076326      076136      .WORD      CON.O
      076330      076150      .WORD      CON.N
      076332      076172      .WORD      CON.R
      076334      076230      .WORD      CON.S

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 076336 010137 064444
15 076342 004737 106260
16 076346 005037 064650
17 076352 013737 064416 064412
18 076360 013737 064420 064414
20 076366 022701 000004
21 076372 001413
22 076374 020137 064442
23 076400 001410
24 076402 012705 001122
25 076406 012737 002122 064432
26 076414 004737 105630
27 076420 103401
28 076422 000207
29
30 076424
    076424 104454
    076426 000007
    076430 000000
    076432 074350
31
32 076434 104444
41
    ;TINIT
    ;
    ;INITIALIZE VARIABLES FOR TEST
    ;
    ;INPUTS:
    ; R1 - TEST NUMBER
    ;OUTPUTS:
    ; LBUFS - CLEARED (DELETES ERROR LOG)
    ; TNUM - TEST NUMBER FROM R1
    ; FNUM - LAST LOADED TEST IN TNUM < 4
    ; ALL REGISTERS CLOBERED
    ;
    ;INIT:  MOV     R1,TNUM           ;SAVE TEST NUMBER
    ;      CALL    RESET           ;RESET ALL UDA'S
    ;      CLR     LBUFS           ;CLEAR ERROR LOG BUFFER POINTER
    ;      MOV     FMEM,FFREE       ;INIT FREE
    ;      MOV     FMEM,FSIZE       ;INIT FSIZE
    ;      CMP     #4,R1           ;ARE WE DOING TEST 4 ?
    ;      BEQ     TIEXIT          ; IF SO, EXIT
    ;      CMP     R1,FNUM         ; IF FILE ALREADY IN MEMORY?
    ;      BEQ     TIEXIT          ; IF SO, EXIT
    ;      MOV     @(<STORAG-DMFRST>,R5),R5 ; R5->ADDRESS TO STORE - DM FIRST ADDRESS
    ;      MOV     @STORAG,DMPROG   ; SAVE DMPROG ADDRESS
    ;      CALL    RDREC           ; READ IN RECORD
    ;      BCS     TINITE          ; IF ERROR, REPORT
    ;
    ;TIEXIT: RETURN
    ;
    ;TINITE:
    ;      TRAP   C1ERSF           ;DO CLEAN-UP TRAP
    ;      .WORD 7
    ;      .WORD 0
    ;      .WORD ERROO7
    ;
    ;      TRAP   C1DCLN
    
```

GLOBAL SUBROUTINES SECTION

```

3
4
5
6
7
8
9
10
11
12
13
14
15
16 076436 005037 064430
17 076442 005037 064432
18 076446 010137 064446
19 076452 012737 002122 064432
20 076460 013737 064446 064452
21 076466 013705 064430
37 076472 004737 076500
38 076476 000421

;RNT4DM
;
;LOAD AND RUN A TEST 4 IN THE CONTROLLERS. RETURN WHEN ALL
;DM PROGRAMS HAVE TERMINATED.
;
;INPUTS:
;   TSTTAB - POINTER TO FIRST CONTROLLER TABLE
;   R1 - NUMBER OF CONTROLLERS TO TEST
;OUTPUTS:
;   DMPROG - POINTER TO START OF DM PROGRAM IN MEMORY
;   Z SET IF NO CONTROLLERS SUCCESSFULLY STARTED
;ALL REGISTERS ARE USED AND PREVIOUS CONTENTS DESTROYED.

RNT4DM: CLR   URNING
        CLR   TYPCNT
        MOV   R1,URUN
        MOV   @STORAG,DMPROG
        MOV   URUN,UCNT
        MOV   TSTTAB,R5
        CALL  GTT452
        BR    STLDOM
; CLEAR FLAGS
; URUN = # OF UNITS
; DMPROG -> WHERE EITHER TEST 4 IS LOADED
; R5 -> CONTROLLER TABLE
; GET TEST 4 FOR U5?
; GO START LOADING DM PROGRAMS

```

1						
2				IGTT452		
3						
4				GET TEST 4 FOR UDA52		
5						
6	076500	0127	000005	GTT452: MOV	05.,R1	R1 = T4 FOR 52 FNUM
7	076504	01270	001122	MOV	0<STORAG-DMFRST>,R5	
8	076510	02013	064442	CMP	R1,FNUM	DMPROG ALREADY IN MEMORY?
9	076514	00140		BEQ	1\$	IF SO, EXIT
10	076516	00473	105630	CALL	RDREC	ELSE, READ RECORD.
11	076522	103002		BCC	1\$	
12	076524	000137	076424	JMP	TINITE	BRANCH IF ERROR
13	076530	000207		1\$:	RETURN	

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16 076532 010137 064446
17 076536 005037 064450
18
19
20
21 076542 013737 064446 064452
22 076550 013705 064430
23 076554
27 076554 005065 000014
29 076560 116537 000002 002074
30 076566 005765 000002
31 076572 100405
33 076574 004737 103410
34 076600 001402
35 076602 005237 064450
36 076606 062705 000046
37 076612 005337 064452
38 076616 001356
39
40
41
42 076620 005737 064450
43
44
45
46
47 076624 000207

```

```

;RUNDM
;
;LOAD AND RUN A DM PROGRAM IN THE CONTROLLERS. RETURN WHEN ALL
;DM PROGRAMS HAVE TERMINATED.
;
;INPUTS:
;   TSTTAB - POINTER TO FIRST CONTROLLER TABLE
;   R1 - NUMBER OF CONTROLLERS TO TEST
;IMPLICIT INPUTS:
;   DMPCG - POINTER TO START OF DM PROGRAM IN MEMORY
;OUTPUTS:
;   Z SET IF NO CONTROLLERS SUCCESSFULLY STARTED
;ALL REGISTERS ARE USED AND PREVIOUS CONTENTS DESTROYED.

RUNDM:  MOV     R1,URUN           ;SAVE NUMBER OF UNITS TO RUN
        CLR     URNING          ;CLEAR NUMBER OF UNITS RUNNING

;LOAD DM PROGRAM INTO EACH CONTROLLER

STLDDH: MOV     URUN,UCNT        ;SET COUNTER OF UNITS
        MOV     TSTTAB,R5       ;GET FIRST CONTROLLER TABLE

LD0M:   CLR     C.FLG(R5)       ;CLEAR ALL FLAGS
        MOVB   C.UNIT(R5),L$LUN ;SEE IF UNIT TO BE TESTED
        TST    C.UNIT(R5)
        BMI    LDNEXT          ;IF NOT, DON'T LOAD THIS UNIT
        CALL  LOADDM           ;LOAD THE DM PROGRAM
        BEQ    LDNEXT          ;IF ERROR, GO TO NEXT CONTROLLER
        INC    URNING          ;IF NO ERROR, COUNT UNIT RUNNING
LDNEXT: ADD    @C.SIZE,R5       ;MOVE TO NEXT CONTROLLER TABLE
        DEC    UCNT            ;CHECK IF MORE CONTROLLERS
        BNE    LD0M           ;LOAD NEXT

;CHECK IF ANY CONTROLLERS LOADED

        TST    URNING          ;ANY UNITS LOADED?

;THE DM PROGRAMS ARE NOW IN CONTROL
;RESPDM MUST BE CALLED TO RESPOND TO THEIR REQUESTS

        RETURN

```



```

1
2
3           ;CHECK FOR TIME TO PRINT STATISTICAL REPORT
4 076770 005737 064616 RSPNXT: TST     KW.CSR           ;ANY CLOCK ON SYSTEM?
5 076774 001412          BEQ     RSPNRP           ;BYPASS IF NOT
6 076776 023737 064630 064634  CMP     KW.EL+2,STIME+2       ; A STATISTICAL REPORT
7 077004 101005          BHI RSPRPT:
8 077006 001005          BNE     RSPNRP
9 077010 023737 064626 064632  CMP     KW.EL,STIME
10 077016 103401         BLO RSPNRP
11 077020          RSPRPT:           ;PRINT A STATISTICAL REPORT
12 077020 104424         TRAP     C:DRPT
13
14           ;SWITCH TO NEXT CONTROLLER
15
16 077022 062705 000046 RSPNRP: ADD     #C.SIZE,R5           ;MOVE TO NEXT TABLE
17 077026 005337 064452  DEC     UCNT           ;CHECK IF MORE CONTROLLERS
18 077032 001302          BNE     RESPCT       ;LOOK AT NEXT CONTROLLER
19 077034 000674          BR      RESPDM       ;LOOK AT FIRST CONTROLLER AGAIN
20
21           ;REMOVE A CONTROLLER FROM TESTING
22
23 077036 042765 000012 000014 RSPDRP: BIC     #CT.RN+CT.MSG,C.FLG(R5) ;CLEAR PROGRAM RUNNING
24 077044 005337 064450  DEC     URNING           ;REDUCE RUNNING CONTROLLERS COUNT
25 077050 001347          BNE     RSPNXT       ;IF ANY STILL RUNNING, LOOK AT THEM
26 077052 000207          RETURN          ;ELSE RETURN TO TEST SECTION

```

```

1
2
3           ;CONTROLLER HAS RESPONDED, LOOK AT MESSAGE PACKET
4
5           ;CHECK FOR PROPER OPCODE IN END PACKET
6 077054 012700 000204          RSPIN: MOV     @OP.END+OP.SSD,R0          ;GET SEND DATA END PACKET OPCODE
7 077060 032765 000020 000014  BIT     @CT.REQ,C.FLG(R5)          ;LOOK IF SEND DATA OR RECEIVE DATA
8 077066 001402                  BEQ     RSPMWR
9 077070 012700 000205          MOV     @OP.END+OP.RSD,R0          ;CHANGE TO RECEIVE DATA END PACKET OPCODE
10 077074 120064 000030        RSPMWR: CMPB   RO,HC.MPK+P.OPCD(R4) ;COMPARE TO OPCODE IN END PACKET
11 077100 001010                  BNE     RSPERR
12
13           ;LOOK AT STATUS CODE
14
15 077102 032764 000037 000032  BIT     @ST.MSK,HC.MPK+P.STS(R4) ;CHECK FOR STATUS CODE ST.SUC (ZERO)
16 077110 001004                  BNE     RSPERR
17
18           ;CHECK FOR EXPECTED REFERENCE NUMBER
19
20 077112 026564 000044 000020  CMP     C.REF(R5),HC.MPK+P.CRF(R4) ;CHECK IF CORRECT REF NUMBER
21 077120 001405                  BEQ     RSPPTW
22 077122                  RSPERR: TRAP   C$ERDF
23 077122 104455                  .WORD  33
24 077124 000041                  .WORD  0
25 077126 000000                  .WORD  ERR033
26 077130 074764                  BR     RSPDRP          ;DROP UNIT FROM TESTING
27 077132 000741
28
29           ;CHECK IF RESPONSE FROM SEND OR RECEIVE DATA COMMAND
30
31 077134 032765 000020 000014  RSPPTW: BIT    @CT.REQ,C.FLG(R5)          ;CHECK IF RESPONSE FROM DM PROGRAM
32 077142 001445                  RSPOU: BEQ    RSPOUT          ;LOOK AT REQUEST NUMBER IF SO

```

```

1
2
3      ;MAINTENANCE READ END PACKET RECEIVED, LOOK AT REQUEST FROM DM PROGRAM
4 077144 016401 000206      RSPPT2: MOV      HC.BF2(R4),R1      ;GET REQUEST NUMBER
5 077150 042701 007777      BIC      #007777,R1      ;CHECK TYPE
6 077154 022701 060000      CMP      #DU.SPC,R1      ;IS SPECIAL TYPE SET?
7 077160 00101C      BNE      1$      ;IF NOT, ERROR
8 077162 042764 170000 000206      BIC      #+C007777,HC.BF2(R4)      ;CLEAR TYPE
9 077170 016401 000206      MOV      HC.BF2(R4),R1      ;GET REQUEST NUMBER
10 077174 020127 000017      CMP      R1,#DSPSIZ      ;CHECK IF IN EXPECTED RANGE
11 077200 103405
12 077202
13 077212 000711      1$:      TRAP      C#ERDF      ;
14      .WORD      32      ;
15      .WORD      0      ;
16      .WORD      EWR032      ;
17      BR      RSPDRP      ;DROP UNIT FROM TESTING
18
19 077214 012700 000004      RSPPT3: MOV      #OP.SSD,R0      ;BUILD A SEND DATA COMMAND PACKET
20 077220 004737 104124      CALL     BLDCMD      ; FOR ANSWER TO DM PROGRAM
21 077224 012700 000100      MOV      #HC.BF1,R0      ;POINT TO BUFFER IN PACKET
22 077230 004737 104256      CALL     CLRBUF      ; AND CLEAR BUFFER
23 077234 010403      MOV      R4,R3      ;R3 POINTS TO COMMAND BUFFER
24 077236 062704 000106      ADD      #HC.BSZ,R4      ;R4 POINTS TO MESSAGE BUFFER
25 077242 011401      MOV      (R4),R1      ;GET REQUEST NUMBER
26 077244 012423      MOV      (R4),R3      ;PUT REQUEST NUMBER INTO COMMAND PACKET
27 077246 060101      ADD      R1,R1      ;DOUBLE REQUEST NUMBER
28 077250 004771 077360      CALL     #RSPDSP(R1)      ;CALL REQUESTED ROUTINE
29 077254 001270      BNE      RSPDRP      ;ROUTINE RETURNS Z CLEAR TO DROP UNIT FROM TESTING
30      ; Z SET IF COMMAND READY TO SEND TO UNIT
31
32      ;SEND COMMAND BACK TO UDA
33
34 077256 042765 000010 000014      RSPOUT: BIC      #CT.MSG,C.FLG(R5)      ;CLEAR MESSAGE RECEIVED FLAG
35 077264 032765 000020 000014      BIT      #CT.REQ,C.FLG(R5)      ;CHECK WHICH COMMAND TO SEND
36 077272 001014      BNE      RSPOU2      ;BRANCH IF RESPONSE TO REQUEST
37
38 077274 012700 000005      MOV      #OP.RSD,R0      ;BUILD RECEIVE DATA COMMAND
39 077300 004737 104124      CALL     BLDCMD
40 077304 012700 000206      MOV      #HC.BF2,R0      ;POINT TO MESSAGE BUFFER
41 077310 004737 104256      CALL     CLRBUF      ; AND CLEAR IT
42 077314 052765 000020 000014      BIS      #CT.REQ,C.FLG(R5)      ;SET REQUEST BIT
43 077322 000403      BR      RSPOU3
44
45 077324 042765 000020 000014      RSPOU2: BIC      #CT.REQ,C.FLG(R5)      ;CLEAR REQUEST BIT
46 077332
47 077332 004737 104210      RSPOU3: CALL     SNDCMD      ;SEND COMMAND TO UDA
48 077336 012700 000264      MOV      #3,*60.,R0      ;SET TIMEOUT FOR 3 MINUTES
49 077342 010501      MOV      R5,R1
50 077344 062701 000040      ADD      #C.TO,R1      ;PUT TIME IN CONTROLLER TABLE
51 077350 004737 104530      CALL     SETTO
52 077354 000137 076770      JMP      RSPNXT      ;NOW WAIT FOR END PACKET

```

```

1
2
3
4 077360 077416
5 077362 077536
6 077364 077702
7 077366 100352
8 077370 100374
9 077372 100654
10 077374 100704
11 077376 100734
12 077400 100762
13 077402 101002
14 077404 101144
15 077406 101250
16 077410 101470
17 077412 101610
18 077414 101722
19
20 000017

;RESPONSE REQUEST DISPATCH TABLE

RSPDSP: .WORD T1MSIZ
        .WORD T2OLL
        .WORD T2CMD
        .WORD T4MPRM
        .WORD T4UPRM
        .WORD T4BB1
        .WORD T4BB2
        .WORD T4SOFT
        .WORD T4SEEK
        .WORD T4MXFR
        .WORD UTOTST
        .WORD ERRMES
        .WORD ERRMC
        .WORD MESSAG
        .WORD DONE

; 0. SET UP FREE MEMORY FOR ADDRESS TESTING
; 1. PROVIDE DIAGNOSTIC PROGRAM FOR DISK DRIVE
; 2. GET MANUAL INTERVENTION COMMAND
; 3. TELL DATA PATTERN 16.
; 4. TELL UNIT PARAMETERS, CLEAR CONTENTS
; 5. TELL BAD BLOCKS (FIRST 14)
; 6. TELL BAD BLOCKS (LAST TWO)
; 7. ADD TO SOFT ERROR AND ECC COUNTS
; 8. ADD 1000 TO SEEK COUNT
; 9. ADD TO MEGABITS READ AND WRITE COUNTS
;10. TELL WHICH DRIVES TO TEST
;11. REPORT ERROR MESSAGE
;12. REPORT ERROR MESSAGE AND COUNT HARD ERROR
;13. PRINT A DESCRIPTIVE MESSAGE
;14. MARK DM PROGRAM AS NO LONGER RUNNING

DSPSIZ = <.-RSPDSP>/2

;LEGAL NUMBERS ARE LOWER THAN THIS
    
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39

;NORMAL MAINTENANCE READ BUFFER DESCRIPTION

;BYTE OFFSET FROM  
;START OF BUFFER

0	REQUEST NUMBER
2	DATA ARGUMENT #1
4	DATA ARGUMENT #2
6	DATA ARGUMENT #3
8	DATA ARGUMENT #4
10	DATA ARGUMENT #5
12	DATA ARGUMENT #6
14	DATA ARGUMENT #7
16	DATA ARGUMENT #8
18	DATA ARGUMENT #9
20	DATA ARGUMENT #10
22	DATA ARGUMENT #11
.	.
.	.
.	.
68	DATA ARGUMENT #34

USED TO SELECT ROUTINE  
 R4 CONTAINS THIS ADDRESS

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39

;NORMAL PSEUDO-TERMINAL IN PACKET DESCRIPTION GIVEN IN RESPONSE TO ABOVE PACKET

;BYTE OFFSET FROM  
;START OF PACKET

0	REQUEST NUMBER
2	DATA ARGUMENT #1
4	DATA ARGUMENT #2
6	DATA ARGUMENT #3
8	DATA ARGUMENT #4
10	DATA ARGUMENT #5
12	DATA ARGUMENT #6
14	DATA ARGUMENT #7
16	DATA ARGUMENT #8
18	DATA ARGUMENT #9
20	DATA ARGUMENT #10
22	DATA ARGUMENT #11
.	.
.	.
.	.
68	DATA ARGUMENT #34

ECHOED FROM REQUEST PACKET

R3 CONTAINS THIS ADDRESS

ALL DATA ARGUMENTS ARE RETURNED  
CONTAINING ZEROS UNLESS  
SPECIFICALLY INDICATED BY  
RESPONSE ROUTINE.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 077416
29 077416 013701 064412
30 077422 013702 064414
31
32
33
34 077426 010111
35 077430 062701 000002
36 077434 005302
37 077436 001373
38
39
40
41 077440 013723 064412
42 077444 005023
43 077446 013700 064414
44 077452 006300
45 077454 063700 064412
46 077460 162700 000002
47 077464 010023
48 077466 005023
49
50
51
52 077470 005023
53 077472 005023
54 077474 013700 002120
55 077500 005001
56 077502 006300
57 077504 006300
58 077506 006300

;TIMSIZ - DM REQUEST 0
;
;SET UP MEMORY FOR ADDRESS TESTING FROM UDA.
;PLACE ADDRESS OF EACH LOCATION INTO EACH LOCATION IN FREE
;MEMORY. RETURN FIRST LOCATION OF FREE MEMORY IN CMD.02 (LOW BITS)
;AND CMD.03 (HIGH BITS). RETURN LAST LOCATION OF FREE MEMORY IN
;CMD.04 AND CMD.05. ALSO RETURN FIRST EXISTANT LOCATION IN CMD.06
;AND CMD.07; LAST EXISTANT LOCATION IN CMD.08 AND CMD.09.
;
;INPUTS:
; R5 - CONTROLLER TABLE ADDRESS
; R4 - MESSAGE PACKET DATA ADDRESS (POINTING TO MSG.02)
; R3 - COMMAND PACKET DATA ADDRESS (POINTING TO CMD.02)
;OUTPUTS:
; COMMAND PACKET CONTAINING:
; (R3) LOW ADDRESS BITS OF FIRST WRITABLE ADDRESS
; 2.(R3) HIGH ADDRESS BITS OF FIRST WRITABLE ADDRESS
; 4.(R3) LOW ADDRESS BITS OF LAST WRITABLE ADDRESS
; 6.(R3) HIGH ADDRESS BITS OF LAST WRITABLE ADDRESS
; 8.(R3) LOW ADDRESS BITS OF FIRST READABLE ADDRESS
; 10.(R3) HIGH ADDRESS BITS OF FIRST READABLE ADDRESS
; 12.(R3) LOW ADDRESS BITS OF LAST READABLE ADDRESS
; 14.(R3) HIGH ADDRESS BITS OF LAST READABLE ADDRESS
; Z SET

TIMSIZ:
MOV FFREE,R1 ;GET FIRST ADDRESS OF FREE MEMORY
MOV FSIZE,R2 ;GET SIZE
;FILL MEMORY WITH ADDRESS PATTERN

MEMFIL: MOV R1,(R1) ;WRITE DATA INTO LOCATION
ADD #2,R1 ;INCREASE ADDRESS TO NEXT LOCATION
DEC R2 ;COUNT THE WORDS
BNE MEMFIL ;FILL ALL WORDS

;SEND LOCATION OF FREE MEMORY TO UDA

MOV FFREE,(R3)+ ;LOAD FIRST ADDRESS OF FREE MEMORY
CLR (R3)+ ;HIGH ORDER BITS ARE ZERO
MOV FSIZE,R0 ;GET SIZE OF FREE MEMORY
ASL R0 ;CONVERT TO BYTES
ADD FFREE,R0 ;COMPUTE LAST LOCATION
SUB #2,R0
MOV R0,(R3)+ ;LOAD LAST LOCATION
CLR (R3)+ ;CLEAR HIGH ORDER BITS

;SEND LOCATION OF READABLE MEMORY

CLR (R3)+ ;SEND ZERO AS START OF READABLE MEMORY
CLR (R3)+
MOV L#HIMEM,R0 ;GET HIGH MEMORY ADDRESS
CLR R1 ;CLEAR HIGH BITS
ASL R0 ;SHIFT LEFT 6 PLACES
ASL R0
ASL R0

```

59	077510	006300		ASL	R0	
60	077512	006300		ASL	R0	
61	077514	006101		ROL	R1	
62	077516	006300		ASL	R0	
63	077520	006101		ROL	R1	
64	077522	052700	000076	BIS	#76,R0	;SET LOW ORDER BITS
65	077526	010023		MOV	R0,(R3)+	;PUT INTO BUFFER
66	077530	010123		MOV	R1,(R3)+	
68	077532	000264		SEZ		
69	077534	000207		RETURN		

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38 077536
39 077536 005037 064656
40 077542 012437 064660
41 077546 012437 064662
42 077552 012437 064664
43 077556 012437 064674
44 077562 012437 064676
45
46
47
48
49 077566 005075 000000
50 077572 013737 064412 064666
51 077600 005037 064670
52 077604 013737 064414 064672
53 077612 004737 105570
54 077616 103002
55 077620 005237 064656
56 077624 005237 064656
57 077630 013737 064672 064414
58 077636 013737 064412 064672

;T2DLL - DM REQUEST 1
;
;PROVIDE DIAGNOSTIC TO DOWNLINE LOAD INTO DISK DRIVE.
;
;THE UDA MAY BE USED TO GET THE DIAGNOSTIC IF THE SYSTEM LOAD DEVICE
;IS ON THE UDA. THIS ACTION WILL CAUSE A REINITIALIZATION OF THE UDA
;AND THE RING STRUCTURE MOVED. SINCE THIS PROGRAM HAS NO WAY TO
;DETERMINE IF THE UDA IS USED, IT WILL ALWAYS ASSUME IT IS USED AND
;WILL INITIALIZE AND RELOAD THE DM PROGRAM AFTER READING THE
;DIAGNOSTIC. THE OUTPUTS OF THIS ROUTINE ARE STORED AND SENT TO THE
;DM PROGRAM IN THE UTOTST REQUEST.
;
;INPUTS:
; R5 - CONTROLLER TABLE ADDRESS
; R4 - MESSAGE DATA ADDRESS
; (R4) DRIVE NUMBER
; 2.(R4) A VALUE THE DM PROGRAM WISHES RETURNED
; 4.(R4) REGION TO WHICH PROGRAM IS TO BE LOADED IN DISK
; 6.(R4) 2 WORD PROGRAM NAME IN RAD50
; R3 - COMMAND DATA ADDRESS
;
;OUTPUTS:
; COMMAND PACKET COULD CONTAIN THE FOLLOWING:
; (R3) ONE IF PROGRAM PROVIDED, TWO IF PROGRAM NOT AVAILABLE
; 2.(R3) DRIVE NUMBER
; 4.(R3) COPY OF THE VALUE FROM DM PROGRAM
; 6.(R3) REGION TO WHICH PROGRAM IS TO BE LOADED
; 8.(R3) ADDRESS OF FIRST BYTE TO BE DOWNLINE LOADED
; 10.(R3) HIGH ORDER BITS OF ADDRESS
; 12.(R3) BYTE COUNT OF PROGRAM TO BE DOWNLINE LOADED
; Z SET
; THIS PROGRAM WILL NOT SEND A COMMAND PACKET IN RESPONSE TO THIS REQUEST.
; THE UDA WILL BE REINITIALIZED AND THE DM PROGRAM RELOADED. THEN THIS DATA
; WILL BE APPENDED TO THE NEXT UTOTST REQUEST.
;
;COPY REQUEST DATA TO STORAGE

T2DLL:
    CLR     DLL
    MOV     (R4),DLLDR
    MOV     (R4),DLLV
    MOV     (R4),DLLR
    MOV     (R4),DLLNAM
    MOV     (R4),DLLNAM.2
; CLEAR CONTROL WORD
; DRIVE NUMBER
; VALUE FROM DM
; REGION
; PROGRAM NAME
; (TWO WORDS)

;RESET UDA AND READ DM PROGRAM

    CLR     R(R5)
    MOV     FFREE,DLLADR
    CLR     DLLADR.2
    MOV     FSIZE,DLLSIZ
    CALL   RDDLL
    BCC    11
    INC     DLL
    INC     DLL
    MOV     DLLSIZ,FSIZE
    MOV     FFREE,DLLSIZ
; RESET THE UDA
; GET ADDRESS WHERE PROGRAM
; TO BE STORED
; SAVE CURRENT SIZE OF MEMORY
; READ DLL PROGRAM FROM DATA FILE
; PROGRAM NOT FOUND IF CARRY SET
; RETURN 1 IF PROGRAM FOUND
; RETURN 2 IF PROGRAM NOT FOUND
; COMPUTE SIZE OF DLL PROGRAM
; AND RESTORE ORIGINAL FFREE

```

## GLOBAL SUBROUTINES SECTION

59	077644	163737	064666	064672	SUB	DLLADR,DLLSIZ	
60	077652	013737	064666	064412	MOV	DLLADR,FFREE	; AND %SIZE VALUES
61	077660	005726			TST	(SP)	
62	077662	012701	000001		MOV	#1,R1	;POP RETURN ADDRESS OFF STACK
63	077666	004737	076532		CALL	RUNDM	;RUN THE DM PROGRAM AGAIN
64	077672	001402			BEQ	24	
65	077674	000137	076626		JMP	RESPDM	
66	077700	000207		241	RETURN		

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29 077702
30 077702 032737 000200 064400
31 077710 001002
32 077712 000137 100334
33 077716
34 077716 104450
35 077720 103406
36 077722
37 077722 004137 075664
38 077726 065757
39 077730 000000
40 077732 000137 100334
41 077736 012401
42 077740 012402
43 077742 001022
44 077744 004737 102274
45 077750 001401
46 077752 000207
47 077754
48 077754 011446
49 077756 011546
50 077760 016446 000002
51 077764 004137 075664
52 077770 066056
53 077772 000006
54 077774 005037 064642
55 100000 005037 064644
56 100004 005037 064646

```

```

;T2CMD - DM REQUEST 2
;GET MANUAL INTERVENTION COMMAND
;INPUTS:
; R5 - CONTROLLER TABLE ADDRESS
; R4 - MESSAGE DATA ADDRESS
; (R4) DRIVE NUMBER
; 2.(R4) OPERATION CODE
; 0 ON FIRST REQUEST FOR DRIVE, ECHO OF PREVIOUS RESPONSE ALL OTHER TIMES.
; IF OPERATION CODE = 2
; 4.(R4) DATA BYTE READ (TO BE PRINTED)
; R3 - COMMAND DATA ADDRESS
;OUTPUTS:
; COMMAND DATA FILLED WITH THE FOLLOWING:
; (R3) OPERATION CODE
; 0 - EXIT
; 1 - WRITE
; 2 - READ
; 3 - DIAGNOSE
; IF OPERATION CODE = 1, 2 OR 3
; 2.(R3) REGION NUMBER
; 4.(R3) OFFSET INTO REGION
; IF OPERATION CODE = 1
; 6.(R3) DATA BYTE
; Z SET IF DATA RETURNED
; Z CLEAR IF DRIVE NUMBER NOT ON THIS CONTROLLER
T2CMD:
BIT #SM,MAN,SFPTBL+SO,BIT ;LOOK AT MANUAL INTERVENTION MODE
BNE T2CMDM ;EXIT IF NOT WANTED
JMP T2CMDX
T2CMDM:
TRAP C#MANI
BCS T2CMD0
T2CMD09:
JSR R1,LPNTF ;CALL LPNTF PRINT ROUTINE
;WORD T2WARN ;ADDRESS OF ASCIZ STRING
;WORD ARG,CT ;ARGUMENT COUNT + 2
JMP T2CMDX
T2CMD0:
MOV (R4)+,R1 ;GET DRIVE NUMBER
MOV (R4)+,R2 ;GET OPERATION CODE
BNE T2CMD2 ;BRANCH IF NOT ZERO
CALL GTDRV ;GET DRIVE TABLE ADDRESS
BEQ 1$ ;CHECK IF DRIVE FOUND
RETURN ;RETURN WITH Z CLEAR IF NOT
1$:
MOV (R4),-(SP) ;PUSH (R4) ON STACK
MOV (R5),-(SP) ;PUSH (R5) ON STACK
MOV D,UNIT(R4),-(SP) ;PUSH D,UNIT(R4) ON STACK
JSR R1,LPNTF ;CALL LPNTF PRINT ROUTINE
;WORD T2CMS1 ;ADDRESS OF ASCIZ STRING
;WORD ARG,CT ;ARGUMENT COUNT + 2
CLR T2WRR ;CLEAR ALL STORAGE WORDS
CLR T2WR0
CLR T2DR

```

```

49
50 100010 022702 000002      T2CMD2: CMP      #2,R2          ;SEE IF LAST OPERATION WAS READ
51 100014 001027              BNE      T2CMDQ          ;BRANCH IF NOT TO QUESTION
52 100016 112700 000040      MOV      #',R0          ;STORE #' IN R0 AND
53 100022 004737 075506      JSR      PC,PRINTC      ;PRINT THE CHARACTER.
54 100026 013701 064642      MOV      T2WRR,R1      ;PRINT REGION
55 100032 004737 103010      CALL     T2PNTW
56 100036 013701 064644      MOV      T2WRO,R1      ;PRINT OFFSET
57 100042 004737 103010      CALL     T2PNTW
58 100046 112700 000057      MOV      #'/,R0        ;STORE #' / IN R0 AND
59 100052 004737 075506      JSR      PC,PRINTC      ;PRINT THE CHARACTER.
60 100056 012401              MOV      (R4),R1        ;PRINT THE DATA
61 100060 004737 103040      CALL     T2PNTB
62 100064 112700 000015      MOV      #CR,R0        ;STORE #CR IN R0 AND
63 100070 004737 075506      JSR      PC,PRINTC      ;PRINT THE CHARACTER.
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93

```

;NOW ASK FOR COMMAND INPUT  
 T2CMDQ:  
 TRAP C#GMAN  
 BR 100004  
 .WORD TEMP  
 .WORD T#CODE  
 .WORD T4OPT7  
 .WORD -1  
 .WORD T#LOLIM  
 .WORD T#HILIM  
 100004:  
 MOV #TEMP,R1 ;GET POINTER TO STRING  
 MOV (R1),R0 ;GET COMMAND CHARACTER  
 CMP #'E,R0  
 BEQ T2CMDV  
 CMP #'D,R0  
 BNE T2CMD3  
 MOV #3,(R3) ;STORE DIAGNOSE OPERATION CODE  
 CALL T2GNUM ;GET REGION FROM COMMAND  
 BEQ 1\$  
 MOV R4,T2DR  
 1\$: MOV T2DR,2(R3)  
 T2CMDV: CALL T2GNUM ;MAKE SURE AT END OF LINE  
 BNE T2CMDE  
 BR T2CMDX  
 ;COMMAND MUST BE EITHER READ OR WRITE  
 T2CMD3: MOV #2,(R3) ;CHECK IF READ  
 CMP #'R,R0  
 BEQ T2CMDR  
 CMP #'W,R0 ;CHECK IF WRITE  
 BNE T2CMDE ; IF NOT - ERROR  
 MOV #1,(R3)  
 CALL T2GNUM ;GET DATA BYTE  
 BEQ T2CMDE ;ERROR IF NO DATA  
 SUB #2,R0  
 BGT T2CMDE  
 MOV R4,6(R3) ;FOR GREATER THAN TWO DIGITS  
 000002 T2CMDR: MOV T2WRR,2(R3) ;STORE DATA BYTES IN BUFFER  
 ;PUT REGION AND OFFSET

94	100244	013763	064644	000004		MOV	T2WRO,4(R3)		; INTO BUFFER
95	100252	021302				CMP	(R3),R2		; IF SO,
96	100254	001002				BNE	T2CMDN		
97	100256	005263	000004			INC	4(R3)		; INCREMENT OFFSET
98	100262	004737	103122		T2CMDN:	CALL	T2GNUM		
99	100266	001411				BEQ	T2CMDW		
100	100270	010463	000002			MOV	R4,2(R3)		
101	100274	005063	000004			CLR	4(R3)		
102	100300	004737	103122			CALL	T2GNUM		
103	100304	001402				BEQ	T2CMDW		
104	100306	010463	000004			MOV	R4,4(R3)		
105	100312	004737	103122		T2CMDW:	CALL	T2GNUM		
106	100316	001010				BNE	T2CMDE		
107	100320	016337	000002	064642		MOV	2(R3),T2WRR		;SAVE REGION
108	100326	016337	000004	064644		MOV	4(R3),T2WRO		;SAVE OFFSET
109	100334	000264			T2CMDX:	SEZ			
110	100336	000207				RETURN			
111	100340				T2CMDE:				
	100340	004137	075664			JSR	R1,LPNTF		;CALL LPNTF PRINT ROUTINE
	100344	066445				.WORD	T2CMS5		;ADDRESS OF ASCII STRING
	100346	000000				.WORD	ARG,CT		;ARGUMENT COUNT * 2
112	100350	000651				BR	T2CMDQ		;GO ASK AGAIN

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 100352 012701 000021
20 100356 012702 064554
21 100362 012223
22 100364 005301
23 100366 001375
24 100370 000264
25 100372 000207

;T4MPRM - DM REQUEST 3
;
;REQUEST FOR TEST 4 CONTENTS OF DATA PATTERN 16.
;
;INPUTS:
;   R5 - CONTROLLER TABLE ADDRESS
;   R4 - MESSAGE DATA ADDRESS
;         (NO DATA)
;   R3 - COMMAND DATA ADDRESS
;
;OUTPUTS:
;   COMMAND DATA FILLED WITH THE FOLLOWING:
;   (R3) NUMBER OF WORDS IN DATA PATTERN 16
;   2.(R3) DATA IN PATTERN 16
;   )
;   32.(R3)
;   Z SET

T4MPRM: MOV     #17.,R1           ;GET COUNT
        MOV     #PAT16C,R2      ; AND ADDRESS OF PATTERN 16 PARAMETERS
1$:     MOV     (R2)+,(R3)+     ;COPY THE DATA TO BUFFER
        DEC     R1
        BNE    1$
        SEZ
        RETURN                   ;RETURN WITH Z SET

```

```

1
2      ;TAUPRM - DM REQUEST 4
3
4      ;REQUEST FOR TEST 4 UNIT PARAMETERS
5
6      ;INPUTS:
7      ;   R5 - CONTROLLER TABLE ADDRESS
8      ;   R4 - MESSAGE DATA ADDRESS
9      ;       (R4) DRIVE NUMBER
10     ;       2.(R4) DRIVE SERIAL NUMBER
11     ;       )
12     ;       5.(R4)
13     ;       8.(R4) HDA SERIAL NUMBER
14     ;       )
15     ;       14.(R4)
16     ;   R3 - COMMAND DATA ADDRESS
17     ;OUTPUTS:
18     ;   COMMAND DATA FILLED WITH THE FOLLOWING:
19     ;   (R3) PARAMETER BITS (1 FOR TRUE)
20     ;   BIT      14 - INITIAL WRITE
21     ;   BIT      13 - DIAGNOSTIC CYLINDERS
22     ;   BIT      12 - ECC CORRECTION
23     ;   BIT      11 - READ ONLY
24     ;   BIT      10 - WRITE ONLY
25     ;   BIT       9 - RETRIES
26     ;   BIT       8 - TRACK/GROUP AND CYLINDERS SPECIFIED
27     ;   BIT       7 - (NOT USED)
28     ;   BIT       6 - SEQUENTIAL SEEKS
29     ;   BIT       5 - BEGIN/END SETS SPECIFIED
30     ;   BIT       4 - TRACK SPECIFIED (0 - GROUPS SPECIFIED)
31     ;   HAS MEANING ONLY WHEN BIT 5 IS ZERO
32     ;   BIT       3 - WRITE CHECKS ENABLED
33     ;   BIT       2 - WRITE CHECKS ALWAYS
34     ;   BIT       1 - DATA COMPARES ENABLED
35     ;   BIT       0 - DATA COMPARE ALWAYS
36     ;   2.(R3) DATA PATTERN NUMBER
37     ;   IF PARAMETER BIT 5 SET
38     ;   4.(R3) COUNT OF BEGIN/END SETS
39     ;   6.(R3) BEGIN BLOCK (2 WORDS) THEN END BLOCK (2 WORDS)
40     ;   )   1 TO 4 SETS
41     ;   )   OR
42     ;   )   IF COUNT OF BEGIN/END BLOCKS = 0
43     ;   36.(R3) START CYLINDER (2 WORDS) THEN END CYLINDER (2 WORDS)
44     ;   END CYLINDER A NEGATIVE VALUE IF TO TEST ENTIRE AREA
45     ;   IF PARAMETER BIT 5 CLEAR
46     ;   4.(R3) STARTING CYLINDER
47     ;   6.(R3) (2 WORDS)
48     ;   8.(R3) ENDING CYLINDER (2 WORDS)
49     ;   10.(R3) NEGATIVE FOR ALL CYLINDERS
50     ;   12.(R3) NUMBER OF TRACKS OR GROUPS SPECIFIED
51     ;   14.(R3) 1 TO 7 TRACK OR GROUP NUMBERS
52     ;   )   DETERMINED BY PARAMETER BIT 4
53     ;   26.(R3)
54     ;   Z SET IF DATA RETURNED
55     ;   Z CLEAR IF UNIT NUMBER NOT ON THIS CONTROLLER

```

```

1
2 100374 012401          T4UPRM: MOV      (R4)+,R1          ;GET DRIVE NUMBER
3 100376 010402          MOV      R4,R2          ;SAVE DATA ADDRESS
4 100400 004737 102274   CALL     G1URVT         ;GET DRIVE TABLE ADDRESS
5 100404 001122          BNE     T4UPRX         ;CHECK IF DRIVE FOUND
6 100406 012264 000200   MOV      (R2)+,D.SERN(R4) ;COPY DRIVE SERIAL NUMBER TO DRIVE TABLE
7 100412 012264 000202   MOV      (R2)+,D.SERN+2(R4)
8 100416 012264 000204   MOV      (R2)+,D.SERN+4(R4)
14 100422 016401 000004   MOV      D.PRM(R4),R1    ;GET PARAMETER BITS
15 100426 042701 140200   BIC     @D.ZERO,R1      ;CLEAR SOME BITS
16 100432 032737 000020 064440   BIT     @ISTRTH,IFLAGS  ;FIRST TIME TEST 4 BEING RUN,
17 100440 001406          BEQ     1$             ;BRANCH IF NOT, ELSE
18 100442 032737 040000 064400   BIT     @SM.IW,SFPTBL+SO.BIT ;GET INITIAL WRITE BIT.
19 100450 001402          BEQ     1$
20 100452 052701 040000   BIS     @D.IW,R1        ;MOVE INTO PARAMETER BITS
21 100456 010123          1$:  MOV     R1,(R3)+       ;PUT INTO BUFFER
22 100460 016423 000006   MOV     D.PAT(R4),(R3)+ ;PUT PATTERN NUMBER IN BUFFER
23 100464 032701 000040   BIT     @D.BE,R1       ;CHECK BEGIN/END PARAMETER BIT
24 100470 001411          BEQ     3$             ;BRANCH IF NOT SET
25
26                      ;RETURN BEGIN/END SETS
27
28 100472 012701 000021   MOV     @4*4+1,R1       ;COUNT OF SETS TIMES WORDS PER SET PLUS COUNT WORD
29 100476 010402          MOV     R4,R2          ;GET INDEX INTO DRIVE TABLE
30 100500 062702 000112   ADD     @D.BEC,R2
31 100504 012223          2$:  MOV     (R2)+,(R3)+   ;TRANSFER THE BEGIN/END SETS
32 100506 005301          DEC     R1
33 100510 001375          BNE     2$
34 100512 000457          BR     T4UPRX
35
36 100514 032764 000400 000004 3$:  BIT     @D.CYL,D.PRM(R4) ;LOOK AT D.CYL BIT
37 100522 001441          BEQ     8$             ;BRANCH IF NOT SET
38
39                      ;RETURN TRACKS/GROUPS AND CYLINDERS
40
41 100524 005764 000112   TST     D.BEC(R4)      ;CHECK IF ANY TRACKS/GROUPS
42 100530 001421          BEQ     6$             ;BRANCH IF NONE
43 100532 012701 000004   MOV     @4,R1          ;COUNT OF CYLINDER WORDS
44 100536 010402          MOV     R4,R2
45 100540 062702 000154   ADD     @D.BCYL,R2
46 100544 012223          4$:  MOV     (R2)+,(R3)+   ;CYLINDERS
47 100546 005301          DEC     R1
48 100550 001375          BNE     4$
49 100552 012701 000010   MOV     @8.,R1
50 100556 010402          MOV     R4,R2
51 100560 062702 000112   ADD     @D.BEC,R2
52 100564 012223          5$:  MOV     (R2)+,(R3)+   ;TRACKS/GROUPS
53 100566 005301          DEC     R1
54 100570 001375          BNE     5$
55 100572 000427          BR     T4UPRX
56
57                      ;RETURN CYLINDERS ONLY
58
59 100574 052763 000040 177774 6$:  BIS     @D.BE,-4(R3)    ;SET D.BE FOR DM PROGRAM
60 100602 005023          CLR     (R3)+         ;SEND ZERO BEGIN/END COUNT
61 100604 012701 000004   MOV     @4,R1
62 100610 010402          MOV     R4,R2

```

```

63 100612 062702 000154          ADD    #D.BCYL,R2
64 100616 012223          7$:    MOV    (R2)+,(R3)+      ;CYLINDERS
65 100620 005301          DEC    R1
66 100622 001375          BNE   7$
67 100624 000412          BR    T4UPRX
68
69                          ;RETURN ENTIRE AREA
70
71 100626 052763 000040 177774 8$:  BIS    #D.BE,-4(R3)      ;SET D.BE FOR DM PROGRAM
72 100634 005023          CLR   (R3)+             ;BEGIN/END COUNT OF ZERO
73 100636 005023          CLR   (R3)+             ;START CYLINDER OF ZERO
74 100640 005023          CLR   (R3)+
75 100642 005023          CLR   (R3)+
76 100644 012723 177777          MOV   #-1,(R3)+        ;END CYLINDER NEGATIVE
77 100650 000264          SEZ
78 100652 000207          T4UPRX: RETURN

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 100654 011401
24 100656 004737 102274
25 100662 001007
26 100664 062704 000010
27 100670 012701 000035
28 100674 012423
29 100676 005301
30 100700 001375
31 100702 000207

;T4BB1 - DM REQUEST 5
;
;REQUEST FOR FIRST 14 BAD BLOCKS
;
;INPUTS:
;   R5 - CONTROLLER TABLE ADDRESS
;   R4 - MESSAGE DATA ADDRESS
;       (R4) DRIVE NUMBER
;   R3 - COMMAND DATA ADDRESS
;
;OUTPUTS:
;   COMMAND DATA FILLED WITH BAD BLOCKS
;   (R3) COUNT OF BAD BLOCKS
;   2.(R3) BAD BLOCK 1 (LOW)
;   4.(R3)          (HIGH)
;   )
;   )
;   56.(R3) BAD BLOCK 14 (LOW)
;   58.(R3)          (HIGH)
;   Z SET IF DATA RETURNED
;   Z CLEAR IF DRIVE NUMBER NOT ON THIS CONTROLLER

T4BB1:  MOV     (R4),R1           ;GET DRIVE NUMBER
        CALL   GTDRVT          ;GET DRIVE TABLE ADDRESS
        BNE    T4BB1E          ;CHECK IF DRIVE FOUND
        ADD    @D.BB,R4        ;INCREASE ADDRESS TO DATA TO COPY
        MOV    @<1+<14.*2>>,R1 ;GET COUNT OF WORDS
1$:     MOV    (R4)+,(R3)+     ;COPY THE WORDS
        DEC   R1
        BNE   1$
T4BB1E: RETURN

```

```

1
2
3 ;T4BB2 - DM REQUEST 6
4 ;
5 ;REQUEST LAST TWO BAD BLOCKS
6 ;
7 ;INPUTS:
8 ; R5 - CONTROLLER TABLE ADDRESS
9 ; R4 - MESSAGE DATA ADDRESS
10 ; (R4) DRIVE NUMBER
11 ; R3 - COMMAND DATA ADDRESS
12 ;OUTPUTS:
13 ; COMMAND DATA FILLED WITH BAD BLOCKS 15 AND 16
14 ; Z SET IF DATA RETURNED
15 ; Z CLEAR IF UNIT NUMBER NOT ON THIS CONTROLLER
16 100704 011401
17 100706 004737 102274
18 100712 001007
19 100714 062704 000102
20 100720 012701 000004
21 100724 012423
22 100726 005301
23 100730 001375
24 100732 000207

T4BB2: MOV (R4),R1 ;GET DRIVE NUMBER
CALL GTDRVT ;GET DRIVE TABLE ADDRESS
BNE T4BB2E ;CHECK IF DRIVE FOUND
ADD #0.8815,R4 ;INCREASE ADDRESS TO DATA TO COPY
MOV #4,R1 ;GET COUNT OF WORDS
1$: MOV (R4)+,(R3)+ ;COPY THE WORDS
DEC R1
BNE 1$

T4BB2E: RETURN

```

GLOBAL SUBROUTINES SECTION

```

1
2           ;T4SOFT - DM REQUEST 7
3           ;
4           ;ADD TO SOFT ERROR AND ECC COUNTS
5           ;
6           ;INPUTS:
7           ;
8           ;   R5 - CONTROLLER TABLE ADDRESS
9           ;   R4 - MESSAGE DATA ADDRESS
10          ;   (R4) DRIVE NUMBER
11          ;   2.(R4) VALUE TO ADD TO SOFT ERROR COUNT
12          ;   4.(R4) VALUE TO ADD TO ECC COUNT
13          ;
14          ;   R3 - COMMAND DATA ADDRESS
15
16
17 100734 012401
18 100736 010402
19 100740 034737 102274
20 100744 001005
21 100746 062264 000172
22 100752 062264 000176
26 100756 000264
27 100760 000207

T4SOFT: MOV      (R4)+,R1           ;GET DRIVE NUMBER
        MOV      R4,R2           ;SAVE DATA ADDRESS
        CALL     G'DRVT          ;GET DRIVE TABLE ADDRESS
        BNE     1$              ;CHECK IF DRIVE FOUND
        ADD     (R2)+,D,SERR(R4) ;ADD TO SOFT ERROR COUNT
        ADD     (R2)+,L,ECCC(R4) ;ADD TO ECC COUNT
        SEZ
        1$: RETURN              ;EXIT

```

1					
2	100762			T4SEEK:	
3				:	DM REQUEST 8.
4				:	
5				:	RECORD 1000 SEEKS COMPLETED ON DRIVE
6				:	
7				:	INPUTS:
8				:	R5 - CONTROLLER TABLE ADDRESS
9				:	R4 - MESSAGE DATA ADDRESS
10				:	(R4) DRIVE NUMBER
11				:	R3 - COMMAND DATA ADDRESS
12				:	
13	100762	011401		MOV	(R4),R1 ; GET DRIVE NUMBER
14	100764	004737	102274	CALL	GDRVT ; GET DRIVE TABLE ADDRESS
15	100770	001003		BNE	SEKERE ; CHECK IF DRIVE FOUND
16	100772	005264	000174	INC	D,SEEK(R4) ; COUNT THE BITS TRANSFERRED
17	100776	000264		SEZ	SEKERE ; NORMAL RETURN
18	101000	000207		SEKERE:	RETURN

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 101002 010402
20 101004 011401
21 101006 004737 102274
22 101012 001053
23 101014 005764 000002
24 101020 100003
26 101022 052713 100000
27 101026 000444
28
29 101030
44 101030 066264 000002 000166
45 101036 066264 000004 000164
46 101044 005737 064376
47 101050 001433
48 101052 026437 000166 064376
49 101060 103427
50 101062 104421
51 101064 032700 000040
52 101070 001023
53 101072 052713 100000
54 101076 042765 000010 000014
55 101104 011446
    101106 011546
    101110 016446 000002
    101114 004137 075704
    101120 065713
    101122 000006
56 101124 004737 106367
57 101130 004137 075704
    101134 065170
    101136 000000
58 101140 000264
59 101142 000207

;TAMXFR - DM REQUEST 9.
;
;RECORD 1M BITS TRANSFERRED ON UNIT. COMPARE TO TRANSFER LIMIT AND
;REPORT LIMIT REACHED.
;
;INPUTS:
;   R5 - CONTROLLER TABLE ADDRESS
;   R4 - MESSAGE DATA ADDRESS
;       (R4) DRIVE NUMBER
;       2.(R4) VALUE TO ADD TO READ COUNT
;       4.(R4) VALUE TO ADD TO WRITE COUNT
;   R3 - COMMAND DATA ADDRESS
;
;OUTPUTS:
;   (R3) BIT 15 SET IF TRANSFER LIMIT REACHED
;   MESSAGE PRINTED IF TRANSFER LIMIT REACHED
;   Z CLEAR IF DRIVE NUMBER NOT ON THIS CONTROLLER

TAMXFR: MOV     R4,R2           ;GET MESSAGE DATA ADDRESS
        MOV     (R4),R1      ;GET DRIVE NUMBER
        CALL   GDRVT        ;GET DRIVE TABLE ADDRESS
        BNE    MXFERE       ;CHECK IF DRIVE FOUND
        TST    D,UNIT(R4)   ;SEE IF UNIT HAS BEEN DROPPED
        BPL    1#          ;CONTINUE IF STILL TO BE TESTED
        BIS    @BIT15,(R3)  ;TELL DM PROGRAM TO STOP TESTING THIS UNIT
        BR     MXFERX       ; AND EXIT WITHOUT ADDING TO ADDING TO COUNTS

1# :
        ADD    2(R2),D,XFRR(R4) ;ADD MEGABITS READ
        ADD    4(R2),D,XFRW(R4) ;ADD MEGABITS WRITTEN
        TST    SFPTBL+50,XL   ;SEE IF LIMIT SPECIFIED
        BEQ    MXFERX        ;BRANCH IF NOT
        CMP    D,XFRR(R4),SFPTBL+50,XL ;CHECK IF LIMIT REACHED
        BLO    MXFERX        ;BRANCH IF LIMIT NOT REACHED
        TRAP   C@RFLA
        BIT    @IDU,R0       ;SEE IF DROPPING UNITS IS INHIBITED
        BNE    MXFERX
        BIS    @BIT15,(R3)   ;SET DROP UNIT BIT
        BIC    @CT,MSG,C,FLG(R5) ;CLEAR MESSAGE RECEIVED FLAG
        MOV    (R4),-(SP)    ;PUSH (R4) ON STACK
        MOV    (R5),-(SP)    ;PUSH (R5) ON STACK
        MOV    D,UNIT(R4),-(SP) ;PUSH D,UNIT(R4) ON STACK
        JSR   R1,LPNTX      ;CALL LPNTX PRINT ROUTINE
        .WORD MSGG         ;ADDRESS OF ASCI2 STRING
        .WORD ARG,CT       ;ARGUMENT COUNT * 2
        CALL  RNTI,%        ;PRINT RUNTIME
        JSR   R1,LPNTX      ;CALL LPNTX PRINT ROUTINE
        .WORD MXFERP       ;ADDRESS OF ASCI2 STRING
        .WORD ARG,CT       ;ARGUMENT COUNT * 2
        SEZ
        MXFERX: SEZ
        MXFERE: RETURN
    
```

```

1
2
3      ;UTOTST - DM REQUEST 10
4      ;
5      ;TELL DM PROGRAM WHICH DRIVES ARE SELECTED FOR TESTING
6      ;AND CLEAR STATISTICS IN DRIVE TABLE
7      ;
8      ;INPUTS:
9      ;   R5 - CONTROLLER TABLE ADDRESS
10     ;   R4 - MESSAGE DATA ADDRESS
11     ;         (NO DATA)
12     ;   R3 - COMMAND DATA ADDRESS
13     ;OUTPUTS:
14     ;   COMMAND PACKET CONTAINING UP TO 8 DRIVE NUMBERS.
15     ;   LIST IS ENDED BY A WORD WITH BIT 15 SET.
16     ;   FOLLOWING LIST IS THE INFORMATION FROM T2DLL REQUEST IF APPLICABLE.
17     ;   D.XFRW, D.XFRR, D.HERR, D.SERR, D.SEEK AND D.ECC CLEARED IN DRIVE TABLE.
18     ;   Z SET
19 101144 010504      UTOTST: MOV      R5,R4      ;GET ADDRESS OF CONTROLLER TABLE.
20 101146 062704 0000'0  ADD      @C.DRO,R4      ;BUMP TO DRIVE TABLE POINTERS
21 101152 012702 000010  MOV      @B.,R2      ;GET COUNT OF PORTS
22 101156 012400      UTOT1:  MOV      (R4),R0      ;SEE IF DRIVE TABLE POINTER EXISTS
23 101160 001415      BEQ      UTOT2      ;BRANCH IF NOT
24 101162 005760 000002  TST      D.UNIT(R0)    ;LOOK IF UNIT AVAILABLE FOR TESTING
25 101166 100410      BMI      UTOT1A
26 101170 011023      MOV      (R0),(R3)    ;LOAD DRIVE NUMBER FROM TABLE
27 101172 062700 000164  ADD      @D.XFRW,R0    ;CLEAR STATISTICS IN DRIVE TABLE
28 101176 012701 000011  MOV      @<D.SIZE-D.XFRW>/2,R1
29 101202 005020      1$:   CLR      (R0)
30 101204 005301      DEC      R1
31 101206 001375      BNE     1$
32 101210 005302      UTOT1A: DEC     R2      ;COUNT THE DRIVE TABLES
33 101212 001361      BNE     UTOT1      ;REPEAT FOR EACH TABLE
34 101214 012723 100000  UTOT2:  MOV      @BIT15,(R3) ;TERMINATE LIST
35 101220 013723 064656  MOV      DLL,(R3)    ;GET DLL CONTROL WORD
36 101224 001407      BEQ     UTOT4      ; IF NON ZERO
37 101226 012701 064660  MOV      @DLLDR,R1    ; TRANSFER ALL DLL WORDS INTO BUFFER
38 101232 012702 000020  MOV      @<DLLNAM:4-DLLDR>,R2
39 101236 012123      UTOT3:  MOV      (R1),(R3)
40 101240 005302      DEC     R2
41 101242 001375      BNE     UTOT3
42 101244 000264      UTOT4:  SEZ
43 101246 000207      RETURN      ;RETURN WITH Z SET

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73

```

ERRMES - DM REQUEST 11
PRINT AN ERROR MESSAGE
INPUTS:
R5 - CONTROLLER TABLE ADDRESS
R4 - MESSAGE DATA ADDRESS
(R4) ERROR PC IN DM PROGRAM
2.(R4) <15:14> ERROR TYPE
(R4) <13:0> ERROR NUMBER
4.(R4) DRIVE NUMBER (-1 IF NOT GIVEN)
6.(R4) MESSAGE POINTER
8.(R4) OPTIONAL PARAMETERS FOR ERROR PRINT ROUTINE
10.(R4) "
) "
) "
58.(R4) "
R3 - COMMAND DATA ADDRESS
OUTPUTS:
COMMAND PACKET CONTAINING THE FOLLOWING:
(R3) - BIT 15 SET IF FATAL ERROR TO INDICATE DRIVE SHOULD NO LONGER BE TESTED
Z SET TO INDICATE DATA RETURNED
Z CLEAR IF DRIVE NUMBER NOT ON THIS CONTROLLER
ERRMES:
TST 2(R4) ;CHECK IF FATAL ERROR
BMI 5; ;BRANCH IF NOT
TRAP C,RFLA
BIT #IDU,R0 ;SEE IF ALLOWED TO DROP UNITS
BNE 6; ;BRANCH IF NOT
BIS #BIT15,(R3) ;SET DROP DRIVE BIT
5;: MOV 2(R4),R0 ;SEE IF SOFT ERROR
COM R0
BIT #140000,R0
BNE 6; ;BRANCH IF NOT
BIT #SM.SSF,SO,BIT,SFPTBL ;SEE IF SOFT ERRORS SUPPRESSED
BNE ERRMSX ;DON'T PRINT IF SO
6;: BIC #CT.MSG,C,FLG(R5) ;CLEAR MESSAGE RECEIVED FLAG
CMP #4,INUM ;ARE WE DOING DISK EXERCISER TEST?
BNE 7; ;BRANCH IF NOT
BIT #SM.LOG,SFPTBL+SU,BIT ;SEE IF LOG BEING USED
BNE ERRMSL
7;: CALL PNTERR ;IF NOT, PRINT THE ERROR MESSAGE
BCC ERRMSX ;IF DRIVE HASN'T BEEN DROPPED, PRINT
CI Z ;ELSE RETURN
RETURN
    
```

25	101250				
26	101250	005764	000002		
27	101254	100406			
28	101256	104421			
29	101260	032700	000040		
30	101264	001014			
31	101266	052713	100000		
32	101272	016400	000002		
33	101276	005100			
34	101300	032700	140000		
35	101304	001004			
36	101306	032737	000400	064400	
37	101314	001063			
38	101316				
39	101316	042765	000010	000014	
40	101324	022737	000004	064444	
41	101332	001004			
42	101334	032737	001000	064400	
43	101342	001005			
44	101344	004737	103250		
45	101350	103045			
46	101352	000244			
47	101354	000207			

```

1
2 101356 005737 064650      ERRMSL: TST      LBUFS
3 101362 001016              BNE      1$
4 101364 013701 064432      MOV      DMPROG,R1
5 101370 005721              TST      (R1),
6 101372 010137 064650      MOV      R1,LBUFS
7 101376 010137 064652      MOV      R1,LBUFN
8 101402 067701 163024      ADD      @DMPROG,R1
9 101406 005741              TST      -(R1)
10 101410 010137 064654     MOV      R1,LBUFE
11 101414 005037 064442     CLR      FNUM
12 101420 013701 064652     1$: MOV      LBUFN,R1
13 101424 062737 000106 064652  ADD      @HC.BSZ,LBUFN
14 101432 023737 064652 064654  CMP      LBUFN,LBUFE
15 101440 103007              BHIS 3$
16 101442 010521              MOV      R5,(R1),
17 101444 012700 000042     MOV      @<HC.BSZ-2>/2,R0
18 101450 012421              2$: MOV      (R4),,(R1),
19 101452 005300              DEC      R0
20 101454 001375              BNE      2$
21 101456 000402              BR       ERRMSX
22 101460 010137 064652     3$: MOV      R1,LBUFN
23 101464 000264              ERRMSX: SEZ
24 101466 000207              RETURN

;SEE IF LOG BUFFER ESTABLISHED
; LBUFS CONTAINS ADDRESS IF ESTABLISHED
; LBUFS <- (DMPROG)+2
; LBUFE <- (LBUFS) + ((DMPROG)) - 2
;GET ADDRESS OF DATA STORAGE AREA
;ADD BYTES OF STORAGE NEEDED
;SEE IF ENOUGH ROOM
; BRANCH IF NOT
;STORE CONTROLLER TABLE ADDRESS
;GET COUNT OF REST OF DATA IN WORDS
;STORE DATA
;RESTORE OLD VALUE OF LBUFN

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29

```

;ERRMC - DM REQUEST 12.
;
;REPORT AN ERROR MESSAGE IDENTICAL TO DM REQUEST ERRMES
;THEN ADD ONE TO THE ERROR COUNT FOR THE DRIVE AND SEE IF
;ERROR LIMIT REACHED.
;
;INPUTS:
;   R5 - CONTROLLER TABLE ADDRESS
;   R4 - MESSAGE DATA ADDRESS
;       (R4) ERROR PC IN DM PROGRAM
;       2.(R4) < 9:8 > ERROR TYPE
;           < 7:0 > ERROR NUMBER
;       4.(R4) DRIVE NUMBER (-1 IF NOT GIVEN)
;       6.(R4) <15:12> TYPE
;           <11:0 > MESSAGE POINTER
;       8.(R4) OPTIONAL PARAMETERS FOR ERROR PRINT ROUTINE
;       10.(R4) "
;           "
;       )
;       )
;       58.(R4) "
;
;   R3 - COMMAND DATA ADDRESS
;
;OUTPUTS:
;   COMMAND PACKET CONTAINING THE FOLLOWING:
;   (R3) BIT 15 SET IF ERROR COUNT REACHED
;       TO INDICATE DRIVE SHOULD NO LONGER BE TESTED.
;   Z CLEAR IF DRIVE NUMBER NOT ON THIS CONTROLLER
;   Z SET TO INDICATE DATA RETURNED

```

```

30 101470
31 101470 010446
32 101472 004737 101250
33 101476 012604
34 101500 005713
35 101502 100436
36 101504 016401 000004
37 101510 016402 000002
38 101514 004737 102274
39 101520 001031
40 101522 042702 037777
41 101526 022702 100000
42 101532 001022
43 101534 005264 000170
44 101540 026437 000170 064374
45 101546 103414
46 101550 104421
47 101552 032700 000040
48 101556 001010
49 101560 016446 000002
50 101564 004137 075704
51 101570 065245
52 101572 000002
53 101574 052713 100000
54 101600 000264
55 101602 000207
56 101604 000244

```

```

ERRMC:
MOV R4, -(SP) ; PUSH R4 ON STACK
CALL ERRMES ; CALL REQUEST ERRMES
MOV (SP)+, R4 ; POP STACK INTO R4
YST (R3) ; SEE IF UNIT ALREADY TO BE DROPPED
BMI 3$ ; IF SO, JUST EXIT NOW
MOV 4(R4), R1 ; GET DRIVE NUMBER
MOV 2(R4), R2 ; GET ERROR TYPE
CALL GTDRVT ; GET DRIVE TABLE
BNE 5$ ; EXIT IF NO TABLE FOR UNIT
BIC #140000, R2
CMP #100000, R2 ; CHECK IF HARD ERROR
BNE 3$ ; BRANCH IF NOT
INC D.HERR(R4) ; COUNT THE ERROR
CMP D.HERR(R4), SFPTBL+50, EL ; CHECK IF AT LIMIT
BLO 3$ ; IF LIMIT REACHED, BRANCH
TRAP C:RFLA
BIT #IDU, R0 ; SEE IF DROPPING UNITS INHIBITED
BNE 3$ ; BRANCH IF SO
MOV D.UNIT(R4), -(SP) ; PUSH D.UNIT(R4) ON STACK
JSR R1, LPNTX ; CALL LPNTX PRINT ROUTINE
; ADDRESS OF ASCIZ STRING
; ARGUMENT COUNT * 2
BIS #BIT15, (R3) ; SET STOP TESTING BIT
3$: SEZ ; SET Z FOR NORMAL RETURN
RETURN ; RETURN TO CALLING PROGRAM
5$: CLZ ; FLAG AS ERROR

```

G15

CZUDCEO LDA & DISK DRY DIAG MACRO V05.00 Wednesday 04-Jan-84 16:12 Page 142-1  
GLOBAL SUBROUTINES SECTION

SFQ 0188

66 101606 000207

RETURN

; RETURN TO CALLING PROGRAM

```

1
2
3      ;MESSAG - DM REQUEST 13.
4
5      ;PRINT A MESSAGE WITH HEADER AS FOLLOWS:
6      ; "UNIT XX UDA AT XXXXXX DRIVE XXX RUNTIME HH:MM:SS "
7      ;ENTIRE MESSAGE IS PRINTED WITH PRINTX CALLS.
8
9      ;INPUTS:
10     ; R5 - CONTROLLER TABLE ADDRESS
11     ; R4 - MESSAGE DATA ADDRESS
12     ; (R4) DRIVE NUMBER
13     ; 2.(R4) MESSAGE POINTER
14     ; 2.(R4) MESSAGE POINTER
15     ; 4.(R4) OPTIONAL MESSAGE PARAMETERS
16     ; )
17     ; )
18     ; 58.(R4) COMMAND DATA ADDRESS
19 101610 042765 000010 000014 MESSAG: BIC      @CT.MSG,C,FLG(R5)      ;CLEAR MESSAGE RECEIVED FLAG
20 101616 012401                MOV      (R4)+,R1          ;GET DRIVE NUMBER
21 101620 010446                MOV      R4,-(SP)         ;PUSH R4 ON STACK
22 101622 004737 102274        CALL    GDRVT          ;GET DRIVE TABLE ADDRESS
23 101626 001033                BNE     1$             ;CHECK IF DRIVE FOUND
24 101630 005764 000002        TST     D,UNIT(R4)     ;IF UNIT DROPPED FROM TESTING
25 101634 100430                BMI     1$             ; DON'T PRINT ANYTHING
26 101636 011446                MOV      (R4),-(SP)    ;PUSH (R4) ON STACK
    101640 011546                MOV      (R5),-(SP)    ;PUSH (R5) ON STACK
    101642 016446 000002        MOV      D,UNIT(R4),-(SP) ;PUSH D.UNIT(R4) ON STACK
    101646 004137 075704        JSR     R1,LPNTX      ;CALL LPNTX PRINT ROUTINE
    101652 065713                .WORD   MESSG         ;ADDRESS OF ASCIZ STRING
    101654 000006                .WORD   ARC.CT       ;ARGUMENT COUNT * 2
27 101656 004737 106362        CALL    RNTIME        ;GET RUNTIME PARAMETERS
28 101662 012604                MOV      (SP)+,R4     ;;POP STACK INTO R4
29 101664 012402                MOV      (R4)+,R2     ;GET MESSAGE POINTER
30 101666 006302                ASL     R2            ;DOUBLE TO MAKE BYTE OFFSET
31 101670 063702 064432        ADD     DMPROG,R2     ;ADD TO START OF MESSAGE STRINGS
32 101674 067702 162532        ADD     SUMPORG,R2   ;ADD SIZE OF MAIN PROGRAM
33 101700 105712                TSTB   (R2)          ;CHECK FIRST BYTE
34 101702 001001                BNE     2$           ;IF ZERO
35 101704 005202                INC     R2           ; INCREMENT TO NEXT BYTE
36 101706 004737 075772        2$:   CALL    OSTRNG    ;OUTPUT ACCORDING TO STRING
    SEZ
37 101712 0 0264                RETURN
38 101714 000207                1$:
39 101716 012604                MOV      (SP)+,R4
40 101720 000207                RETURN

```

```

1
2
3      ;DONE - DM REQUEST 14
4      ;
5      ;MARK DM PROGRAM AS NO LONGER RUNNING
6      ;
7      ;INPUTS:
8      ;      R5 - CONTROLLER TABLE ADDRESS
9      ;      R4 - MESSAGE DATA ADDRESS
10     ;      (NO DATA)
11     ;      R3 - COMMAND DATA ADDRESS
12     ;OUTPUTS:
13     ;      Z CLEAR TO DROP UNIT FROM TESTING
14 101722 000244
15 101724 000207
DONE:  CLZ      ;DROP UNIT FROM TESTING
      RETURN

```

```

1          .SBTTL  PRE-PROGRAMMED SUBROUTINES
2
3          ;PRE-PROGRAMMED ROUTINE 1
4          ;
5          ;CALL ALTERNATE PRINT STRING IN DM PROGRAM IMAGE
6
7          CALR1:
8          MOV     R2, -(SP)          ;; PUSH R2 ON STACK
9          MOV     (R4)+, R2        ;; GET NEW STRING POINTER
10         ASL     R2                ;; DOUBLE FOR WORD COUNT
11         ADD     DMPROG, R2        ;; ADD START OF STRING STORAGE
12         ADD     @DMPROG, R2       ;; ADD SIZE OF MAIN PROGRAM
13         CALL    OSTRNG           ;; OUTPUT USING THIS STRING
14         MOV     (SP)+, R2        ;; POP STACK INTO R2
15         RETURN                   ;; NOW CONTINUE THE OLD STRING

```

```

1
2
3
4
5
6 101754
7 101754 010246
8 101756 012402
9 101760 010246
10 101762 042702 177400
11 101766 001414
12 101770 012700 000000
13 101774 012701 000040
14 102000 004737 102450
15 102004 112700 000015
16 102010 004737 075506
17 102014 005302
18 102016 001364
19 102020
20 102020 012601
21 102022 000301
22 102024 042701 177400
23 102030 001406
24 102032 004737 076066
25 102036 112700 000015
26 102042 004737 075506
27 102046
28 102046 012602
29 102050 000207

;PRE-PROGRAMMED ROUTINE 2
;
;PRINT AN SDI DIAGNOSE RESPONSE

CALR2:
MOV R2, -(SP) ;; PUSH R2 ON STACK
MOV (R4)+, R2 ;; GET COUNTS
MOV R2, -(SP) ;; PUSH R2 ON STACK
BIC #177400, R2 ;; GET BINARY COUNT
BEQ 2$ ;; BYPASS BINARY IF COUNT IS ZERO
1$: MOV #16, R0 ;; RADIX IS HEX
MOV #32, R1 ;; 32 BIT NUMBERS
CALL PNTNUS ;; PRINT THE NUMBER
MOVB #CR, R0 ;; STORE #CR IN R0 AND
JSR PC, PRINTC ;; PRINT THE CHARACTER.
DEC R2
BNE 1$

2$: MOV (SP)+, R1 ;; POP STACK INTO R1
SWAB R1 ;; GET ASCII COUNT
BIC #177400, R1
BEQ 3$ ;; BYPASS IS COUNT IS ZERO
CALL CON.A1 ;; PRINT THE ASCII
MOVB #CR, R0 ;; STORE #CR IN R0 AND
JSR PC, PRINTC ;; PRINT THE CHARACTER.

3$: MOV (SP)+, R2 ;; POP STACK INTO R2
RETURN

```

```

1
2           ;PRE-PROGRAMMED ROUTINE 3
3           ;
4           ;DECIDE WHETHER TO PRINT RBN
5
6           ;FOUR PARAMETERS ARE PROVIDED FOR THIS ROUTINE. THE FIRST PARAMETER
7           ;SHOULD BE CHECKED TO SEE IF BIT 7 IS SET:
8           ; IF SET - TURN INTO A CALL TO ROUTINE 1 (WHICH WILL USE OTHER 3 PARAMETERS)
9           ; IF CLEAR - SKIP OVER NEXT 3 PARAMETERS AND END ROUTINE
10
11 102052 032724 000200 CALR3: BIT    #BIT7,(R4)      ;CHECK BIT 7 IN FIRST PARAMETER WORD
12 102056 001323          BNE    CALR1      ;IF SET, TURN INTO A CALR1
13 102060 062704 000006          ADD    #6,R4      ;ELSE, SKIP OVER NEXT 3 PARAMETERS
14 102064 000207          RETURN

```

```

1
2           ;PRE-PROGRAMMED ROUTINE 4
3           ;
4           ;PRINT BASIC LINE FOR HOST PROGRAM ERROR WITHOUT UDA ADDRESS
5           ;THEN SWITCH TO EXTENDED FORMAT
6
7 102066    CALR4:
102066     MOV     #BAS, -(SP)           ;PUSH #BAS ON STACK
102072     MOV     #BAS, -(SP)           ;PUSH #BAS ON STACK
102076     MOV     #BAS, -(SP)           ;PUSH #BAS ON STACK
102102     MOV     #BASNO, -(SP)         ;PUSH #BASNO ON STACK
102106     JSR     R1, LPNTB             ;CALL LPNTB PRINT ROUTINE
102112     .WORD   BASLN                 ;ADDRESS OF ASCIZ STRING
102114     .WORD   ARG.CT                ;ARGUMENT COUNT * 2
8 102116     CALL  RNTIME
9 102122     MOVB  #CR, R0               ;STORE #CR IN R0 AND
102126     JSR     PC, PRINTC            ;PRINT THE CHARACTER.
10 102132     MOV     #PX, PTYPE
11 102140     RETURN

```

```

1
2
3
4
5
6
7 102142
102142 012746 067047
102146 012746 067047
102152 011546
102154 012746 067013
102160 012746 066651
102164 004137 075674
102170 067050
102172 000012
8 102174 004737 106362
9 102200 112700 000015
102204 004737 075506
10 102210 012737 075612 075770
11 102216 000207

;PRE-PROGRAMMED ROUTINE 5
;
;PRINT BASIC LINE FOR HOST PROGRAM ERROR WITH UDA ADDRESS
;THEN SWITCH TO EXTENDED FORMAT

CALR5:
MOV #BAS,-(SP) ;PUSH #BAS ON STACK
MOV #BAS,-(SP) ;PUSH #BAS ON STACK
MOV (R5),-(SP) ;PUSH (R5) ON STACK
MOV #BASL2,-(SP) ;PUSH #BASL2 ON STACK
MOV #BASNO,-(SP) ;PUSH #BASNO ON STACK
JSR R1,LPNTB ;CALL LPNTB PRINT ROUTINE
.WORD BASLN ;ADDRESS OF ASCIZ STRING
.WORD ARG.CT ;ARGUMENT COUNT * 2
CALL RNTIME
MOVB #CR,R0 ;STORE #CR IN R0 AND
JSR PC,PRINTC ;PRINT THE CHARACTER.
MOV #PX,PTYPE
RETURN

```

1				
2				
3				
4				
5				
6	102220			CALR6:
7	102220	010246		MOV R2, (SP)
8	102224	004737	075772	MOV (R4)+, R2
9	102230	012602		CALL OSTRING
10	102232	000207		MOV (SP)+, R2
				RETURN

;PRE PROGRAMMED ROUTINE 6  
 ;CALL ALTERNATE PRINT ROUTINE IN PDP-11 MEMORY  
 ;PUSH R2 ON STACK  
 ;GET NEW STRING POINTER  
 ;OUTPUT USING THIS STRING  
 ;POP STACK INTO R2  
 ;NOW CONTINUE THE OLD STRING

```

1
2
3
4
5
6 102234
   102234 010246
7 102236 012702 074215
8 102242 004737 075772
9 102246 012602
10 102250 000207

;PRE-PROGRAMMED ROUTINE 7
;
;PRINT "REPLACE UDA MODULE M7161"
CALLR7:
MOV R2, (SP) ;PUSH R2 ON STACK
MOV 0XFRU,R2
CALL OSTRNG
MOV (SP)+,R2 ;POP STACK INTO R2
RETURN

```

```

1
2
3
4
5
6 102252
   102252 010246
7 102254 012702 074164
8 102260 004737 075772
9 102264 012602
10 102266 000207

;PRE-PROGRAMMED ROUTINE 8
;PRINT " UDASA CONTAINS XXXXXX"

CALR8:
MOV     R2, (SP)           ;PUSH R2 ON STACK
MOV     @XSA,R2
CALL   OSTRNG
MOV     (SP)+,R2         ;POP STACK INTO R2
RETURN

```

1  
2  
3  
4  
5  
6 102270 005744  
7 102272 000207

PRE-PROGRAMMED ROUTINE 9  
:  
: REPRINT LAST NUMBER  
: R4 -> TABLE  
CALR9: TST -(R4)  
RETURN

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 102274
15 102274 010246
16 102276 010504
17 102300 062704 000020
18 102304 012702 000010
19 102310 005714
20 102312 001406
21 102314 027401 000000
22 102320 001412
23 102322 005724
24 102324 005302
25 102326 001370
26 102330
27 102330 104455
28 102332 000043
29 102334 000000
30 102336 075000
31 102340 012602
32 102342 000244
33 102344 000207
34 102346 011404
35 102350 116437 000002 002074
36 102356 012602
37 102360 000264
38 102362 000207

;GDRVT
;
;GET DRIVE TABLE POINTER
;
;INPUTS:
; R5 - CONTROLLER TABLE ADDRESS
; R1 - DRIVE NUMBER
;OUTPUTS:
; R4 - DRIVE TABLE ADDRESS
; L$LUN - LOADED WITH UNIT NUMBER OF DRIVE
; Z CLEAR IF DRIVE TABLE NOT FOUND AFTER ERROR PRINTED

GDRVT:
MOV R2, (SP) ;; PUSH R2 ON STACK
MOV R5, R4 ;; GET CONTROLLER TABLE ADDRESS
ADD @C.DRO, R4 ;; ADJ OFFSET TO DRIVE TABLE ADDRESS
MOV @B, R2 ;; GET COUNT OF DRIVES
1$: TST (R4) ;; CHECK IF AN ADDRESS HERE
BEQ 3$
CMP @ (R4), R1 ;; COMPARE DRIVE NUMBERS
BEQ 4$ ;; BRANCH IF A MATCH
2$: TST (R4)+ ;; BUMP ADDRESS
DEC R2
BNE 1$ ;; LOOK AT ALL OF THEM
3$: TRAP C$ERDF
;WORD 3$
;WORD 0
;WORD ERRO35
MOV (SP)+, R2 ;; POP STACK INTO R2
CLZ ;; CLEAR Z AS ERROR FLAG
RETURN

4$: MOV (R4), R4 ;; GET ADDRESS OF TABLE
MOVB D.UNIT(R4), L$LUN ;; GET UNIT NUMBER
MOV (SP)+, R2 ;; POP STACK INTO R2
SEZ ;; SET Z FLAG
RETURN

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 102364
    102364 010046
15 102366 005001
16 102370 121227 000060
17 102374 103415
18 102376 121227 000071
19 102402 101012
20 102404 006301
21 102406 010100
22 102410 006301
23 102412 006301
24 102414 060001
25 102416 112200
26 102420 162700 000060
27 102424 060001
28 102426 000760
29 102430 005701
30 102432 001001
31 102434 005201
32 102436
    102436 012600
33 102440 000207

;GETCNT
;
;GET COUNT IN NEXT CHARACTERS OF STRING POINTED TO BY R2.
;NUMBER WILL BE IN DECIMAL. IF NO NUMBER, RETURN A
;DEFAULT OF 1.
;
;INPUTS:
; R2 - POINTER TO ASCII STRING
;OUTPUTS:
; R1 - NUMBER READ OR A ONE
; R2 - POINTING TO CHARACTER AFTER NUMBER

GETCNT:
MOV     RO, -(SP)           ;;PUSH RO ON STACK
CLR     R1                 ;;START WITH ZERO COUNT
GETCNX: CMPB    (R2), #'0    ;;CHECK IF CHARACTER A DIGIT
        BLO     GETCDN      ;;BRANCH IF LOWER THAN ZERO
        CMPB    (R2), #'9
        BHI     GETCDN      ;;BRANCH IF HIGHER THAN NINE
        ASL     R1           ;;MULTIPLY NUMBER BY 10
        MOV     R1, RO       ;;SAVE 2N
        ASL     R1           ;;COMPUTE 4N
        ASL     R1           ;;COMPUTE 8N
        ADD     RO, R1       ;;8N + 2N = 10N
        MOVB   (R2)+, RO    ;;GET DIGIT FROM STRING
        SUB     #'0, RO     ;;GET RID OF ASCII
        ADD     RO, R1       ;;ADD TO NUMBER
        BR     GETCNX       ;;GO TO NEXT CHARACTER
        TST     R1           ;;CHECK IF NUMBER IS ZERO
        BNE     GETCXX      ;;IF ZERO, CHANGE
        INC     R1           ;;TO DEFAULT OF ONE

GETCDN:
GETCXX: MOV     (SP)+, RO    ;;POP STACK INTO RO
        RTS     PC
    
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15 102442 010100
16 102444 004737 102364
17 102450
18 102456 012403
19 102460 005005
20 102462 020127 000020
21 102466 003401
22 102470 012405
23 102472
24 102474 010504
25 102476 012702 000020
26 102502 160102
27 102504 002002
28 102506 062702 000020
29 102512 001414
30 102514 012705 100000
31 102520 005302
32 102522 001402
33 102524 006205
34 102526 000774
35 102530 020127 000020
36 102534 003402
37 102536 040504
38 102540 000401
39 102542 040503
40 102544 004737 102704
41 102550 010546
42 102552 005202
43 102554 005703
44 102556 001372
45 102560 005704
46 102562 001370
47 102564 020027 000012
48 102570 001423
49 102572 010103
50 102574 162700 000014
51 102600 003602
52 102602 012700 000003
53 102606 004737 102704

```

```

;PNTNUM
;
;PRINT A NUMBER
;
;INPUTS:
; R1 - RADIX OF NUMBER
; R2 - ASCII STRING TO COUNT OF BITS IN NUMBER
; R4 - POINTER TO NUMBER (LOW WORD)
;
;OUTPUTS:
; NUMBER IS PRINTED. LEADING ZEROS ARE PRINTED EXCEPT FOR
; DECIMAL NUMBERS (LEFT JUSTIFIED).
;
; R0 - CONTENTS DESTROYED

```

```

PNTNUM. MOV R1,R0 ;SAVE RADIX
        JSR PC,GETCNT ;GET COUNT OF BITS
PNTNUMJS:
        MOV R2,-(SP) ;;PUSH R2 ON STACK
        MOV R3,-(SP) ;;PUSH R3 ON STACK
        MOV R5,-(SP) ;;PUSH R5 ON STACK
        MOV (R4),R2 ;GET ONE PARAMETER WORD
        CLR R3 ;CLEAR STORAGE FOR OTHER
        CMP R1,#16. ;MORE THAN 16 BITS IN NUMBER?
        BLE 1$ ;YES, GET SECOND PARAMETER WORD
        MOV (R4),R5 ;;PUSH R4 ON STACK
        MOV R5,R4 ;;PUT HIGH WORD IN R4
        MOV #16.,R2 ;COMPUTE BITS NOT WANTED
        SUB R1,R2 ;BY SUBTRACTING BITS TO USE
        BGE 2$ ;FROM 16.
        ADD #16.,R2 ;IF NEGATIVE, ADD 16 FOR FIRST WORD
        BEQ 3$ ;IF ZERO, NO BITS NEED BE CLEARED
        MOV #BIT15,R5 ;START MASK WITH SIGN BIT SET
        DEC R2 ;COUNT BITS IN MASK
        BEQ 4$ ;SHIFT MORE BITS TO RIGHT
        ASR R5
        BR 3$ ;MORE THAN 16 BITS IN NUMBER?
        CMP R1,#16. ;YES, CLEAR IN HIGH WORD
        BLE 5$ ;NO, CLEAR IN LOW WORD
        BIC R5,R4 ;DIVIDE BY RADIX IN R0
        JSR PC,DIVIDE ;DIVIDE BY RADIX IN R0
        MOV R5,-(SP) ;;PUSH R5 ON STACK
        INC R2 ;COUNT DIGITS ON STACK
        TST R3 ;CHECK IF QUOTIENT IS ZERO
        BNE 6$
        TST R4
        BNE 6$
        CMP R0,#10. ;IF RADIX IS DECIMAL
        BEQ 10$ ;JUST GO PRINT DIGITS ON STACK
        MOV R1,R3 ;OTHERWISE COMPUTE NUMBER OF LEADING ZEROS
        SUB #12.,R0 ;DIVIDEND IS BITS IN NUMBER
        BGT 7$ ;DIVISOR IS BITS PER DIGIT PRINTED
        MOV #3,R0 ; (3 OR 4)
        JSR PC,DIVIDE

```

54	102612	005705		TST	R5		;IF REMAINDER NOT ZERO
55	102614	001401		BEG	8		;INCREMENT QUOTIENT
56	102616	005203		INC	R3		
57	102620	160203	8	SUB	R2,R3		;SUBTRACT DIGITS ON STACK
58	102622	001406		BEG	10		;NO LEADING ZEROS IF ZERO
59	102624		9				
	102624	112700	000060	MOVB	0,R0		;STORE 0 IN R0 AND
	102630	004737	075506	JSR	PC,PRINTC		;PRINT THE CHARACTER.
60	102634	005303		DEC	R3		
61	102636	001372		BNE	9		;REPEAT UNTIL COUNT REACHES ZERO
62	102640		10				
	102640	012605		MOV	(SP)+,R5		;POP STACK INTO R5
63	102642	062705	000060	ADD	0,R5		;CONVERT TO ASCII DIGIT
64	102646	020527	000071	CMP	R5,0'9		;IF GREATER THAN A 9
65	102652	003402		BLE	11		; CONVERT TO A OR HIGHER
66	102654	062705	000007	ADD	0'A-0'9-1,R5		; FOR HEX DIGIT
67	102660		11				
	102660	110500		MOVB	R5,R0		;STORE R5 IN R0 AND
	102662	004737	075506	JSR	PC,PRINTC		;PRINT THE CHARACTER.
68	102666	005302		DEC	R2		;REPEAT FOR ALL DIGITS
69	102670	001363		BNE	10		; ON STACK
70	102672	012604		MOV	(SP)+,R4		;POP STACK INTO R4
	102674	012605		MOV	(SP)+,R5		;POP STACK INTO R5
	102676	012603		MOV	(SP)+,R3		;POP STACK INTO R3
	102700	012602		MOV	(SP)+,R2		;POP STACK INTO R2
71	102702	000207		RTS	PC		

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17 102704
18 102704 010246
19 102706 012702 000040
20 102712 005005
21 102714 006303
22 102716 006104
23 102720 006105
24 102722 020005
25 102724 101002
26 102726 160005
27 102730 005203
28 102732 005302
29 102734 001367
30 102736 012602
31 102740 000207

;DIVIDE
;
;DIVIDE A 32 BIT UNSIGNED NUMBER BY A 16 BIT UNSIGNED NUMBER.
;REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
;WILL NOT CHECK FOR DIVIDE BY ZERO.
;
;INPUTS:
; R3 - LOW 16 BITS OF DIVIDEND
; R4 - HIGH 16 BITS OF DIVIDEND
; R0 - DIVISOR
;
;OUTPUTS:
; R3 - LOW 16 BITS OF QUOTIENT
; R4 - HIGH 16 BITS OF QUOTIENT
; R5 - REMAINDER

DIVIDE:
MOV R2, -(SP) ;;PUSH R2 ON STACK
MOV #32, R2 ;;SET UP SHIFT COUNT
CLR R5 ;;START WITH ZERO REMAINDER
1$: ASL R3 ;;SHIFT LEFT INTO R5
ROL R4
ROL R5
CMP R0, R5 ;;WILL DIVISOR GO INTO REMAINDER
BHI 2$ ;;ONLY SUBTRACT IF IT WILL
SUB R0, R5 ;;SUBTRACT DIVISOR
INC R3 ;;PUT A ONE INTO QUOTIENT
2$: DEC R2 ;;COUNT THE SHIFTS
BNE 1$
MOV (SP)+, R2 ;;POP STACK INTO R2
RTS PC
    
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 102742
    102742 010046
21 102744 012700 000100
22 102750 005005
23 102752 006301
24 102754 006102
25 102756 006103
26 102760 006104
27 102762 006105
28 102764 022705 000012
29 102770 101003
30 102772 162705 000012
31 102776 005201
32 103000 005300
33 103002 001363
34 103004 012600
35 103006 000207

;DIV10
;
;DIVIDE A 64 BIT UNSIGNED NUMBER BY A 10,
;REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
;WILL NOT CHECK FOR DIVIDE BY ZERO.
;
;INPUTS:
; R1 - LOW 16 BITS OF DIVIDEND
; R2 - NEXT 16 BITS OF DIVIDEND
; R3 - NEXT 16 BITS OF DIVIDEND
; R4 - HIGH 16 BITS OF DIVIDEND
;
;OUTPUTS:
; R1 - QUOTIENT,
; R2 - QUOTIENT,
; R3 - QUOTIENT,
; R4 - QUOTIENT,
; R5 - REMAINDER

DIV10:
MOV     R0, -(SP)           ;; PUSH R0 ON STACK
MOV     #64, R0            ;; SET UP SHIFT COUNT
CLR     R5                  ;; START WITH ZERO REMAINDER

1$:
ASL     R1
ROL     R2                  ;; SHIFT LEFT INTO R5
ROL     R3
ROL     R4
ROL     R5

CMP     #10, R5            ;; WILL DIVISOR GO INTO REMAINDER?
BHI     2$                ;; ONLY SUBTRACT IF IT WILL
SUB     #10, R5           ;; SUBTRACT DIVISOR
INC     R1                 ;; PUT A ONE INTO QUOTIENT
2$:
DEC     R0                 ;; COUNT THE SHIFTS
BNE     1$

MOV     (SP)+, R0          ;; POP STACK INTO R0
RTS     PC                 ;; <R4,R3,R2,R1> AND REMAINDER IN R5

```

```

2
3
4 103010
103010 112700 000040
103014 004737 075506
5 103020 010146
6 103022 000301
7 103024 004737 103050
8 103030 012601
9 103032 004737 103050
10 103036 000207
11
12 103040
103040 112700 000040
103044 004737 075506
13
14
15
16 103050
103050 010146
17 103052 006001
18 103054 006001
19 103056 006001
20 103060 006001
21 103062 004737 103070
22 103066 012601
23 103070 042701 177760
24 103074 062701 000060
25 103100 020127 000071
26 103104 003402
27 103106 062701 000007
28 103112
103112 110100
103114 004737 075506
29 103120 000207

;PRINT HEX NUMBERS WITH LEADING SPACE
T2PNTW:
MOV B #' ,R0 ;STORE #' IN R0 AND
JSR PC,PRINTC ;PRINT THE CHARACTER.
MOV R1,-(SP) ;;PUSH R1 ON STACK
SWAB R1
CALL T2PNT ;PRINT HIGH TWO DIGITS
MOV (SP)+,R1 ;;POP STACK INTO R1
CALL T2PNT ;PRINT LOW TWO DIGITS
RETURN

T2PNTB:
MOV B #' ,R0 ;STORE #' IN R0 AND
JSR PC,PRINTC ;PRINT THE CHARACTER.

;PRINT TWO HEX DIGITS FROM NUMBER IN R1
T2PNT:
MOV R1,-(SP) ;;PUSH R1 ON STACK
RCR R1 ;SHIFT TO GET HIGH DIGIT
ROR R1
ROR R1
ROR R1
CALL T2PNTD ;PRINT TWO DIGITS
MOV (SP)+,R1 ;;POP STACK INTO R1
T2PNTD: BIC #C17,R1 ;CLEAR OTHER BITS
ADD #'0,R1 ;CONVERT TO ASCII CHARACTER
CMP R1,#'9 ;IF GREATER THAN A 9
BLE T2PNTD ; CONVERT TO A OR HIGHER
ADD #'A-'9-1>,R1 ; FOR HEX DIGIT

T2PNTD:
MOV B R1,R0 ;STORE R1 IN R0 AND
JSR PC,PRINTC ;PRINT THE CHARACTER.
RETURN

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13 103122 005000
14 103124 105711
15 103126 001442
16 103130 121127 000040
17 103134 001002
18 103136 005201
19 103140 000770
20 103142 005004
21 103144
    103144 010246
22 103146 112102
23 103150 162702 000060
24 103154 100431
25 103156 020227 000011
26 103162 003410
27 103164 020227 000021
28 103170 103423
29 103172 020227 000026
30 103176 101020
31 103200 162702 000007
32 103204 006304
33 103206 006304
34 103210 006304
35 103212 006304
36 103214 050204
37 103216 005200
38 103220 012602
39 103222 105711
40 103224 001403
41 103226 121127 000040
42 103232 001344
43 103234 005700
44 103236 000207
45
46 103240
    103240 012602
    103242 012600
47 103244 000137 100340

;T2GNUM
;
;GET A HEX DIGIT FROM AN ASCII INPUT STRING
;
;INPUTS:
;   R1 - STRING POINTER
;OUTPUTS:
;   R4 - NUMBER
;   R1 - UPDATED STRING TO CHARACTER AFTER NUMBER
;   R0 - COUNT OF DIGITS (0 IF END OF LINE FOUND)

T2GNUM: CLR     R0                ;CLEAR DIGIT COUNT
        TSTB   (R1)            ;CHECK IF END OF LINE
        BEQ    T2GNX          ;REPORT NULL CHARACTER FOUND
        CMPB   (R1),#' '      ;CHECK IF A SPACE
        BNE    T2GND1        ;IF SO, IGNORE IT
        INC    R1
        BR     T2GNUM

T2GND1: CLR     R4                ;CLEAR NUMBER STORAGE
T2GND2: MOV     R2,-(SP)        ;;PUSH R2 ON STACK
        MOVB   (R1),R2        ;GET CHARACTER
        SUB    #'0,R2         ;CONVERT TO HEX DIGIT
        BMI    T2GNE
        CMP    R2,#9.
        BLE    T2GND3
        CMP    R2,#<'A-'0>
        BLO    T2GNE
        CMP    R2,#<'F-'0>
        BHI    T2GNE
        SUB    #'A-'9-1>,R2
T2GND3: ASL    R4
        ASL    R4
        ASL    R4
        ASL    R4
        BIS    R2,R4
        INC    R0
        MOV    (SP),R2        ;;POP STACK INTO R2
        TSTB   (R1)
        BEQ    T2GNX
        CMPB   (R1),#' '
        BNE    T2GND2
T2GNX:  TST    R0
        RETURN

T2GNE:  MOV    (SP),R2        ;;POP STACK INTO R2
        MOV    (SP),R0        ;;POP STACK INTO R0
        JMP    T2CMDE

```

USER DOCUMENTATION	....B1	USER DOCUMENTATION	....B5	USER DOCUMENTATION	....B9	GLOBAL SUBROUTINES S....	B13
USER DOCUMENTATION	....C1	USER DOCUMENTATION	....C5	USER DOCUMENTATION	....C9	GLOBAL SUBROUTINES S....	C13
USER DOCUMENTATION	....D1	USER DOCUMENTATION	....D5	USER DOCUMENTATION	....D9	GLOBAL SUBROUTINES S....	D13
USER DOCUMENTATION	....E1	USER DOCUMENTATION	....E5	USER DOCUMENTATION	....E9	GLOBAL SUBROUTINES S....	E13
USER DOCUMENTATION	....F1	USER DOCUMENTATION	....F5	USER DOCUMENTATION	....F9	GLOBAL SUBROUTINES S....	F13
USER DOCUMENTATION	....G1	USER DOCUMENTATION	....G5	USER DOCUMENTATION	....G9	GLOBAL SUBROUTINES S....	G13
USER DOCUMENTATION	....H1	USER DOCUMENTATION	....H5	USER DOCUMENTATION	....H9	GLOBAL SUBROUTINES S....	H13
USER DOCUMENTATION	....I1	USER DOCUMENTATION	....I5	USER DOCUMENTATION	....I9	GLOBAL SUBROUTINES S....	I13
USER DOCUMENTATION	....J1	USER DOCUMENTATION	....J5	USER DOCUMENTATION	....J9	GLOBAL SUBROUTINES S....	J13
USER DOCUMENTATION	....K1	USER DOCUMENTATION	....K5	PROGRAM HEADER	....K9	GLOBAL SUBROUTINES S....	K13
USER DOCUMENTATION	....L1	USER DOCUMENTATION	....L5	PROGRAM HEADER	....L9	GLOBAL SUBROUTINES S....	L13
USER DOCUMENTATION	....M1	USER DOCUMENTATION	....M5	DISPATCH TABLE	....M9	GLOBAL SUBROUTINES S....	M13
USER DOCUMENTATION	....N1	USER DOCUMENTATION	....N5	DEFAULT HARDWARE P-T....	N9	GLOBAL SUBROUTINES S....	N13
USER DOCUMENTATION	....B2	USER DOCUMENTATION	....B6	SOFTWARE P-TABLE	....B10	GLOBAL SUBROUTINES S....	B14
USER DOCUMENTATION	....C2	USER DOCUMENTATION	....C6	GLOBAL EQUATES SECTI....	C10	GLOBAL SUBROUTINES S....	C14
USER DOCUMENTATION	....D2	USER DOCUMENTATION	....D6	GLOBAL EQUATES SECTI....	D10	GLOBAL SUBROUTINES S....	D14
USER DOCUMENTATION	....E2	USER DOCUMENTATION	....E6	UDA BIT DEFINITIONS	....E10	GLOBAL SUBROUTINES S....	E14
USER DOCUMENTATION	....F2	USER DOCUMENTATION	....F6	UDA BIT DEFINITIONS	....F10	GLOBAL SUBROUTINES S....	F14
USER DOCUMENTATION	....G2	USER DOCUMENTATION	....G6	HOT COMMUNICATION A....	G10	GLOBAL SUBROUTINES S....	G14
USER DOCUMENTATION	....H2	USER DOCUMENTATION	....H6	HOT COMMUNICATION A....	H10	GLOBAL SUBROUTINES S....	H14
USER DOCUMENTATION	....I2	USER DOCUMENTATION	....I6	COMMAND PACKET OPCOD....	I10	GLOBAL SUBROUTINES S....	I14
USER DOCUMENTATION	....J2	USER DOCUMENTATION	....J6	COMMAND PACKET OPCOD....	J10	GLOBAL SUBROUTINES S....	J14
USER DOCUMENTATION	....K2	USER DOCUMENTATION	....K6	COMMAND PACKET OFFSE....	K10	GLOBAL SUBROUTINES S....	K14
USER DOCUMENTATION	....L2	USER DOCUMENTATION	....L6	END PACKET OFFSETS	....L10	GLOBAL SUBROUTINES S....	L14
USER DOCUMENTATION	....M2	USER DOCUMENTATION	....M6	END PACKET OFFSETS	....M10	GLOBAL SUBROUTINES S....	M14
USER DOCUMENTATION	....N2	USER DOCUMENTATION	....N6	STATUS AND EVENT COD....	N10	GLOBAL SUBROUTINES S....	N14
USER DOCUMENTATION	....H3	USER DOCUMENTATION	....B7	CONTROLLER TABLE DEF....	B11	GLOBAL SUBROUTINES S....	B15
USER DOCUMENTATION	....C3	USER DOCUMENTATION	....C7	CONTROLLER TABLE DEF....	C11	GLOBAL SUBROUTINES S....	C15
USER DOCUMENTATION	....D3	USER DOCUMENTATION	....D7	CONTROLLER TABLE DEF....	D11	GLOBAL SUBROUTINES S....	D15
USER DOCUMENTATION	....E3	USER DOCUMENTATION	....E7	GLOBAL DATA SECTION	....E11	GLOBAL SUBROUTINES S....	E15
USER DOCUMENTATION	....F3	USER DOCUMENTATION	....F7	GLOBAL DATA SECTION	....F11	GLOBAL SUBROUTINES S....	F15
USER DOCUMENTATION	....G3	USER DOCUMENTATION	....G7	GLOBAL DATA SECTION	....G11	GLOBAL SUBROUTINES S....	G15
USER DOCUMENTATION	....H3	USER DOCUMENTATION	....H7	GLOBAL TEXT SECTION	....H11	GLOBAL SUBROUTINES S....	H15
USER DOCUMENTATION	....I3	USER DOCUMENTATION	....I7	GLOBAL TEXT SECTION	....I11	GLOBAL SUBROUTINES S....	I15
USER DOCUMENTATION	....J3	USER DOCUMENTATION	....J7	GLOBAL TEXT SECTION	....J11	PRE-PROGRAMMED SUBRO....	J15
USER DOCUMENTATION	....K3	USER DOCUMENTATION	....K7	GLOBAL TEXT SECTION	....K11	PRE-PROGRAMMED SUBRO....	K15
USER DOCUMENTATION	....L3	USER DOCUMENTATION	....L7	GLOBAL ERROR REPORT	....L11	PRE-PROGRAMMED SUBRO....	L15
USER DOCUMENTATION	....M3	USER DOCUMENTATION	....M7	GLOBAL ERROR REPORT	....M11	PRE-PROGRAMMED SUBRO....	M15
USER DOCUMENTATION	....N3	USER DOCUMENTATION	....N7	GLOBAL ERROR REPORT	....N11	PRE-PROGRAMMED SUBRO....	N15
USER DOCUMENTATION	....B4	USER DOCUMENTATION	....B8	GLOBAL ERROR REPORT	....B12	PRE-PROGRAMMED SUBRO....	B16
USER DOCUMENTATION	....C4	USER DOCUMENTATION	....C8	GLOBAL ERROR REPORT	....C12	PRE-PROGRAMMED SUBRO....	C16
USER DOCUMENTATION	....D4	USER DOCUMENTATION	....D8	GLOBAL ERROR REPORT	....D12	PRE-PROGRAMMED SUBRO....	D16
USER DOCUMENTATION	....E4	USER DOCUMENTATION	....E8	GLOBAL SUBROUTINES S....	E12	PRE-PROGRAMMED SUBRO....	E16
USER DOCUMENTATION	....F4	USER DOCUMENTATION	....F8	GLOBAL SUBROUTINES S....	F12	PRE-PROGRAMMED SUBRO....	F16
USER DOCUMENTATION	....G4	USER DOCUMENTATION	....G8	GLOBAL SUBROUTINES S....	G12	PRE-PROGRAMMED SUBRO....	G16
USER DOCUMENTATION	....H4	USER DOCUMENTATION	....H8	GLOBAL SUBROUTINES S....	H12	PRE-PROGRAMMED SUBRO....	H16
USER DOCUMENTATION	....I4	USER DOCUMENTATION	....I8	GLOBAL SUBROUTINES S....	I12	PRE-PROGRAMMED SUBRO....	I16
USER DOCUMENTATION	....J4	USER DOCUMENTATION	....J8	GLOBAL SUBROUTINES S....	J12	PRE-PROGRAMMED SUBRO....	J16
USER DOCUMENTATION	....K4	USER DOCUMENTATION	....K8	GLOBAL SUBROUTINES S....	K12	PRE-PROGRAMMED SUBRO....	K16
USER DOCUMENTATION	....L4	USER DOCUMENTATION	....L8	GLOBAL SUBROUTINES S....	L12	PRE-PROGRAMMED SUBRO....	L16
USER DOCUMENTATION	....M4	USER DOCUMENTATION	....M8	GLOBAL SUBROUTINES S....	M12	PRE-PROGRAMMED SUBRO....	M16
USER DOCUMENTATION	....N4	USER DOCUMENTATION	....N8	GLOBAL SUBROUTINES S....	N12	PRE-PROGRAMMED SUBRO....	N16

```

1
3
4
5
6
7
8
9
10
11
12
13
14
15 103250
103250 010046
103252 010146
103254 010246
16 103256 005764 000004
17 103262 002004
18 103264 116537 000002 002074
19 103272 000416
20 103274
103274 010446
21 103276 016401 000004
22 103302 004737 102274
23 103306 001036
24 103310 005764 000002
25 103314 100004
26 103316 052713 100000
32 103322 012604
33 103324 000423
34 103326
103326 012504
35 103330 012702 064402
36 103334 016412 000002
37 103340 006112
38 103342 006112
39 103344 006112
40 103346 042722 177774
41 103352 016412 000002
42 103356 042722 140000
43 103362 005022
44 103364 012712 075250
45 103370 104460
46 103372 000241
47 103374
103374 012602
103376 012601
103400 012600
48 103402 000207
49 103404 000261
50 103406 000772

;PNTERR
;PRINT ERROR MESSAGE FROM DM PROGRAM REQUEST 11 OR 12.
;INPUTS:
; R5 - CONTROLLER TABLE ADDRESS
; R4 - MESSAGE DATA ADDRESS
; R3 - COMMAND DATA ADDRESS
;OUTPUTS:
; ERROR MESSAGE PRINTED
; BIT 15 SET IN COMMAND DATA IF DRIVE HAS BEEN DROPPED

PNTERR:
MOV R0,-(SP) ;PUSH R0 ON STACK
MOV R1,-(SP) ;PUSH R1 ON STACK
MOV R2,-(SP) ;PUSH R2 ON STACK
TST 4(R4) ;GET DRIVE NUMBER
BGE 1$ ;CHECK IF BIT 15 SET
MOVB C.UNIT(R5),L$LUN ;IF SO, GET UNIT FROM CONTROLLER TABLE
BR 2$

1$:
MOV R4,-(SP) ;PUSH R4 ON STACK
MOV 4(R4),R1 ;GET DRIVE NUMBER
CALL GTDRV ;GET DRIVE TABLE ADDRESS
BNE 3$ ;IF UNIT DROPPED, EXIT
TST D.UNIT(R4) ;SEE IF UNIT HAS BEEN DROPPED FROM TESTING
BPL 3$ ;PROCEED IF STILL TO BE TESTED
BIS #BIT15,(R3) ;TELL DM PROGRAM TO STOP TESTING THIS UNIT
MOV (SP)+,R4 ;POP STACK INTO R4
BR 4$

3$:
MOV (SP)+,R4 ;POP STACK INTO R4
2$:
MOV #ERRTYP,R2 ;GET POINTER TO ERROR TABLE
MOV 2(R4),(R2) ;GET ERROR TYPE
ROL (R2)
ROL (R2)
ROL (R2)
BIC #1C3,(R2)+ ;CLEAR LOW 2 BITS
MOV 2(R4),(R2)
BIC #140000,(R2)+ ;MASK LOW 14 BITS
CLR (R2)+ ;CLEAR MESSAGE POINTER
MOV #ERR.TN,(R2) ;GET ROUTINE NUMBER
TRAP C$ERROR ;DRIVE HAS NOT BEEN DROPPED
CLC

4$:
MOV (SP)+,R2 ;POP STACK INTO R2
MOV (SP)+,R1 ;POP STACK INTO R1
MOV (SP)+,R0 ;POP STACK INTO R0
RETURN

5$:
SEC ;DRIVE HAS BEEN DROPPED
BR 4$

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56

104612			
104612	010346		
104614	010400		
104616	000300		
104620	042700	177770	
104624	004737	105550	
104630	010102		
104632	010400		
104634	000300		
104636	006000		
104640	006000		
104642	006000		
104644	042700	177770	
104650	004737	105550	
104654	060201		
104656	006301		
104660	062701	000002	
104664	020137	064414	
104670	101402		
104672	000137	075410	
104676	013702	064412	
104702	010103		
104704	012722	177777	
104710	005303		
104712	003374		
104714	004737	105070	

```

;UDAIN
;
;FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO INITIALIZE A UDA AND BRING IT ON-LINE.
;   ALL STEPS ARE CHECKED. AN ERROR MESSAGE IS REPORTED IF ANY ERROR
;   DETECTED.
;
;INPUTS:
;   R5 - ADDRESS OF CONTROLLER TABLE.
;   R4 - LENGTH, INTERRUPT AND VECTOR FIELDS TO SEND TO UDA
;IMPLICIT INPUTS:
;   FFREE - FIRST FREE ADDRESS OF MEMORY. THIS ADDRESS IS GIVEN TO UDA
;           AS START OF RING BUFFER.
;   FSIZE - SIZE OF FREE MEMORY AVAILABLE IN WORDS.
;OUTPUTS:
;   R1 - SIZE OF RING BUFFER IN WORDS IF NO ERROR,
;   R4 - ADDRESS OF UDAIP REGISTER IN UDA,
;   R5 - UNCHANGED,
;
;   Z CLR  IF NO ERROR,
;   Z SET  IF ANY ERROR REPORTED
;
;CHECK IF ENOUGH FREE MEMORY FOR RING BUFFER

```

```

UDAIN:
MOV     R3, -(SP)           ;PUSH R3 ON STACK
MOV     R4, R0              ;GET MESSAGE LENGTH
SWAB   R0
BIC     @177770, R0
JSR     PC, CLOG            ;COMPUTE LOGARITHMIC VALUE
MOV     R1, R2              ;SAVE RESULT IN R2
MOV     R4, R0              ;GET COMMAND LENGTH
SWAB   R0
ROR     R0
ROR     R0
ROR     R0
BIC     @177770, R0
JSR     PC, CLOG            ;COMPUTE LOGARITHMIC VALUE
ADD     R2, R1              ;ADD THE TWO RESULTS
ASL     R1                  ;MULTIPLY BY 2 WORDS PER RING
ADD     @<HC.ISZ>/2, R1     ;ADD SPACE FOR INTERRUPT INDICATORS
LMP     R1, FSIZE           ;COMPARE WITH SIZE OF FREE MEMORY
BLOS   1#                   ;FATAL ERROR IF NOT ENOUGH MEMORY
JMP     FMERR
;FILL HOST COMMUNICATION AREA WITH ALL ONES
11:    MOV     FFREE, R2     ;GET FIRST ADDRESS OF RING BUFFER
MOV     R1, R3              ;GET SIZE OF RING BUFFER
21:    MOV     @-1, (R2)+    ;WRITE ONES TO BUFFER
DEC     R3                  ;COUNT THE WORDS IN BUFFER
BGT     2#                   ;LOOP UNTIL ENTIRE BUFFER WRITTEN
;DO THE INITIALIZATION
JSR     PC, UDAIS1          ;DO FIRST THREE STEPS

```

```

57 104720 103460          BCS      9$          ;GET OUT IF UDA MICROCODE REPORTED FAILURE
58 104722 012364 000002   MOV      (R3)+,2(R4)   ;WRITE NEXT WORD TO UDASA REGISTER
59 104726 012700 000310   MOV      #200.,R0     ;GET TRY COUNTER
60 104732 016402 000002   3$:    MOV      2(R4),R2 ;LOOK AT UDASA
61 104736 001410         BEQ      5$
62 104740 100005         SPL      4$
63 104742 104455         TRAP    C$ERDF
    104744 000030         .WORD   24
    104746 000000         .WORD   0
    104750 074602         .WORD  ERR024
64 104752 000443         BR      5$
65
66 104754 005300         4$:    DEC      R0
67 104756 001365         BNE     3$
68 104760 010264 000002   5$:    MOV      R2,2(R4)     ;WRITE 0 TO UDASA (PURGE)
69 104764 011402         MOV      (R4),R2      ;READ FROM UDAIP (POLL)
70 104768 004737 105410   JSR     PC,UDARSP     ;WAIT FOR STEP OR ERROR BIT
71 104772 103433         BCS     9$          ;GET OUT IF UDA MICROCODE REPORTED FAILURE
72 104774 010146         MOV     R1,-(SP)      ;PUSH R1 ON STACK
73 104776 004733         JSR     PC,B(R3)+     ;CALL LAST ROUTINE
74 105000 012601         MOV     (SP)+,R1     ;POP STACK INTO R1
75
76          ;CHECK HOST COMMUNICATION AREA FOR ALL ZEROS
77
78 105002 013702 064412   MOV     FFREE,R2     ;GET FIRST ADDRESS OF RING BUFFER
79 105006 010103         MOV     R1,R3        ;GET SIZE OF RING BUFFER
80 105010 005722         6$:    TST     (R2)+        ;CHECK WORD IN BUFFER
81 105012 001003         BNE     7$          ;GO TO ERROR REPORTER IF NOT ZERO
82 105014 005303         DEC     R3           ;COUNT THE WORDS IN BUFFER
83 105016 003374         BGT     6$          ;LOOP UNTIL ALL WORDS CHECKED
84 105020 000405         BR      8$
85
86 105022         7$:    TRAP    C$ERDF
    105022 104455         .WORD   23
    105024 000027         .WORD   0
    105026 000000         .WORD   0
    105030 074516         .WORD  ERR023
87 105032 000413         BR      9$
88
89          ;SEND GO BIT TO UDASA REGISTER TO END INITIALIZATION
90
91
100 105034 016500 000006   8$:    MOV     C.BST(R5),R0 ;GET BURST VALUE
102 105040 006300         ASL     R0           ;SHIFT TO POSITION
103 105042 006300         ASL     R0
104 105044 052700 000001   BIC     #5A.GO,R0    ;SET THE GO BIT
105 105050 010064 000002   MOV     R0,2(R4)     ;SEND TO UDA
106 105054 012603         MOV     (SP)+,R3     ;POP STACK INTO R3
107 105056 000244         CLZ
108 105060 000207         RTS     PC          ;CLEAR Z AS NO ERROR INDICATION
109
110          ;ERROR RETURN
111
112 105062         9$:    MOV     (SP)+,R3     ;POP STACK INTO R3
    105062 012603         SEZ
113 105064 000264         RTS     PC          ;SET Z TO INDICATE ERROR OCCURRED
114 105066 000207
    
```

E1

PRE-PROGRAMMED SUBROUTINES

```

1
2
3      ;LOAD DM PROGRAM FROM MEMORY SPACE TESTED DURING
4      ;INITIALIZATION IN TEST 1
5 103674 017704 160532 LOADT1: MOV    @DMPROG,R4      ;GET SIZE OF DM PROGRAM IN BYTES
6 103700 162704 000040      SUB    @DMMAIN,R4
7 103704 013700 064432      MOV    @DMPROG,R0      ;GET ADDRESS OF DM PROGRAM
8 103710 062700 000040      ADD    @DMMAIN,R0
9 103714 005001      CLR    R1              ;START WITH OFFSET OF ZERO
10
11 103716 012703 000214 LT1L1: MOV    @<HC.BSZ*2>,R3      ;GET SIZE OF BOTH BUFFERS
12 103722 020403      CMP    R4,R3          ;IF FEWER BYTES REMAINING IN PROGRAM
13 103724 103001      BHS   LT11
14 103726 010403      MOV    R4,R3          ;USE ACTUAL BYTE COUNT
15 103730
16 103730 010746      MOV    R3,-(SP)       ;PUSH R3 ON STACK
17 103732 013702 064412      MOV    @FFREE,R2      ;GET ADDRESS OF BUFFER
18 103736 162702 000214      SUB    @<HC.BSZ*2>,R2
19 103742 010246      MOV    R2,-(SP)       ;PUSH R2 ON STACK
20 103744 012022      LT1L2: MOV    (R0)+,(R2)+    ;MOVE DATA TO BUFFER
21 103746 162703 000002      SUB    @2,R3          ;COUNT BYTES
22 103752 001374      BNE   LT1L2
23 103754 012602      MOV    (SP)+,R2       ;POP STACK INTO R2
24 103756 012603      MOV    (SP)+,R3       ;POP STACK INTO R3
25 103760 004737 104006      CALL   @LOAD          ;LOAD INTO UDA
26 103764 001455      BEQ   @LOADER        ;IF ERROR, GET OUT NOW
27 103766 006203      ASR   R3              ;CONVERT BYTES TO WORDS
28 103770 060301      ADD   R3,R1          ;INCREASE OFFSET FOR NEXT BUFFER
29 103772 006303      ASL   R3              ;CONVERT WORDS TO BYTES
30 103774 160304      SUB   R3,R4          ;REDUCE REMAINING BYTE COUNT
31 103776 001347      BNE   LT1L1          ;GET NEXT BUFFER
32 104000 012701 000040      MOV    @DMMAIN,R1    ;GET A BYTE COUNT OF HEADER ONLY
33 104004 000675      BR    @LOADB         ;NOW START

```

```

49 105244 103412          BCS      4$          ;GET OUT IF ERROR
50 105246 006337 105546   ASL      UDARSD       ;SHIFT TO NEXT STEP BIT
51 105252 032737 040000 105546   BIT      @SA.S4,UDARSD ;CHECK IF NOW AT STEP 4
52 105260 001003          BNE      3$          ;GET OUT IF SO
53 105262 012364 000002   MOV      (R3)+,2(R4)   ;WRITE DATA TO UDASA REGISTER
54 105266 000762          BR       2$          ;STAY IN LOOP
55
56 105270 000241          3$:     CLC              ;CLEAR CARRY FOR NO ERROR INDICATION
57 105272          4$:     MOV      (SP)+,R1      ;POP STACK INTO R1
105272 012601          RTS      PC
105274 000207
58 105274 000207          ;DATA TO BE SENT AND RECEIVED BY UDA INITIALIZA'ION
59
60
61
62 105276 105314          INITBL: .WORD   RSP.S1      ;1ST WORD RESPONSE CHECK ROUTINE
63 105300 000000          SND.S1: .WORD   0          ;1ST WORD TO SEND TO UDASA
64 105302 105322          .WORD   RSP.S2      ;2ND WORD RESPONSE CHECK ROUTINE
65 105304 000000          SND.S2: .WORD   0          ;2ND WORD TO SEND TO UDASA
66 105306 105342          .WORD   RSP.S3      ;3RD WORD RESPONSE CHECK ROUTINE
67 105310 000000          SND.S3: .WORD   0          ;3RD WORD TO SEND TO UDASA
68 105312 105360          .WORD   RSP.S4      ;4TH WORD RESPONSE CHECK ROUTINE
69
70          ;RESPONSE CHECK FOR FIRST WORD (STEP 1) FROM UDASA
71          ;CHECK FOR PROPER CONTROLLER TYPE
72
73 105314 012701 004400   RSP.S1: MOV      @SA.S1+SA.DI,R1   ;SET STEP ONE BIT
74 105320 000422          BR       RSP.CK          ;NOW DO A RESPONSE CHECK
75
76          ;RESPONSE CHECK FOR SECOND WORD (STEP 2) FROM UDASA
77          ;CHECK FOR ECHO OF INTI AND VECTOR
78
79 105322 013701 105300   RSP.S2: MOV      SND.S1,R1        ;GET WORD SENT TO UDASA
80 105326 000301          SWAB     R1              ;GET HIGH 8 BITS
81 105330 042701 177400   BIC      @177400,R1
82 105334 052701 010000   BIS      @SA.S2,R1
83 105340 000412          BR       RSP.CK          ;SET STEP 2 BIT
84          ;NOW DO A RESPONSE CHECK
85
86          ;RESPONSE CHECK FOR THIRD WORD (STEP 3) FROM UDASA
87          ;CHECK FOR ECHO OF MESSAGE AND COMMAND RING LENGTHS
88
88 105342 013701 105300   RSP.S3: MOV      SND.S1,R1        ;GET WORD SENT TO UDASA
89 105346 042701 177400   BIC      @177400,R1        ;JUST LOW 8 BITS
90 105352 052701 020000   BIS      @SA.S3,R1
91 105356 000403          BR       RSP.CK          ;SET STEP 3 BIT
92          ;NOW DO A RESPONSE CHECK
93
94          ;RESPONSE CHECK FOR FOURTH WORD (STEP 4) FROM UDASA
95          ;CHECK FOR ECHO OF PURGE AND L'FAIL BITS
96
96 105360 010201          RSP.S4: MOV      R2,R1          ;GET RESPONSE FROM UDA AND
97 105362 010237 105406   MOV      R2,SSTEP4       ;SAVE STEP 4 VALUE.
98
99          ;RESPONSE CHECK, COMPARE EXPECTED DATA IN R1 WITH ACTUAL DATA IN R2
100
101 105366 020102          RSP.CK: CMP      R1,R2
102 105370 001405          BEQ      1$
103          ;COMPARE THE DATA
104          ;EXIT IF COMPARED CORRECTLY
          ;ERROR, 'UDA DID NOT RETURN CORRECT DATA IN
          ;UDASA REGISTER DURING INITIALIZATION'

```

G2

105	105372	104455	TRAP	C#ERDF	
	105374	000031	.WORD	25	
	105376	000000	.WORD	0	
	105400	074616	.WORD	ERR025	
106	105402	000261	SEC		
107	105407	000207	1\$: RTS	PC	
108					
109	105406	000000	SSTEP4: .WORD	0	;SAVE STEP 4 VALUE HERE

T\$EXCP	207-66#	207-73	207-73#											
	130-64	130-64#	218-65	218-65#	218-67	218-67#	218-69	218-69#	218-71	218-71#	218-73	218-73#	220-12	220-12#
	220-15	220-15#												
T\$FLAG	186-169	186-169#	186-169#	197-31	197-31	197-31#	197-31#	200-28	200-28	200-28#	200-28#	201-19	201-19#	201-19#
	202-20	202-20#	202-20#	203-70	203-70	203-70#	203-70#	216-54	216-54	216-54#	216-54#			
T\$FREE	220-100	221-24#												
T\$CMAN	83-374#	130-64	130-64#	130-64#										
T\$HILI	130-64	130-64#	218-65	218-65#	218-67	218-67#	218-69	218-69#	218-71	218-71#	218-73	218-73#	220-12	220-12#
	220-15	220-15#												
T\$LAST	83-374#	220-100#	221-15											
T\$LOLI	130-64	130-64#	218-65	218-65#	218-67	218-67#	218-69	218-69#	218-71	218-71#	218-73	218-73#	220-12	220-12#
	220-15	220-15#												
T\$LSYM	83-374	83-374#	86-27	87-30	106-28	106-32	106-36	106-40	106-45	106-49	106-54	106-58	106-62	106-84
	106-89	106-103	106-107	106-111	106-116	106-120	106-124	106-128	106-133	106-137	106-142	106-146	106-150	106-155
	106-159	106-163	106-167	106-214	171-13	172-22	181-10	181-14	186-200	198-43	199-18	200-43	201-34	202-35
	204-17	205-16	206-33	207-79	208-5	209-8	210-9	210-29	211-40	212-34	216-73	218-79	220-32	
T\$LTNO	220-100#													
T\$NEST	83-374#	83-401	83-401	83-411#	86-10	86-10	86-10#	86-27	86-27	86-27	86-27#	87-10	87-10	87-10#
	87-30	87-30	87-30	87-30#	87-31	87-31	87-31	87-31#	83-51	88-51	88-51#	106-26	106-26	106-26#
	106-28	106-28	106-28	106-28#	106-30	106-30	106-30#	106-32	106-32	106-32	106-32#	106-34	106-34	106-34#
	106-36	106-36	106-36	106-36#	106-38	106-38	106-38#	106-40	106-40	106-40	106-40#	106-43	106-43	106-43#
	106-45	106-45	106-45	106-45#	106-47	106-47	106-47#	106-49	106-49	106-49	106-49#	106-52	106-52	106-52#
	106-54	106-54	106-54	106-54#	106-56	106-56	106-56#	106-58	106-58	106-58	106-58#	106-60	106-60	106-60#
	106-62	106-62	106-62	106-62#	106-75	106-75	106-75#	106-84	106-84	106-84	106-84#	106-86	106-86	106-86#
	106-89	106-89	106-89	106-89#	106-91	106-91	106-91#	106-103	106-103	106-103	106-103#	106-105	106-105	106-105#
	106-107	106-107	106-107	106-107#	106-109	106-109	106-109#	106-111	106-111	106-111	106-111#	106-114	106-114	106-114#
	106-116	106-116	106-116	106-116#	106-118	106-118	106-118#	106-120	106-120	106-120	106-120#	106-122	106-122	106-122#
	106-124	106-124	106-124	106-124#	106-126	106-126	106-126#	106-128	106-128	106-128	106-128#	106-131	106-131	106-131#
	106-133	106-133	106-133	106-133#	106-135	106-135	106-135#	106-137	106-137	106-137	106-137#	106-139	106-139	106-139#
	106-142	106-142	106-142	106-142#	106-144	106-144	106-144#	106-146	106-146	106-146	106-146#	106-148	106-148	106-148#
	106-150	106-150	106-150	106-150#	106-152	106-152	106-152#	106-155	106-155	106-155	106-155#	106-157	106-157	106-157#
	106-159	106-159	106-159	106-159#	106-161	106-161	106-161#	106-163	106-163	106-163	106-163#	106-165	106-165	106-165#
	106-167	106-167	106-167	106-167#	106-193	106								

```

108 107062 016503 000204      MOV      D.SERN+4(R5),R3
109 107066 005004      CLR      R4
110 107070 004737 102742      2$:     CALL     DIV10          ;DIVIDE BY 10
111 107074 062705 000060      ADD      #0,R5          ;CONVERT TO ASCII CHARACTER
112 107100 110540      MOV8     R5,-(R0)       ;PUT DIGIT INTO TEMP STORAGE
113 107102 010146      MOV      R1,-(SP)
114 107104 050216      BIS      R2,(SP)
115 107106 050316      BIS      R3,(SP)
116 107110 050426      BIS      R4,(SP)+
117 107112 001366      BNE     2$             ;SEE IF QUOTIENT IS ZERO
118 107114 012601      MOV      (SP)+,R1
119 107116 016146 000164      MOV      D.XFRW(R1),-(SP) ;IF NOT, DIVIDE AGAIN
107122 016146 000166      MOV      D.XFRR(R1),-(SP) ;POP STACK INTO R1
107126 016146 000174      MOV      D.SEEK(R1),-(SP)
107132 012746 064476      MOV      @TEMP,-(SP)
107136 011146      MOV      (R1),-(SP)
107140 016146 000002      MOV      D.UNIT(R1),-(SP)
107144 012746 107534      MOV      @RPTMSD,-(SP)
107150 012746 000007      MOV      @7,-(SP)
107154 010600      MOV      SP,R0
107156 104416      TRAP     C:PNTS
107160 062706 000020      ADD      #20,SP
121 107164 016146 000176      MOV      D.ECCC(R1),-(SP)
107170 016146 000172      MOV      D.SERR(R1),-(SP)
107174 016146 000170      MOV      D.HERR(R1),-(SP)
107200 012746 107603      MOV      @RPTMD2,-(SP)
107204 012746 000004      MOV      @4,-(SP)
107210 010600      MOV      SP,R0
107212 104416      TRAP     C:PNTS
107214 062706 000012      ADD      #12,SP
145 107220 012605      MOV      (SP)+,R5        ;POP STACK INTO R5
107222 012604      MOV      (SP)+,R4        ;POP STACK INTO R4
107224 012603      MOV      (SP)+,R3        ;POP STACK INTO R3
146 107226 005303      RPTDTN: DEC      R3
147 107230 003265      BGT     RPTD           ;COUNT THE DRIVE TABLES
148 107232 062705 000046      RPTCTN: ADD     @C.SIZE,R5 ;REPEAT FOR ALL DRIVE TABLES
149 107236 005715      TST     (R5)           ;GO TO NEXT CONTROLLER TABLE
154 107240 001251      BNE     RPTCT
156 107242      RPTXX:  MOV      (SP)+,R5        ;POP STACK INTO R5
107244 012604      MOV      (SP)+,R4        ;POP STACK INTO R4
107246 012603      MOV      (SP)+,R3        ;POP STACK INTO R3
107250 012602      MOV      (SP)+,R2        ;POP STACK INTO R2
107252 012601      MOV      (SP)+,R1        ;POP STACK INTO R1
107254 012600      MOV      (SP)+,R0        ;POP STACK INTO R0
168
169 107256 000167      .WORD   J$JMP
107260 000344      .WORD   L10042-2-.
170
174 107262      116      042      124 RPTMSG: .ASCIZ  \N"TEST "D3" IN PROGRESS. "\
175 107316      116      042      125 RPTMSH: .ASCII  \N"UNIT DRIVE      SERIAL-NUMBER SEEKS MBYTES MBYTES HARD SOFT ECC"N\
176 107430      042      040      040      .ASCIZ  \"
177 107534      045      123      062 RPTMSD: .ASCIZ  \#S2#D2#S3#D3#S1#T#S1#D5#S2#D5#S3#D5#S2\
178 107603      045      104      065 RPTMD2: .ASCIZ  \#D5#S2#D5#S1#D5#N\
198
199
200 107626      L10042:

```

```

1
2
3      ;CLOG
4      ;
5      ; COMPUTE LOGARITHMIC VALUE OF NUMBER TO BASE 2.
6      ;
7      ; INPUTS:
8      ;       R0 - LOGARITHM TO BE CONVERTED
9      ;
10     ; OUTPUTS:
11     ;       R1 - VALUE OF 2 RAISED TO POWER OF INPUT NUMBER
12
13     CLOG:
14         MOV     R0, -(SP)           ; PUSH R0 ON STACK
15         CLR     R1                 ; SET UP ZERO START VALUE
16         SEC
17         ; WITH CARRY READY TO SHIFT IN
18     1$:     ROL     R1                 ; SHIFT TO LEFT
19         DEC     R0                 ; UNTIL R0
20         BPL     1$                 ; GOES NEGATIVE
21         MOV     (SP)+, R0          ; POP STACK INTO R0
22         RTS     PC

```

```

1
2
3
4
5
6
7
8
9
10
11 105550
12 105550 010046
13 105552 005001
14 105554 000261
15 105556 006101
16 105560 005300
17 105562 100375
18 105564 012600
19 105566 000207

;CLOG
;
; COMPUTE LOGARITHMIC VALUE OF NUMBER TO BASE 2.
;
; INPUTS:
; R0 - LOGARITHM TO BE CONVERTED
;
; OUTPUTS:
; R1 - VALUE OF 2 RAISED TO POWER OF INPUT NUMBER

CLOG:
MOV    R0, -(SP)      ; PUSH R0 ON STACK
CLR    R1              ; SET UP ZERO START VALUE
SEC    R1              ; WITH CARRY READY TO SHIFT IN
1$:   ROL    R1        ; SHIFT TO LEFT
      DEC    R0        ; UNTIL R0
      BPL    1$        ; GOES NEGATIVE
      MOV    (SP)+, R0 ; POP STACK INTO R0
      RTS    PC

```

```

1
2      ;APRINT
3
4      ;CONVERT AN 18 BIT ADDRESS STORED IN TWO WORDS INTO A FORMAT
5      ;THAT WILL ALLOW PRINTING OF THE 18 BIT NUMBER.
6
7      ;INPUTS:
8      ;      RO - ADDRESS OF TWO WORD BLOCK CONTAINING ADDRESS.
9      ;              FIRST WORD CONTAINING LOW 16 BITS.
10     ;              SECOND WORD CONTAINING HIGH 2 BITS.
11     ;OUTPUTS:
12     ;      R1 - HIGH 3 BITS OF ADDRESS
13     ;      R2 - LOW 15 BITS OF ADDRESS
14
15 104472 016001 000002  APRINT: MOV     2(RO),R1      ;GET HIGH 2 BITS
16 104476 006301          ASL     R1              ;SHIFT LEFT
17 104500 011002          MOV     (RO),R2          ;GET LOW 16 BITS
18 104502 100001          BPL     APRIZ           ;IF 16TH BIT SET
19 104504 005201          INC     R1              ;PLACE IT IN WITH HIGH 2 BITS
20 104506 000207          APRIZ:  RETURN

```



```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19 104520
20 104520 052710 000010
21 104524 012600
22 104526
   104526 000002

;UDASRV
;
;UDA INTERRUPT SERVICE ROUTINE. MARKS UDA CONTROLLER TABLE THAT AN
;INTERRUPT HAS BEEN RECEIVED.
;
;THIS ROUTINE IS CALLED BY A [JSR R0,UDASRV] INSTRUCTION FROM WITHIN
;THE CONTROLLER TABLE. THE PC STORED IN R0 IS THE ADDRESS OF THE C.FLG
;WORD IN THE CONTROLLER TABLE. THE STACK CONTAINS THE SAVED CONTENTS
;OF R0 FOLLOWED BY THE INTERRUPTED PC AND PS.
;
;INPUTS:
;   R0 - ADDRESS OF C.FLG WORD IN CONTROLLER TABLE
;   STACK - SAVED CONTENTS OF R0
;OUTPUTS:
;   CT.CMD CLEARED AND CT.MSG SET IN C.FLG WORD OF CONTROLLER TABLE
;   R0 - RESTORED FROM STACK

UDASRV::
    BIS     @CT.MSG,(R0)           ;SET CT.MSG
    MOV     (SP)+,R0              ;;POP STACK INTO R0
L10037:
    RTI

```

```

1
2
3
4
5
6
10
11
12
13
14
15
16
17
28 104530
    104530 010246
    104532 010346
30 104534 005002
31 104536 010703 064624
46 104542 006200
47 104544 103001
48 104546 060302
49 104550 006303
50 104552 005700
51 104554 001372
53
54
55 104556 013700 064626
56 104562 013703 064630
57 104566 020037 064626
58 104572 001371
59
60
61
62 104574 060200
63 104576 005503
67
68
69
70 104600 010021
71 104602 010311
72
76 104604 012603
    104606 012602
78 104610 000207

;SETT0
;
;SET TIMEOUT COUNTER TO SOME NUMBER OF SECONDS FROM CURRENT TIME.
;
;INPUTS:
;   R0 - NUMBER OF SECONDS FOR TIMEOUT
;   R1 - ADDRESS WHERE TWO WORD TIME TO BE PUT
;OUTPUTS:
;   R0 - CONTENTS DESTROYED
;   R1 - INCREMENTED BY 2
;
;COMPUTE CLOCK TICKS TIL TIMEOUT

SETT0:
MOV     R2,-(SP)           ;;PUSH R2 ON STACK
MOV     R3,-(SP)           ;;PUSH R3 ON STACK
CLR     R2                 ;;CLEAR PRODUCT
MOV     KW,HZ,R3           ;;GET MULTIPLICAND
SET00: ASR     R0           ;;SHIFT MULTIPLIER TO RIGHT
        BCC    SET01       ;;IF A ONE BIT SHIFTED OUT
        ADD    R3,R2       ;;ADD MULTIPLICAND TO PRODUCT
SET01:  ASL     R3           ;;DOUBLE THE MULTIPLICAND
        TST    R0
        BNE    SET00       ;;CONTINUE UNTIL MULTIPLIER IS ZERO
        ;GET CURRENT TIME

SET02:  MOV     KW,EL,R0    ;;GET TIME
        MOV     KW,EL*2,R3
        CMP     K0,KW,EL   ;;IF CHANGED DURING RETRIEVAL
        BNE    SET02       ;;GET IT AGAIN

;ADD TIME TIL TIMEOUT

ADD     R2,R0              ;;ADD
ADC     R3
;PUT RESULT IN STORAGE

MOV     R0,(R1)+
MOV     R3,(R1)

MOV     (SP)+,R3           ;;POP STACK INTO R3
MOV     (SP)+,R2           ;;POP STACK INTO R2
RETURN
    
```

```

1
2      ;UDAIN
3
4      ;FUNCTIONAL DESCRIPTION:
5      ;   SUBROUTINE TO INITIALIZE A UDA AND BRING IT ON-LINE.
6      ;   ALL STEPS ARE CHECKED. AN ERROR MESSAGE IS REPORTED IF ANY ERROR
7      ;   DETECTED.
8
9      ;INPUTS:
10     ;   R5 - ADDRESS OF CONTROLLER TABLE.
11     ;   R4 - LENGTH, INTERRUPT AND VECTOR FIELDS TO SEND TO UDA
12
13     ;IMPLICIT INPUTS:
14     ;   FFREE - FIRST FREE ADDRESS OF MEMORY. THIS ADDRESS IS GIVEN TO UDA
15     ;           AS START OF RING BUFFER.
16     ;   FSIZE - SIZE OF FREE MEMORY AVAILABLE IN WORDS.
17
18     ;OUTPUTS:
19     ;   R1 - SIZE OF RING BUFFER IN WORDS IF NO ERROR,
20     ;   R4 - ADDRESS OF UDAIP REGISTER IN UDA,
21     ;   R5 - UNCHANGED,
22
23     ;   Z CLR  IF NO ERROR,
24     ;   Z SET  IF ANY ERROR REPORTED
25
26     ;CHECK IF ENOUGH FREE MEMORY FOR RING BUFFER
27
28     ;UDAIN:
29     MOV     R3, -(SP)           ; PUSH R3 ON STACK
30     MOV     R4, R0             ; GET MESSAGE LENGTH
31     SWAB   R0
32     BIC    @177770, R0
33     JSR    PC, CLOG           ; COMPUTE LOGARITHMIC VALUE
34     MOV     R1, R2             ; SAVE RESULT IN R2
35     MOV     R4, R0             ; GET COMMAND LENGTH
36     SWAB   R0
37     ROR    R0
38     ROR    R0
39     ROR    R0
40     BIC    @177770, R0
41     JSR    PC, CLOG           ; COMPUTE LOGARITHMIC VALUE
42     ADD     R2, R1             ; ADD THE TWO RESULTS
43     ASL    R1                  ; MULTIPLY BY 2 WORDS PER RING
44     ADD     @<HC.ISZ>/2, R1    ; ADD SPACE FOR INTERRUPT INDICATORS
45     LMP    R1, FSIZE          ; COMPARE WITH SIZE OF FREE MEMORY
46     BLOS   1$                 ; FATAL ERROR IF NOT ENOUGH MEMORY
47     JMP    FMERR
48
49     ;FILL HOST COMMUNICATION AREA WITH ALL ONES
50
51     1$: MOV     FFREE, R2       ; GET FIRST ADDRESS OF RING BUFFER
52     MOV     R1, R3            ; GET SIZE OF RING BUFFER
53     2$: MOV     @-1, (R2)+     ; WRITE ONES TO BUFFER
54     DEC    R3                 ; COUNT THE WORDS IN BUFFER
55     BGT    2$                 ; LOOP UNTIL ENTIRE BUFFER WRITTEN
56
57     ;DO THE INITIALIZATION
58
59     JSR    PC, UDAIS1         ; DO FIRST THREE STEPS

```

```

26 104612
27 104612 010346
28 104614 010400
29 104616 000300
29 104620 042700 177770
30 104624 004737 105550
31 104630 010102
32 104632 010400
33 104634 000300
34 104636 006000
35 104640 006000
36 104642 006000
37 104644 042700 177770
38 104650 004737 105550
39 104654 060201
40 104656 006301
41 104660 062701 000002
42 104664 020137 064414
43 104670 101402
44 104672 000137 075410
45
46
47
48 104676 013702 064412
49 104702 010103
50 104704 012722 177777
51 104710 005303
52 104712 003374
53
54
55
56 104714 004737 105070

```

57	104720	103460			BCS	9#		;GET OUT IF UDA MICROCODE REPORTED FAILURE
58	104722	012364	000002		MOV	(R3)+,2(R4)		;WRITE NEXT WORD TO UDASA REGISTER
59	104724	012700	000310		MOV	#200,R0		;GET TRY COUNTER
60	104732	016402	000002	3#:	MOV	2(R4),R2		;LOOK AT UDASA
61	104734	001410			BEQ	5#		
62	104740	100005			SPL	4#		
63	104742	104455			TRAP	C#ERDF		
	104744	000030			.WORD	24		
	104746	000000			.WORD	0		
	104750	074602			.WORD	ERR024		
64	104752	000443			BR	9#		
65								
66	104754	005300		4#:	DEC	R0		
67	104756	001365			BNE	3#		
68	104760	010264	000002	5#:	MOV	R2,2(R4)		;WRITE 0 TO UDASA (PURGE)
69	104764	011402			MOV	(R4),R2		;READ FROM UDAIP (POLL)
70	104768	004737	105410		JSR	PC,UDARSP		;WAIT FOR STEP OR ERROR BIT
71	104772	103433			BCS	9#		;GET OUT IF UDA MICROCODE REPORTED FAILURE
72	104774	010146			MOV	R1,-(SP)		;PUSH R1 ON STACK
73	104776	004733			JSR	PC,8(R3)+		;CALL LAST ROUTINE
74	105000	012601			MOV	(SP)+,R1		;POP STACK INTO R1
75								
76								
77								;CHECK HOST COMMUNICATION AREA FOR ALL ZEROS
78	105002	013702	064412		MOV	FFREE,R2		;GET FIRST ADDRESS OF RING BUFFER
79	105006	010103			MOV	R1,R3		;GET SIZE OF RING BUFFER
80	105010	005722		6#:	TST	(R2)+		;CHECK WORD IN BUFFER
81	105012	001003			BNE	7#		;GO TO ERROR REPORTER IF NOT ZERO
82	105014	005303			DEC	R3		;COUNT THE WORDS IN BUFFER
83	105016	003374			BGT	6#		;LOOP UNTIL ALL WORDS CHECKED
84	105020	000405			BR	8#		
85								
86	105022			7#:	TRAP	C#ERDF		
	105022	104455			.WORD	23		
	105024	000027			.WORD	0		
	105026	000000			.WORD	0		
	105030	074516			.WORD	ERR023		
87	105032	000413			BR	9#		
88								
89								
90								;SEND GO BIT TO UDASA REGISTER TO END INITIALIZATION
91								
100	105034	016500	000006	8#:	MOV	C.BST(R5),R0		;GET BURST VALUE
102	105040	006300			ASL	R0		;SHIFT TO POSITION
103	105042	006300			ASL	R0		
104	105044	052700	000001		BIC	#5A,GO,R0		;SET THE GO BIT
105	105050	010064	000002		MOV	R0,2(R4)		;SEND TO UDA
106	105054	012603			MOV	(SP)+,R3		;POP STACK INTO R3
107	105056	000244			CLZ			;CLEAR Z AS NO ERROR INDICATION
108	105060	000207			RTS	PC		
109								
110								;ERROR RETURN
111								
112	105062			9#:	MOV	(SP)+,R3		;POP STACK INTO R3
	105062	012603			SEZ			;SET Z TO INDICATE ERROR OCCURRED
113	105064	000264			RTS	PC		
114	105066	000207						



```

49 105244 103412          BCS      4$          ;GET OUT IF ERROR
50 105246 006337 105546  ASL      UDARSD        ;SHIFT TO NEXT STEP BIT
51 105252 032737 040000 105546  BIT      @SA.S4,UDARSD ;CHECK IF NOW AT STEP 4
52 105260 001003          BNE      3$          ;GET OUT IF SO
53 105262 012364 000002  MOV      (R3)+,2(R4)   ;WRITE DATA TO UDASA REGISTER
54 105266 000762          BR       2$          ;STAY IN LOOP
55
56 105270 000241          3$:    CLC          ;CLEAR CARRY FOR NO ERROR INDICATION
57 105272          4$:    MOV      (SP)+,R1      ;POP STACK INTO R1
105272 012601          RTS      PC
105274 000207
58 105274 000207          ;DATA TO BE SENT AND RECEIVED BY UDA INITIALIZATION
59
60
61
62 105276 105314  INITBL: .WORD  RSP.S1      ;1ST WORD RESPONSE CHECK ROUTINE
63 105300 000000  SND.S1: .WORD  0          ;1ST WORD TO SEND TO UDASA
64 105302 105322          .WORD  RSP.S2      ;2ND WORD RESPONSE CHECK ROUTINE
65 105304 000000  SND.S2: .WORD  0          ;2ND WORD TO SEND TO UDASA
66 105306 105342          .WORD  RSP.S3      ;3RD WORD RESPONSE CHECK ROUTINE
67 105310 000000  SND.S3: .WORD  0          ;3RD WORD TO SEND TO UDASA
68 105312 105360          .WORD  RSP.S4      ;4TH WORD RESPONSE CHECK ROUTINE
69
70
71          ;RESPONSE CHECK FOR FIRST WORD (STEP 1) FROM UDASA
72          ;CHECK FOR PROPER CONTROLLER TYPE
73 105314 012701 004400  RSP.S1: MOV      @SA.S1+SA.DI,R1 ;SET STEP ONE BIT
74 105320 000422          BR       RSP.CK        ;NOW DO A RESPONSE CHECK
75
76          ;RESPONSE CHECK FOR SECOND WORD (STEP 2) FROM UDASA
77          ;CHECK FOR ECHO OF INTI AND VECTOR
78
79 105322 013701 105300  RSP.S2: MOV      SND.S1,R1      ;GET WORD SENT TO UDASA
80 105326 000301          SWAB     R1          ;GET HIGH 8 BITS
81 105330 042701 177400          BIC     @177400,R1
82 105334 052701 010000          BIS     @SA.S2,R1      ;SET STEP 2 BIT
83 105340 000412          BR       RSP.CK        ;NOW DO A RESPONSE CHECK
84
85          ;RESPONSE CHECK FOR THIRD WORD (STEP 3) FROM UDASA
86          ;CHECK FOR ECHO OF MESSAGE AND COMMAND RING LENGTHS
87
88 105342 013701 105300  RSP.S3: MOV      SND.S1,R1      ;GET WORD SENT TO UDASA
89 105346 042701 177400          BIC     @177400,R1      ;JUST LOW 8 BITS
90 105352 052701 020000          BIS     @SA.S3,R1      ;SET STEP 3 BIT
91 105356 000403          BR       RSP.CK        ;NOW DO A RESPONSE CHECK
92
93          ;RESPONSE CHECK FOR FOURTH WORD (STEP 4) FROM UDASA
94          ;CHECK FOR ECHO OF PURGE AND LFAIL BITS
95
96 105360 010201          RSP.S4: MOV      R2,R1          ;GET RESPONSE FROM UDA AND
97 105362 010237 105406  MOV      R2,SSTEP4      ;SAVE STEP 4 VALUE.
98
99          ;RESPONSE CHECK, COMPARE EXPECTED DATA IN R1 WITH ACTUAL DATA IN R2
100
101 105366 020102          RSP.CK: CMP      R1,R2          ;COMPARE THE DATA
102 105370 001405          BEQ     1$          ;EXIT IF COMPARED CORRECTLY
103          ;ERROR, UDA DID NOT RETURN CORRECT DATA IN
104          ;UDASA REGISTER DURING INITIALIZATION

```

G2

105	105372	104455	TRAP	C\$ERDF	
	105374	000031	.WORD	25	
	105376	000000	.WORD	0	
	105400	074616	.WORD	ERR025	
106	105402	000261	SEC		
107	105404	000207	1\$: RTS	PC	
108					
109	105406	000000	SSTEP4: .WORD	0	;SAVE STEP 4 VALUE HERE



51	105542	000241			
52	105544	000207	5:	CLC	
53				RTS	PC
54					
55					
56	105546	000000	UDARSD:	.WORD	0

;CLEAR CARRY AS NO ERROR INDICATION  
;LOCATION FOR STEP BIT MASK  
;LOAD BY CALLING ROUTINE

```

1
2
3      ;CLOG
4      ;
5      ; COMPUTE LOGARITHMIC VALUE OF NUMBER TO BASE 2.
6      ;
7      ; INPUTS:
8      ;      R0 - LOGARITHM TO BE CONVERTED
9      ; OUTPUTS:
10     ;      R1 - VALUE OF 2 RAISED TO POWER OF INPUT NUMBER
11     105550
12     105550 010046
13     105552 005001
14     105554 000261
15     105556 006101
16     105560 005300
17     105564 012600
18     105566 000207

CLOG:
      MOV     R0, -(SP)      ; PUSH R0 ON STACK
      CLR     R1             ; SET UP ZERO START VALUE
      SEC     ; WITH CARRY READY TO SHIFT IN
1$:   ROL     R1             ; SHIFT TO LEFT
      DEC     R0             ; UNTIL R0
      BPL     1$            ; GOES NEGATIVE
      MOV     (SP)+, R0     ; POP STACK INTO R0
      RTS     PC
    
```

PRE-PROGRAMMED SUBROUTINES

```

1
2
3
4
5
6
7
8
9
10
11
16 105570 012701 000006
17 105574 004737 105630
18 105600 006101
19 105602 004737 105612
20 105606 006001
21 105610 000207

;RDDLL
;
;READ DISK DRIVE DOWNLINE LOAD PROGRAM INTO MEMORY
;
;INPUTS:
;   DLLNAM - NAME OF PROGRAM IN RAD50 (TWO WORDS)
;OUTPUTS:
;   FREE MEMORY CONTAINING PROGRAM
;   CARRY CLEAR IF NO ERROR, CARRY SET IF PROGRAM NOT FOUND

RDDLL:  MOV     06.,R1           ;TYPE OF PROGRAM IN DATA FILE
        CALL   RDREC          ;READ PROGRAM INTO MEMORY
        ROL    R1             ;PRESERVE CARRY STATE IN R1
        CALL   CLOSEF        ; WHILE CLOSING THE DATA FILE
        ROR    R1             ; AS NORMAL POSITION IS LOST
        RETURN

```

L2

```
1
2
3
4
5
6
7
8
9
10
12 105612 005737 064474
13 105615 001403
14 105620 104435
15 105622 005037 064474
16 105626 000207

;CLOSEF
;
;CLOSE DATA FILE FOR DM PROGRAMS
;
;INPUTS:
;   FILOPN - ZERO IF FILE NOT OPEN
;OUTPUTS:
;   NONE

CLOSEF: TST     FILOPN           ;SEE IF FILE CURRENTLY OPEN
        BEQ     1$
        TRAP    C$CLOS
        CLR     FILOPN
1$:     RETURN

;AND MARK AS SO
```





102							
103	106150			FWORD:	TRAP	C#GETB	;READ A BYTE FROM FILE
104	106150	104426			ADD	R0,R4	;UPDATE CHECKSUM ERROR
105	106152	060004			MOV8	R0,FDATA	;START TO BUILD WORD
106	106154	110037	064472				;READ A BYTE FROM FILE
107							
108	106160	104426			TRAP	C#GETB	
109	106162	060004			ADD	R0,R4	;UPDATE CHECKSUM
110	106164	110037	064473		MOV8	R0,FDATA+1	;COMPLETE WORD
111	106170	000207			RETURN		
112							
113	106172	004737	105612	RDERR:	CALL	CLOSEF	;CLOSE FILE AS POSITION IS LOST
114	106176	012605			MOV	(SP),R5	;POP STACK INTO R5
	106200	012604			MOV	(SP),R4	;POP STACK INTO R4
	106202	012603			MOV	(SP),R3	;POP STACK INTO R3
	106204	012602			MOV	(SP),R2	;POP STACK INTO R2
	106206	012601			MOV	(SP),R1	;POP STACK INTO R1
	106210	012600			MOV	(SP),R0	;POP STACK INTO R0
115	106212	000261			SEC		;ERROR RETURN, FILE NOT FOUND
116	106214	000207			RETURN		
117							
118	106216			RWRDE1:	TRAP	C#ERSF	
	106216	104454			.WORD	5	
	106220	000005			.WORD	0	
	106222	000000			.WORD	ERR005	
	106224	074336					
119							;DO CLEAN-UP TRAP
120	106226	104444			TRAP	C#DCLN	

```

1
2
3
4
5
6 106230
7 106230 062737 000001 064626
8 106236 005537 064630
9 106242 012777 000105 156346
10 106250
    106250 000002
11
12 106252
13 106252 005237 064454
14 106256
    106256 000002

```

```

;KW11I
;CLOCK INTERRUPT SERVICE ROUTINE
KW11I::
    ADD     #1,KW.EL           ;COUNT THE INTERRUPT
    ADC    KW.EL,#2
    MOV     #KW.OUT,#KW.CSR   ;RESTART THE CLOCK
L10040:
    RTI
INTSRV::
    INC     INTRCV           ; FLAG INTERRUPT AS RECEIVED
L10041:
    RTI

```

```

1
2
3
4
5
6
7
8
9
10
11
12 106260 005037 064636
13 106264 010346
14 106266 010446
15 106270 012746 000340
16 106274 012746 104510
17 106300 012746 000004
18 106304 012746 000003
19 106310 104437
20 106312 062706 000010
21 106316 012703 000010
22 106322 012704 064534
23 106326 005714
24 106330 001403
25 106332 005034
26 106334 005303
27 106336 001373
28 106340 005737 064616
29 106344 001403
30 106346 012777 000105 156242
31 106354
32 106354 012604
33 106356 012603
34 106360 000207

;RESET
;
; RESET ALL UDA-50S IN THE CONTROLLER TABLES
;
; INPUTS:
; IPADRS - CONTAINS ALL IP ADDRESSES
; OUTPUTS:
; NONE
;

RESET: CLR NXMAD ;CLEAR NON-EXISTANT MEMORY ADDRESS
MOV R3,-(SP) ;PUSH R3 ON STACK
MOV R4,-(SP) ;PUSH R4 ON STACK
;SETUP TIMEOUT ERROR VECTOR
MOV @PRI07,-(SP)
MOV @NXMI,-(SP)
MOV @ERRVEC,-(SP)
MOV @3,-(SP)
TRAP C#SVEC
ADD @10,SP
MOV @8,R3 ;R3 = COUNTER OF ENTRIES
MOV @IPADRS,R4 ;R4 -> IP ADDRESS
1$: TST (R4) ;IS THERE AN ENTRY?
BEQ 2$ ;IF NOT, DONE
CLR @R4+ ;INIT UDA
DEC R3 ;MAKE SURE WE DO NOT EXTEND OVER AREA
BNE 1$ ;IF NOT DONE, BRANCH
2$: TST KW.CSR ;SEE IF CLOCK PRESENT,
BEQ 3$ ;BRANCH IF NOT, ELSE
MOV @KW.OUT,@KW.CSR ;START THE CLOCK.
3$: MOV (SP)+,R4 ;POP STACK INTO R4
MOV (SP)+,R3 ;POP STACK INTO R3
RETURN

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 106362 005737 064616
15 106366 001465
16 106370 010046
   106372 010346
   106374 010446
   106376 010546
17 106400 013703 064626
18 106404 013704 064630
19 106410 013700 064624
20 106414 004737 102704
21 106420 012700 000074
22 106424 004737 102704
23 106430 010546
24 106432 004737 102704
25 106436 010346
   106440 004137 075722
   106444 065045
   106446 000002
26 106450 020527 000011
27 106454 003004
28 106456 112700 000060
   106462 004737 075506
29 106466
   106466 010546
   106470 004137 075722
   106474 065070
   106476 000002
30 106500 012605
31 106502 020527 000011
32 106506 003004
33 106510 112700 000060
   106514 004737 075506
34 106520
   106520 010546
   106522 004137 075722
   106526 065076
   106530 000002
35 106532 012605
   106534 012604
   106536 012603
   106540 012600
36 106542
   106542 112700 000040
   106546 004737 075506

```

```

;RNTIME
;PRINT RUNTIME
;INPUTS:
;   KW.EL - CONTAINS ELAPSED TIME
;   KW.HZ - HERTZ OF CLOCK
;OUTPUTS:
;   IF CLOCK ON SYSTEM:
;   " RNTIME HH:MM:SS " PRINTED
;   IF NO CLOCK: ONE SPACE IS PRINTED
RNTIME: TST     KW.CSR           ;CHECK IF A CLOCK PRESENT
        BEQ     RNTIMX         ;BRANCH IF NOT
        MOV     R0,-(SP)       ;PUSH R0 ON STACK
        MOV     R3,-(SP)       ;PUSH R3 ON STACK
        MOV     R4,-(SP)       ;PUSH R4 ON STACK
        MOV     R5,-(SP)       ;PUSH R5 ON STACK
        MOV     KW.EL,R3       ;GET ELAPSED TIME
        MOV     KW.EL+2,R4
        MOV     KW.HZ,R0
        CALL    DIVIDE         ;GET SPEED OF CLOCK
        MOV     #60,R0         ;COMPUTE SECONDS OF ELAPSED TIME
        CALL    DIVIDE         ;NOW DIVIDE BY 60
        MOV     R5,-(SP)       ;TO COMPUTE MINUTES
        CALL    DIVIDE         ;PUSH R5 ON STACK
        MOV     R3,-(SP)       ;DIVIDE BY 60 AGAIN
        JSR     R1,LPNT        ;PUSH R3 ON STACK
        .WORD   RNTIM          ;CALL LPNT PRINT ROUTINE
        .WORD   ARG.CT         ;ADDRESS OF ASCIZ STRING
        CMP     R5,#9          ;ARGUMENT COUNT * 2
        BGT 1$                ;IF MINUTES 9 OR LESS
        MOV     #'0,R0         ;STORE #'0 IN R0 AND
        JSR     PC,PRINTC      ;PRINT THE CHARACTER.
1$:     MOV     R5,-(SP)       ;PUSH R5 ON STACK
        JSR     R1,LPNT        ;CALL LPNT PRINT ROUTINE
        .WORD   RNTIM1         ;ADDRESS OF ASCIZ STRING
        .WORD   ARG.CT         ;ARGUMENT COUNT * 2
        MOV     (SP)+,R5       ;POP STACK INTO R5
        CMP     R5,#9          ;IF 9 OR LESS
        BGT 2$                ;STORE #'0 IN R0 AND
        MOV     #'0,R0         ;PRINT THE CHARACTER.
2$:     MOV     R5,-(SP)       ;PUSH R5 ON STACK
        JSR     R1,LPNT        ;CALL LPNT PRINT ROUTINE
        .WORD   RNTIM2         ;ADDRESS OF ASCIZ STRING
        .WORD   ARG.CT         ;ARGUMENT COUNT * 2
        MOV     (SP)+,R5       ;POP STACK INTO R5
        MOV     (SP)+,R4       ;POP STACK INTO R4
        MOV     (SP)+,R3       ;POP STACK INTO R3
        MOV     (SP)+,R0       ;POP STACK INTO R0
RNTIMX: MOV     #'0,R0         ;STORE #'0 IN R0 AND
        JSR     PC,PRINTC      ;PRINT THE CHARACTER.

```

F3

CZUDCEO UDA & DISK DRV DIAG MACRO V05.00 Wednesday 04-Jan-84 16:12 Page 183-1  
PRE-PROGRAMMED SUBROUTINES

SFQ 0233

37 106552 000207

RETURN

```

1
2
3           ;WCHNG
4           ;
5           ;      WAIT UNTIL UDASA CHANGES FROM WHAT IS IN WCHNGD
6 106554 012700 000012      WCHNG:  MOV     #10.,R0           ;SET TIMEOUT FOR 10 SECONDS
7 106560 010501             MOV     R5,R1           ;POINT TO CONTROLLER TABLE
8 106562 062701 000040             ADD     #C.TO,R1
9 106566 004737 104530             CALL    SETTO
10 106572 026437 000002 106650 1$:  CMP     2(R4),WCHNGD      ;SEE IF CHANGED
11 106600 001022             BNE     2$
12
13 106602 104422             TRAP    C$BRK           ;>>>>>>>>>BREAK BACK TO MONITOR<<<<<<<<<<
14 106604 005737 064616             TST     KW.CSR           ;SEF IF CLOCK ON SYSTEM
15 106610 001770             BEQ     1$
16 106612 023765 064630 000042      CMP     KW.EL+2,C.TO(R5)  ;CHECK IF TIME OUT OCCURRED
17 106620 101005             BHI    3$
18 106622 001363             BNE     1$
19 106624 023765 064626 000040      CMP     KW.EL,C.TO(R5)
20 106632 103757             BLO    1$
21 106634             3$:
22 106634 104455             TRAP    C$ERDF
23 106636 000033             .WORD  27
24 106640 000000             .WORD  0
25 106642 074654             .WORD  ERR027
26 106644 000264             SEZ
27 106646 000207             2$:  RETURN           ; FLAG AS ERROR
28                                     ; RETURN TO CALLING PROGRAM
29
30 106650      WCHNGD: .BLKW 1           ; OLD PORT CONTENTS
31 106652      BRLEV:  .BLKW 1           ; WORD FOR BRANCH LEVEL STORAGE
32
33
34
35
36
37
38
39
40
41
42
43

```

```

12          .SBTTL REPORT CODING SECTION
40
42
43          ; THE REPORT CODING SECTION CONTAINS THE
44          ; "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
45          ;--
46
47 106654   L$RPT::
48
49 106654   010046   MOV      R0,-(SP)          ;;PUSH R0 ON STACK
106656   010146   MOV      R1,-(SP)          ;;PUSH R1 ON STACK
106660   010246   MOV      R2,-(SP)          ;;PUSH R2 ON STACK
106662   010346   MOV      R3,-(SP)          ;;PUSH R3 ON STACK
106664   010446   MOV      R4,-(SP)          ;;PUSH R4 ON STACK
106666   010546   MOV      R5,-(SP)          ;;PUSH R5 ON STACK
50 106670   013746   064444   MOV      TNUM,-(SP)       ;PUSH TNUM ON STACK
106674   004137   075714   JSR      R1,LPNTS        ;CALL LPNTS PRINT ROUTINE
106700   107262   .WORD   RPTMSG           ;ADDRESS OF ASCIZ STRING
106702   000002   .WORD   ARG,CT           ;ARGUMENT COUNT * 2
51 106704   004737   106362   CALL    RNTIME           ;GET RUNTIME PARAMETERS
52 106710   112700   000015   MOV     #CR,R0           ;STORE #CR IN R0 AND
106714   004737   075506   JSR     PC,PRINTC        ;PRINT THE CHARACTER.
53 106720   012701   064632   MOV     #TIME,R1         ;AT 15 MINUTES FROM NOW
57 106724   012700   001604   MOV     #15,*60.,R0      ;SET TIME FOR NEXT REPORT
58 106730   004737   104530   CALL    SETTO            ;
66 106734   022737   000004   064444   CMP     #4,TNUM          ;IF NOT TEST 4
68 106742   001402   BEQ     1$              ;BRANCH IF SO, ELSE
69 106744   000137   107242   JMP     RPTXX            ;EXIT REPORT SECTION.
70
71 106750   1$:
106750   004137   075714   JSR     R1,LPNTS        ;CALL LPNTS PRINT ROUTINE
106754   107316   .WORD   RPTMSG           ;ADDRESS OF ASCIZ STRING
106756   000000   .WORD   ARG,CT           ;ARGUMENT COUNT * 2
72 106760   013705   064424   MOV     CTABS,R5         ;GET ADDRESS OF 1ST CONTROLLER TABLE
73
74 106764   005765   000002   RPTCT: YST     C,UNIT(R5)      ;SEE IF CONTROLLER AVAILABLE FOR TESTING
76 106770   109520   BMI     RPTCTN           ;
83 106772   010504   MOV     R5,R4            ;COMPUTE ADDRESS OF DRIVE TABLE POINTERS
84 106774   062704   000020   ADD     #C.DRO,R4
85 107000   012703   000010   MOV     #8.,R3           ;GET COUNT OF DRIVES
86 107004   012401   RPTDT: MOV     (R4)+,R1      ;LOOK AT POINTER
87 107006   001511   BEQ     RPTCTN           ;GO TO NEXT IF NO TABLE
88 107010   005761   000002   TST     D,UNIT(R1)       ;SEE IF DRIVE AVAILABLE
90 107014   100504   BMI     RPTDTN           ;
98 107016   010346   MOV     R3,-(SP)         ;;PUSH R3 ON STACK
107020   010446   MOV     R4,-(SP)         ;;PUSH R4 ON STACK
107022   010546   MOV     R5,-(SP)         ;;PUSH R5 ON STACK
107024   010146   MOV     R1,-(SP)         ;;PUSH R1 ON STACK
99 107026   012700   064476   MOV     #TEMP,R0         ;PLACE 18 SPACE CHARACTERS INTO
100 107032   012701   000022   MOV     #18.,R1          ;TEMP STORAGE
101 107036   112720   000040   1$:  MOV     #',(R0)+
102 107042   005301   DEC     R1
103 107044   001374   BNE     1$
104 107046   005010   CLR     (R0)             ;THEN A NULL CHARACTER
105 107050   011605   MOV     (SP),R5          ;GET DRIVE TABLE STORAGE ADDRESS
106 107052   016501   000200   MOV     D,SERN(R5),R1    ;GET SERIAL NUMBER
107 107056   016502   000202   MOV     D,SERN+2(R5),R2

```

```

108 107062 016503 000204      MOV      D.SERN+4(R5),R3
109 107066 005004             CLR      R4
110 107070 004737 102742      2$:     CALL   DIV10             ;DIVIDE BY 10
111 107074 062705 000060      ADD      #0,R5             ;CONVERT TO ASCII CHARACTER
112 107100 110540             MOV      R5,-(R0)         ;PUT DIGIT INTO TEMP STORAGE
113 107102 010146             MOV      R1,-(SP)
114 107104 050216             BIS      R2,(SP)         ;SEE IF QUOTIENT IS ZERO
115 107106 050316             BIS      R3,(SP)
116 107110 050426             BIS      R4,(SP)
117 107112 001366             BNE     2$               ;IF NOT, DIVIDE AGAIN
118 107114 012601             MOV      (SP)+,R1         ;POP STACK INTO R1
119 107116 016146 000164      MOV      D.XFRW(R1),-(SP)
107122 016146 000166      MOV      D.XFRR(R1),-(SP)
107126 016146 000174      MOV      D.SEEK(R1),-(SP)
107132 012746 064476      MOV      @TEMP,-(SP)
107136 011146             MOV      (R1),-(SP)
107140 016146 000002      MOV      D.UNIT(R1),-(SP)
107144 012746 107534      MOV      @RPTMSD,-(SP)
107150 012746 000007      MOV      @7,-(SP)
107154 010600             MOV      SP,R0
107156 104416             TRAP    C:PNTS
107160 062706 000020      ADD      #20,SP
121 107164 016146 000176      MOV      D.ECCC(R1),-(SP)
107170 016146 000172      MOV      D.SERR(R1),-(SP)
107174 016146 000170      MOV      D.HERR(R1),-(SP)
107200 012746 107603      MOV      @RPTMD2,-(SP)
107204 012746 000004      MOV      @4,-(SP)
107210 010600             MOV      SP,R0
107212 104416             TRAP    C:PNTS
107214 062706 000012      ADD      #12,SP
145 107220 012605             MOV      (SP)+,R5         ;POP STACK INTO R5
107222 012604             MOV      (SP)+,R4         ;POP STACK INTO R4
107224 012603             MOV      (SP)+,R3         ;POP STACK INTO R3
146 107226 005303      RPTDTN: DEC      R3         ;COUNT THE DRIVE TABLES
147 107230 003265      BGT     RPTD:           ;REPEAT FOR ALL DRIVE TABLES
148 107232 062705 000046      RPTCTN: ADD     @C.SIZE,R5 ;GO TO NEXT CONTROLLER TABLE
149 107236 005715      TST     (R5)
154 107240 001251      BNE     RPTCT
156 107242      RPTXX:
107242 012605             MOV      (SP)+,R5         ;POP STACK INTO R5
107244 012604             MOV      (SP)+,R4         ;POP STACK INTO R4
107246 012603             MOV      (SP)+,R3         ;POP STACK INTO R3
107250 012602             MOV      (SP)+,R2         ;POP STACK INTO R2
107252 012601             MOV      (SP)+,R1         ;POP STACK INTO R1
107254 012600             MOV      (SP)+,R0         ;POP STACK INTO R0
168
169 107256 000167      .WORD   J$JMP
107260 000344      .WORD   L10042-2-.
170
174 107262      116      042      124  RPTMSG: .ASCIZ  \N"TEST "D3" IN PROGRESS. "\
175 107316      116      042      125  RPTMSH: .ASCII  \N"UNIT DRIVE SERIAL NUMBER SEEKS MBYTES MBYTES HARD SOFT ECC"N\
176 107430      042      040      040      .ASCIZ  \
177 107534      045      123      062  RPTMSD: .ASCIZ  \#S2#D2#S3#D3#S1#T#S1#D5#S2#D5#S3#D5#S2\
178 107603      045      104      065  RPTMD2: .ASCIZ  \#D5#S2#D5#S1#D5#N\
198
199
200 107626      L10042:
    
```

J3

CZUDCEO UDA & DISK DRV DIAG MACRO V05.00 Wednesday 04-Jan-84 15:12 Page 186-2  
REPORT CODING SECTION

SFQ 0242

107626 104425

TRAP C\$RPT

K3

```
1  
2          .SBTTL  PROTECTION TABLE  
3  
4          ;++  
5          ; THIS TABLE IS USED BY THE RUNTIME SERVICES  
6          ; TO PROTECT THE LOAD MEDIA.  
7          ;--  
8  
9 107630    L$PROT::  
10  
11 107630    177777          -1          ;OFFSET INTO P-TABLE FOR CSR ADDRESS  
12 107632    177777          -1          ;OFFSET INTO P-TABLE FOR MASSBUS ADDRESS  
13 107634    177777          -1          ;OFFSET INTO P-TABLE FOR DRIVE NUMBER  
14
```

INITIALIZE SECTION

```

1      .SBTTL  INITIALIZE SECTION
2
3
4      ;**
5      ; THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
6      ; AT THE BEGINNING OF EACH PASS.
7      ;--
8
9      ;*****
10     ; IF HERE FROM START COMMAND
11     ;     THEN
12     ;         SET ISTRT BIT & CLEAR OTHER BITS IN FLAG
13     ;     ENDF
14     ; IF HERE FROM RESTART COMMAND
15     ;     THEN
16     ;         SET IREST BIT IN IFLAGS
17     ;     ENDF
18     ; IF HERE FROM START OR RESTART COMMAND
19     ;     THEN
20     ;         RESET ALL UNITS
21     ;         ESTABLISH FREE MEMORY
22     ;         CLEAR TNUM
23     ;         INITIALIZE CLOCK
24     ;         BUILD CONTROLLER & DRIVES TABLES IN MEMORY
25     ;         EXIT INIT SECTION
26     ;     ENDF
27     ; IF HERE FROM CONTINUE COMMAND
28     ;     THEN
29     ;         SET ICONT BIT IN IFLAGS
30     ;         EXIT INIT SECTION
31     ;     ENDF
32     ; IF HERE FROM POWER FAIL RESTART
33     ;     THEN
34     ;         EXIT INIT SECTION
35     ;     ENDF
36     ; IF HERE FROM NEW PASS OR SUB-PASS
37     ;     THEN
38     ;         LOOK FOR ANY ADDED OR DROPPED UNITS
39     ;         EXIT INIT SECTION
40     ;     ENDF
41     ;*****

```

```

INITIALIZE SECTION
1 107636
2
3 107636 012700 000040          L$INIT::
  107642 104447          MOV    4EF.STA,RO
  TRAP  C$REFG          ;HERE FROM START COMMAND?
4
5 107644 103004          BCC   1$              ;BRANCH TO 1$ IF NOT, ELSE
6 107646 012737 000010 064440  MOV   4ISTRH,IFLAGS  ;SET START BIT IN FLAG.
7 107654 000531          BR    1$              INIT1
8 107656
9 107656 012700 000037          1$:
  107662 104447          MOV   4EF.RES,RO
  TRAP  C$REFG          ;HERE FROM RESTART COMMAND?
10
11 107664 103004          BCC   2$              ;BRANCH TO 2$ IF NOT, ELSE
12 107666 052737 000004 064440  BIS   4IREST,IFLAGS  ;SET RESTART BIT IN FLAG.
13 107674 000521          BR    2$              INIT1
14 107676
15 107676 012700 000036          2$:
  107700 104447          MOV   4EF.CON,RO
  TRAP  C$REFG          ;HERE FROM CONTINUE COMMAND?
16
17 107704 103007          BCC   3$              ;BRANCH TO 3$ IF NOT, ELSE
18 107706 042737 000020 064440  BIC   4ISTRH,IFLAGS  ;CLEAR 1ST TIME THRU TEST 4 FLAG AND
19 107714 052737 000002 064440  BIS   4ICONT,IFLAGS  ;SET CONTINUE BIT IN FLAG.
20 107722 000476          BR    3$              13$
21 107724
22 107724 012700 000034          3$:
  107730 104447          MOV   4EF.PWR,RO
  TRAP  C$REFG          ;HERE FROM POWER FAIL?
23
24 107732 103001          BCC   4$              ;BRANCH TO 4$ IF NOT, ELSE
25 107734 000471          BR    4$              13$

```

```

1
2
3
4 107736 013705 064424
5 107742 052765 100000 000002
6 107750 010502
7 107752 062702 000020
8 107756 012703 000010
9 107762 012200
10 107764 C01403
11 107766 052760 100000 000002
12 107774 005303
13 107776 001371
14 110000 062705 000046
15 110004 005715
16 110006 001012
17 110010 062705 000046
18 110014 005715
19 110016 001351
20
21
22
23
24 110020 005003
25 110022
26 110022 010300
   110024 104442
27
28 110026 103030
29 110030 013705 064424
30 110034 021015
31 110036 001411
32 110040 062705 000046
33 110044 005715
34 110046 001372
35 110050
   110052 000006
   110054 000000
   110056 074400
37
38 110060 104444
39
40 110062 016001 000010
41 110066 004737 102274
42 110072 001366
43 110074 042765 100000 000002
44 110102 042764 100000 000002
45 110110 005203
46 110112 001337 002012
47 110116 001741
48 110120 001701 064632
52 110124 001700 001604
53 110130 001707 104530
55 110134 001737 111170

;MAKE ALL CONTROLLER/DRIVE TABLES NOT AVAILABLE FOR TESTING
4$: MOV CTABS,R5 ;GET ADDRESS OF 1ST CONTROLLER TABLE
5$: BIS #CT.AVL,C.UNIT(R5) ;SET CONTROLLER TABLE NOT AVAILABLE
MOV R5,R2 ;GET POINTER TO DRIVE TABLES
ADD #C.DR0,R2
MOV #8.,R3 ;GET NUMBER OF DRIVES PER CONTROLLER TABLE
6$: MOV (R2)+,R0 ;SEE IF THIS DRIVES HAS A TABLE,
BEQ 7$ ;BRANCH IF NOT, ELSE
BIS #DT.AVL,D.UNIT(R0) ;SET DRIVE TABLE NOT AVAILABLE.
7$: DEC R3 ;LOOK AT NEXT DRIVE IN CONTROLLER TABLE,
BNE 6$ ;BRANCH IF NO DRIVES, ELSE
ADD #C.SIZE,R5 ;LOOK AT NEXT CONTROLLER TABLE.
TST (R5) ;SEE IF THERE IS ANOTHER CONTROLLER TABLE.
BNE 9$ ;BRANCH IF SO, ELSE
ADD #C.SIZE,R5 ;MOVE TO NEXT CONTROLLER TABLE
TST (R5) ;IS THERE A NEXT ONE?
BNE 5$ ;IF SO, CLEAR THE BITS THERE

;NOW GET EACH P-TABLE AND MAKE THE APPROPRIATE CONTROLLER/DRIVE
;TABLES AVAILABLE FOR TESTING.
8$: CLR R3 ;START WITH LOGICAL UNIT 0
MOV R3,R0 ;GET POINTER TO IT'S P-TABLE
TRAP C$GPHRD
;BRANCH TO 12$ IF NOT AVAILABLE
BCC 12$
9$: MOV CTABS,R5 ;GET ADDRESS OF 1ST CONTROLLER TABLE
CMP (R0),(R5) ;SEE IF UDA ADDRESSES ARE THE SAME,
BEQ 11$ ;BRANCH IF SO, ELSE
ADD #C.SIZE,R5 ;LOOK AT NEXT CONTROLLER TABLE.
TST (R5) ;SEE IF THERE IS ANOTHER CONTROLLER TABLE.
BNE 9$ ;BRANCH IF SO, ELSE
10$: TRAP C$ERSF ;REPORT TABLE CONSISTANCY ERROR.
.WORD 6
.WORD 0
.WORD ERR006
;DO CLEAN-UP TRAP
TRAP C$DCLN

11$: MOV H.DRV(R0),R1 ;GET DRIVE NUMBER FROM P-TABLE
CALL GTDRVT ;FIND THE DRIVE TABLE ADDRESS
BNE 10$ ;BRANCH IF NOT FOUND, ELSE
BIC #CT.AVL,C.UNIT(R5) ;CLEAR AVAILABLE BIT IN CONTROLLER AND
BIC #DT.AVL,D.UNIT(R4) ;THE DRIVE TABLES.
12$: INC R3 ;INCREMENT TO NEXT UNIT IN P-TABLE
CMP #8.,R3 ;SEE IF ALL P-TABLES CHECKED,
BLT 8$ ;BRANCH IF NOT, ELSE
13$: MOV #15,TIME,R1 ;AT 15 MINUTES FROM NOW
MOV #01E+50.,R0 ;SET TIME FOR NEXT REPORT
CALL SETO
JMP IMPYKA ;EXIT THE INITIALIZE SECTION.

```

```

1
2
3
4
5 110140 005037 064626
6 110144 005037 064630
7 110150 012700 000114
  110154 104462
8 110156 103413
9 110160 012700 000120
  110164 104462
10 110166 103407
11 110170 005037 064616
12 110174 004137 075664
  110200 066466
  110202 000000
13 110204 000434
14
15 110206 012037 064616
16 110212 012037 064620
17 110216 012037 064622
18 110222 012037 064624
19
20 110226 012746 000340
  110232 012746 106230
  110236 013746 064622
  110242 012746 000003
  110246 104437
  110250 062706 000010
21 110254 012777 000105 154334
22 110262 012701 064632
26 110266 012700 001604
27 110272 004737 104530
29 110276 004737 106260
30 110302 104431
  110304 010037 064412
31 110310 017737 154076 064414
32 110316 005037 064444
33 110322 005037 064442

;INITIALIZE KW11 CLOCY, FREE MEMORY AND IP ADDRESS TABLE
;DURING START OR RESTART COMMAND ONLY

INIT1: CLR KW,EL ;CLEAR ELAPSED TIME
        CLR KW,EL+2
        MOV #L,R0
        TRAP C#CLCK
        BCS 1#
        MOV #P,R0
        TRAP C#CLCK
        BCS 1#
        CLR KW,CSR ;IF NEITHER, CLEAR CSR STORAGE WORD
        JSR R1,LPNTF ;CALL LPNTF PRINT ROUTINE
        .WORD NOCLOCK ;ADDRESS OF ASCIZ STRING
        .WORD ARG,CT ;ARGUMENT COUNT * 2
        BR 2#

1#: MOV (R0)+,KW,CSR ;STORE DATA RETURNED
    MOV (R0)+,KW,BRL
    MOV (R0)+,KW,VEC
    MOV (R0)+,KW,HZ

;SETUP KW11 VECTOR ADDRESS
MOV #PRI07,-(SP)
MOV #KW11I,-(SP)
MOV KW,VEC,-(SP)
MOV #3,-(SP)
TRAP C#SVEC
ADD #10,SP
MOV #KW.OUT,#KW.CSR ;START THE CLOCK
MOV #STIME,R1 ;AT 15 MINUTES FROM NOW
MOV #15,*60.,R0 ;SET TIME FOR NEXT REPORT

2#: CALL SETTO
    CALL RESET ;RESET ALL UDA'S
    TRAP C#MEM
    MOV R0,FFREE
    MOV #FFREE,FSIZE
    CLR TNUM
    CLR FNUM

;RESET SIZE OF FREE MEMORY
;INITIALIZE TEST NUMBER TO NO TEST RUNNING
;INITIALIZE FILE NUMBER TO NO FILE IN MEMORY

```

```

1
2
3           ;ALLOCATE DRIVE TABLES TO MEMORY
4 110326 013737 064412 064422 INIT2: MOV   FFREE,DTABS           ;STORE START OF DRIVE TABLES AND
5 110334 005077 154062           CLR   @DTABS           ;MARK ZERO END.
6 110340 013700 002012           MOV   L#UNIT,R0        ;GET NUMBER OF LOGICAL UNITS TO RUN.
7 110344 012701 000001           MOV   #1,R1           ;GET INITIAL SIZE OF DRIVE TABLE AND
8 110350 062701 000105           1$:  ADD   #<D.SIZE>/2,R1 ;ACCUMULATE DRIVE TABLE SIZE.
9 110354 005300           DEC   R0              ;SEE IF ANY MORE LOGICAL UNITS.
10 110356 001374           BNE   1$             ;BRANCH IF NOT, ELSE
11 110360 004737 075422           CALL  ALOCM          ;ALLOCATE ALL DRIVE TABLES TO MEMORY.
12                                     ; R1 POINTS TO 1ST WORD IN DRIVE TABLE
13
14           ;INITIALIZE CONTROLLER TABLE STORAGE WITH A WORD OF ZEROS
15
16 110364 013737 064412 064424 INIT3: MOV   FFREE,CTABS           ;STORE START OF CONTROLLER TABLES AND
17 110372 005077 154026           CLR   @CTABS           ;MARK ZERUS END.
18 110376 005037 064426           CLR   CTRLRS          ;CLEAR CONTROLLER COUNT
19 110402 012701 064534           MOV   #IPADRS,R1      ; R1 -> IP ADDRESS
20 110406 012702 000010           MOV   #8.,R2          ; R2 IS A COUNTER
21 110412 005021           1$:  CLR   (R1)+        ; CLEAR ENTRY
22 110414 005302           DEC   R2              ; DONE?
23 110416 001375           BNE   1$             ; IF NOT, BRANCH

```

```

1
2
3          ;BUILD CONTROLLER TABLES
4 110420 005005      INIT4: CLR      R5
5 110422 005002      CLR      R2
6 110424
7 110424 010200      1$:      MOV      R2,R0
   110426 104442      TRAP     C:GPHRD
8
9 110430 103156
10 110432 013703 064424
11 110436 005713      2$:      BCC     16$
12 110440 001435      MOV     CTABS,R3
13 110442 021013      TST     (R3)
16 110444 001017      BEQ     6$
   110446 016004 000004  MOV     H,BRL(R0),R4
   110452 000304      CMP     R4,C.VEC(R3)
   110454 006104      BNE     3$
   110456 056004 000002  ROL     R4
   110462 020463 000004  BIS     H,VEC(R0),R4
   110466 001004      CMP     R4,C.VEC(R3)
   110470 026063 000006 000006  BNE     3$
   110476 001461      CMP     H,BST(R0),C.BST(R3)
   110500 000137 111212  3$:      BEQ     11$
   110504 016304 000004  JMP     CTABER
   110510 042704 177000  4$:      MOV     C,VEC(R3),R4
   110514 026004 000002  BIC     *C<CT.VEC>,R4
   110520 001002      CMP     H,VEC(R0),R4
   110522 000137 111262  BNE     5$
   110526 062703 000046  JMP     SAMVEC
   110532 000741      ADD     *C,SIZE,R3
   110534 000741      BR     2$

```

;CLEAR CUSTOMER DATA FLAG  
;START WITH LOGICAL UNIT 0  
;GET POINTER TO IT'S P-TABLE  
  
;BRANCH TO 16\$ IF NOT AVAILABLE  
  
;GET ADDRESS OF 1ST CONTROLLER TABLE  
;CHECK IF ANY MORE TABLES  
;BUILD NEW TABLE IF FOUND ZERO WORD  
;CHECK IF SAME UNIBUS ADDRESS,  
;BRANCH IF NOT, ELSE  
;CHECK THAT OTHER PARAMETERS MATCH.  
;GET BR LEVEL FROM P-TABLE  
;SWAP TO HIGH BYTE  
;SHIFT ONE MORE TO LEFT  
;ADD VECTOR ADDRESS  
;COMPARE VECTOR AND BR LEVELS,  
;BRANCH IF DIFFERENT, ELSE  
;COMPARE BURST RATES,  
;BRANCH IF SAME, ELSE  
;FOUND SAME UDA WITH DIFFERENT  
;BR LEVEL, VECTOR ADDR'S OR BURST RATE.  
;GET VECTOR FROM CONTROLLER TABLE  
;AND  
;COMPARE VECTOR ADDRESSES,  
;BRANCH IF DIFFERENT, ELSE  
;FOUND TWO UDA'S WITH SAME VECTOR ADDRESS.  
  
;POINT TO BEGINNING OF NEXT CONTROLLER  
;TABLE IN MEMORY.

```

1
2
3
4 110534 012704 064534
5 110540 020427 064554
6 110544 101004
7 110546 005724
8 110550 001401
9 110552 000772
10
11 110554 011044
12
13 110556 012701 000023
14 110562 004737 075422
15
16
17 110566 011021
18 110570 010221
19 110572 016004 000004
20 110576 000304
21 110600 006104
22 110602 056004 000002
23 110606 010421
24 110610 016021 000006
25 110614 012721 004037
26 110620 012721 104520
27
28 110624 012704 000015
29 110630 005021
30 110632 005304
31 110634 002375
32 110636 005237 064426

```

```

;BUILD NEW CONTROLLER TABLE
6: MOV #IPADRS,R4 ;GET BEGINNING OF IP ADDRESS TABLE
7: CMP R4,#IPADRS+16. ;SEE IF END OF IP ADDRESS TABLE,
;BRANCH IF SO, ELSE
;DID WE FIND AN OPEN ENTRY ?
;BRANCH IF SO, ELSE
;LOOK AGAIN.
8: MOV (R0),(R4) ;TAKE UNIBUS ADDRESS FROM P-TABLE
;AND STORE IT IN THE IP ADDRESS TABLE.
9: MOV #<C.SIZE>/2,R1 ;GET # OF ENTRIES IN CONTROLLER TABLE
;AND ALLOCATE A TABLE TO MEMORY.
; R0 POINTS TO 1ST WORD P-TABLE
; R1 POINTS TO 1ST WORD IN CONTROLLER TABLE
MOV (R0),(R1)+ ;STORE UDA IP ADDRESS AND
MOV R2,(R1)+ ;LOGICAL UNIT NUMBER IN THE CONTROLLER TABLE.
MOV H,BRL(R0),R4 ;GET THE BR LEVEL,
SWAB R4 ;SWAP TO HIGH BYTE,
ROL R4 ;SHIFT ONE MORE TO LEFT,
BIS H,VEC(R0),R4 ;ADD VECTOR ADDRESS AND
MOV R4,(R1)+ ;STORE IT IN THE CONTROLLER TABLE.
MOV H,BST(R0),(R1)+ ;STORE THE BURST RATE,
MOV #4037,(R1)+ ;THE 'JSR R0' INSTRUCTION AND
MOV #UDASRV,(R1)+ ;THE ADDRESS OF THE INTERRUPT SERVICE
;ROUTINE IN THE CONTROLLER TABLE.
10: MOV #<C.SIZE-C.FLG>/2,R4 ;GET # OF ENTRIES TO END OF TABLE,
CLR (R1)+ ;CLEAR REST OF TABLE AND
DEC R4 ;ADD ZERO WORD AT END.
BGE 10: ;LOOP TIL ALL CLEARED
INC CTRLRS ;KEEP TRACK OF CONTROLLER COUNT

```

INITIALIZE SECTION

```

1
2
3          :BUILD DRIVE TABLES
4 110642 013701 064422 111:  MOV    DTABS,R1          ;GET ADDRESS OF CURRENT DRIVE TABLE
5 110646 062703 000020      ADD    @C.DRO,R3        ;INDEX TO 1ST DRIVE IN CONTROLLER TABLE
6 110652 012704 000010      MOV    @B.,R4          ;GET MAXIMUM # OF DRIVES PER CONTROLLER
7 110656 005713      121:  TEST   (R3)            ;ANY ENTRY TO DRIVE TABLE,
8 110660 001411      BEQ    141            ;BRANCH IF NOT, ELSE
9 110662 026033 000010      CMP    H.DRV(R0),@R3) ;COMPARE DRIVE NUMBER IN DRIVE TABLE,
10 110666 001002      BNE    131            ;BRANCH IF DIFFERENT, ELSE
11 110670 000137 111226      JMP    MLDREX          ;FOUND TWO P-TABLES WITH SAME DRIVE.
12
13 110674 005304      131:  DEC    R4              ;COUNT DRIVES
14 110676 001367      BNE    121            ;IF EIGHT DRIVE TABLES EXIST,
15 110700 000137 111244      JMP    TOOMER         ; THEN REPORT ERROR
16
17 110704 010113      141:  MOV    R1,(R3)        ;STORE ADDRESS OF DRIVE TABLE IN
18                                ;CONTROLLER TABLE.
19 110706 016021 000010      MOV    H.DRV(R0),(R1) ;STORE DRIVE NUMBER AND
20 110712 010221      MOV    R2,(R1)        ;LOGICAL UNIT NUMBER IN DRIVE TABLE.
21 110714 016011 000012      MOV    H.FRM(R0),(R1) ;GET TEST AREA BIT
22 110720 051105      BIS    (R1),R5        ;SAVE "OR" OF BIT FROM ALL DRIVES
23 110722 005111      COM    (R1)          ;COMPLIMENT IT
24 110724 042711 157777      BIC    @DEFM.Y1>,(R1) ;LOAD DEFAULT PARAMETER BITS
25 110730 052721 011012      BIS    @DEF,(R1)
26 110734 012704 000100      MOV    @D.SIZE/2>-3,R4 ;CLEAR REST OF TABLE
27 110740 005021      151:  CLR    (R1)
28 110742 005304      DEC    R4
29 110744 003375      BGT    151
30 110746 012761 177777 177754  MOV    *-1,<D.ECYL+2-D.SIZE>(R1) ;MARK CYLINDERS AT TEST ALL
31
32 110754 062737 000206 064422  ADD    @D.SIZE,DTABS ;NEXT DRIVE TABLE ADDRESS AND
33 110762 005077 152434      CLR    @DTABS         ;MARK ZERO END.
34 110766 005202      161:  INC    R2              ;INCREMENT LOGICAL UNIT NUMBER
35 110770 020237 002012      CMP    R2,L#UNIT      ;CHECK IF GOT ALL TABLES
36 110774 002613      BLT    11             ;IF NOT, GO BACK FOR NEXT, ELSE
37 110776 012701 000001      MOV    @1,R1          ;GET 1 WORD TO TERMINATE ALL CONTROLLER
38 111002 004737 075422      CALL  ALOCM           ;TABLES AND ALLOCATE IT TO MEMORY.

```

```

1
2
3      ;CHECK FOR CUSTOMER WARNING MESSAGE
4 111006 032705 020000      INITS:  BIT    @HM.CYL,R5      ;CHECK IF BIT EVER SET
5 111012 001460              BEQ    5$          ;BYPASS IF NOT
6 111014 004137 075664      JSR    R1,LPNTF    ;CALL LPNTF PRINT ROUTINE
    111020 065454              .WORD  INITWA      ;ADDRESS OF ASCIZ STRING
    111022 000000              .WORD  ARG.CT      ;ARGUMENT COUNT * 2
7 111024 013705 064424      MOV    CTABS,R5    ;GET ADDRESS 1ST CONTROLLER TABLE
8 111030 010504              1$:  MOV    R5,R4      ;GET ADDRESS OF POINTER TO DRIVE TABLE
9 111032 062704 000020      ADD    @C.DRO,R4
10 111036 012701 000010     MOV    @B.,R1
11 111042 012403              2$:  MOV    (R4)+,R3    ;GET COUNT OF DRIVE TABLES
12 111044 001422              BEQ    4$          ;GET ADDRESS OF DRIVE TABLE
13 111046 032763 020000 000004 BIT    @D.DCY,D.PRM(R3) ;CHECK IF CUSTOMER DATA SELECTED
14 111054 001014              BNE    3$
15 111056 011346              MOV    (R3),-(SP)
    111060 011546              MOV    (R5),-(SP)
    111062 016346 000002      MOV    D.UNIT(R3),-(SP)
    111066 012746 065560      MOV    @INITWB, -(SP)
    111072 012746 000004      MOV    @4, -(SP)
    111076 010600              MOV    SP,R0
    111100 104417              TRAP   C$PNTF
    111102 062706 000012      ADD    @12,SP
16 111106 005301              3$:  DEC    R1
17 111110 001354              BNE    2$          ;COUNT THE DRIVE TABLES
18 111112 062705 060046      4$:  ADD    @C.SIZE,R5 ;LOOK AT ALL OF THEM
19 111116 005715              TST   (R5)        ;MOVE TO NEXT CONTROLLER TABLE
20 111120 001343              BNE    1$          ;SEE IF ANOTHER TABLE AND
21                                     ;LOOK AT IT
22
23      ;GET CONFIRMATION TO PROCEED
24
25 111122 104450              TRAP   C$MANI
26 111124 103013              BCC   5$
27 111126 104443              TRAP   C$GMAN
    111130 000404              BR    10000$
    111132 064476              .WORD  TEMP
    111134 000120              .WORD  T$CODE
    111136 064763              .WORD  INITWC
    111140 000001              .WORD  1
    111142
28 111142 032737 000001 064476 10000$: BIT    @1.TEMP    ;LOOK AT RESPONSE
29 111150 001001              BNE    5$          ;BRANCH IF YES WAS ANSWER
30                                     ;DO CLEAN-UP TRAP
31 111152 104444              TRAP   C$DCLN
32
33      ;SAVE CURRENT PARAMETERS TO FREE MEMORY SO EACH TEST CAN USE ALL OF IT
34
35
36 111154 013737 064412 064416 5$:  MOV    F$FREE,F$HEM ;SAVE START ADDRESS
37 111162 013737 064414 064420      MOV    F$SIZE,F$MEMS ;SAVE SIZE

```

```
1  
2  
3  
4 111170  
111170 012700 000000  
111174 104441  
5 111176 005037 064656  
6 111202 004737 105612  
30  
31 111206 104432  
111210 000066
```

EXIT INITIALIZE SECTION  
INITXX:  
MOV #PRIO0,R0  
TRAP C\$SPRI  
CLR DLL  
CALL CLOSEF  
;ERASE DOWNLINE LOAD DATA  
;MAKE SURE DATA FILE IS CLOSED  
TRAP C\$EXIT  
.WORD L10044-.

```

1          .SBTTL  INITIALIZE ERRORS
2
3
4          ;DIFFERENT VECTORS, BR LEVELS OR BURST RATES FOR ONE CONTROLLER
5 111212  010305  CTABER: MOV      R3,R5          ;GET CONTROLLER ADDRESS
6 111214  104454  TRAP     C$ERSF
   111216  000001  .WORD    1
   111220  000000  .WORD    0
   111222  074252  .WORD    ERR001
7
8 111224  104444  TRAP     C$DCLN          ;DO CLEAN-UP TRAP
9
10         ;TWO P-TABLES FOR SAME DRIVE
11
12 111226  013705  064476  MLDREER: MOV     TEMP,R5        ;GET CON*ROLLER ADDRESS
13 111232  104454  TRAP     C$ERSF
   111234  000002  .WORD    2
   111236  000000  .WORD    0
   111240  074270  .WORD    ERR002
14
15 111242  104444  TRAP     C$DCLN          ;DO CLEAN-UP TRAP
16
17         ;MORE THAN EIGHT DRIVES SELECTED ON ONE CONTROLLER
18
19 111244  013705  064476  TOOMER: MOV     TEMP,R5        ;GET CONTROLLER ADDRESS
20 111250  104454  TRAP     C$ERSF
   111252  000003  .WORD    3
   111254  000000  .WORD    0
   111256  074306  .WORD    ERR003
21
22 111260  104444  TRAP     C$DCLN          ;DO CLEAN-UP TRAP
23
24         ;TWO UDA'S USE THE SAME VECTOR
25
26 111262  010305  SAMVEC: MOV     R3,R5          ;GET CONTROLLER ADDRESS
27 111264  104454  TRAP     C$ERSF
   111266  000010  .WORD    8
   111270  000000  .WORD    0
   111272  074412  .WORD    ERR008
28
29 111274  104444  TRAP     C$DCLN          ;DO CLEAN-UP TRAP
41 .EVEN
42
43 111276  L10044: TRAP     C$INIT
   111276  104411

```

J4

```
1          .SBTTL  AUTODROP SECTION
2
3          :++
4          : THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
5          : THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
6          : SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
7          : DROPPED FROM TESTING.
8          :--
9
10         L$AUTO:
11
18         L10045:
           TRAP      C$AUTO
           111300    104461
```

```

1          .SBTTL  CLEANUP CODING SECTION
2
3          ;**
4          ; THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
5          ; AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
6          ;--
7
8 111302    L$CLEAN::
9
10 111302  004737  105612          CALL  CLOSEF          ;CLOSE DATA FILE
12 111306  022737  000004  064444  CMP      #4,TNUM      ;ARE WE DOING TEST 4 ?
13 111314  001402          BEQ      1$              ;BRANCH IF SO, DON'T RESET BUS
16 111316  004737  106250          CALL  RESET          ;RESET ALL UDA'S
17 111322          1$:
27
28 111322  104432          TRAP  C$EXIT
   111324  000002          .WORD  L10046-.
29
41          .EVEN
42
43 111326          L10046:
   111326  104412          TRAP  C$CLEAN

```

DROP UNIT SECTION

```
1          .SBTTL DROP UNIT SECTION
2
3          ;**
4          ; THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
5          ; TO NO LONGER BE TESTED.
6          ;--
7
8 111330    L$DU::
9
18
19 11330    000167      .WORD  J$JMP
   '1' '2  000000      .WORD  L10047-2-.
20
21
22          .EVEN
23
24 111334    L10047:
   111334    104453    TRAP   C$DU
```

```

1      .SBTTL  ADD      UNIT SECTION
2
3
4      ;++
5      ; THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
6      ; TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
7      ; TO THE TEST CYCLE.
8      ;--
9      111336      L$AU::
10
11
12
13
14
15
16
17
18
19
20      111336      000167      .WORD  J$JMP
21      111340      000000      .WORD  L10050-2-.
22
23
24
25
26
27
28
29
30
31
32
33
34
35      111342      104452      L10050:
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

HARDWARE TESTS

```

2          .SBTTL  HARDWARE TESTS
14         .SBTTL  TEST 1: UNIBUS ADDRESSING TEST
51
54 111344          T1::
59 111344 012701 000001          MOV    #1,R1          ;INITIALIZE TEST PARAMETERS
60 111350 004737 076336          CALL   TINIT
61 111354 013737 064424 064430  MOV    CTABS,TSTTAB  ;GET ADDRESS OF 1ST CONTROLLER TABLE
62 111362 013705 064430          T1NEXT: MOV   TSTTAB,R5  ; GET CONTROLLER TABLE ADDRESS
63 111366 116537 000002 002074  MOVB   C.UNIT(R5),L#LUN ; CHECK IF UNIT AVAILABLE FOR TESTING
64 111374 005765 000002          TST    C.UNIT(R5)
65 111400 100010          BPL    T1NOW          ; TEST IF AVAILABLE
67 111402 062737 000046 064430  T1SKIP: ADD   #C.SIZE,TSTTAB ; MOVE TO NEXT CONTROLLER
68 111410 005777 153014          TST    @TSTTAB        ; CHECK IF ANOTHER CONTROLLER TABLE
69 111414 001362          BNE    T1NEXT
70 111416 104432          TRAP  C$EXIT
    111420 000776          .WORD  L10051-.
71
72 111422 004737 106260          T1NOW: CALL   RESET          ;RESET ALL UDA'S

```

1					
2	111426			T1.1:	
	111426	104402		TRAP	C1BSUB
3	111430	005037	064636	CLR	NXMAD
4					;CLEAR MEMORY ERROR FLAG
5	111434	012746	000340	MOV	@PRT07,-(SP)
	111440	012746	104510	MOV	@NXMI,-(SP)
	111444	012746	000004	MOV	@ERRVEC,-(SP)
	111450	012746	000003	MOV	@3,-(SP)
	111454	104437		TRAP	C1SVEC
	111456	062706	000010	ADD	@1C,SP
6	111462	011504		MOV	(R5),R4
7	111464	005714		TST	(R4)
8	111466	005764	000002	TST	2(R4)
9					;GET ADDRESS OF UDAIP REGISTER
10	111472	012700	000004	MOV	@ERRVEC,R0
	111476	104436		TRAP	C1CVEC
11	111500	005737	064636	TST	NXMAD
12	111504	001406		BEG	T1GOOD
13	111506	104455		TRAP	C1ERDF
	111510	000046		.WORD	38
	111512	000000		.WORD	0
	111514	075044		.WORD	ERR038
14	111516	104406		TRAP	C1CLP1
15	111520	000730		BR	T1SKIP
16	111522				;END TEST NOW
17	111522	104403		T1GOOD:	
				L10052:	
				TRAP	C1ESUB

```

1
2
3
4
5
6
7 111524
   111524 104402
8 111526 005014
9 111530 012737 004000 105545
10 111536 004737 105410
11 111542 103410
12 111544 012764 100000 000002
13 111552 012737 010000 105546
14 111560 004737 105410
15 111564
16 111564
   111564 104403

```

```

:
: MAKE SURE UDA PASSES INTERNAL DIAGNOSTIC
: MAKE SURE UDA CAN SENSE STEP 1 AND 2
:
T1.2:
TRAP C#SSUB
CLR (R4)
MOV #SA.S1,UDARSD
CALL UDARSP
BCS 1#
MOV #SA.STP.2(R4)
MOV #SA.S2,UDARSD
CALL UDARSP
: INIT UDA
: STEP 1 ASSERTED?
: WAIT FOR RESPONSE
: IF FAIL, EXIT
: SEND STEP 1
: STEP 2 ASSERTED?

```

```

1#:
L10053:
TRAP C#ESUB

```

TEST 1: UNIBUS ADDRESSING TEST

```

1
2
3
4
5
6 111566
   111566 104402
7 111570 011504
9 111572 005014
10 111574 012737 004000 105546
11 111602 004737 105410
12 111606 103444
13 111610 016437 000002 106650
14 111616 012764 140000 000002
15 111624 004737 106554
16 111630 001433
17 111632 022764 140000 000002
18 111640 001017
19 111642 012702 000001
20 111646 012703 000020
21 111652 016437 000002 106650
22 111660 010264 000002
23 111664 004737 106554
24 111670 001413
25 111672 020264 000002
26 111676 001405
27 111700
   111700 104455
   111702 000032
   111704 000000
   111706 074634
28 111710 000403
29 111712 006302
30 111714 005303
31 111716 001355
32 111720
33 111720
   111720 104403

```

```

;
; TEST THE DIAGNOSTIC LOOP MODE OF ALL UDA'S ON THE SYSTEM
;
T1.3:
TRAP C#BSUB
MOV (R5),R4 ; R4 POINTS TO UDAIP REGISTER
CLR (R4) ; INITIALIZE THE UDA
MOV #SA.S1,UDARSD ; LOCK FOR STEP 1
CALL UDARSP ; WAIT FOR RESPONSE
BCS 5# ; IF ERROR, BRANCH
MOV 2(R4),WCHNGD ; MOVE OLD PORT CONTENTS TO STORAGE
MOV #<SA.STP+SA.WRP>,2(R4) ; INITIALIZE FOR PORT WRAP
CALL WCHNG ; WAIT FOR THE PORT TO CHANGE
BEQ 5# ; IF ERROR, BRANCH
CMP #<SA.STP+SA.WRP>,2(R4) ; COMPARE WITH DATA WRITTEN
BNE 3#
1#: MOV #1,R2 ; SET UP FOR SHIFTING '1'
MOV #16,R3 ; SET UP LOOP COUNT
2#: MOV 2(R4),WCHNGD ; SAVE OLD PORT CONTENTS
MOV R2,2(R4) ; WRITE PATTERN TO UDASA FOR LOOP
CALL WCHNG ; WAIT FOR UDASA TO CHANGE
BEQ 5# ; IF ERROR, BRANCH
CMP R2,2(R4) ; COMPARE R0 WITH WHAT WAS ECHOED
BEQ 4# ; IF MATCH, BRANCH
3#: TRAP C#ERDF
; .WORD 26
; .WORD 0
; .WORD ERRO26
BR 5# ; BRANCH
4#: ASL R2 ; MOVE THE SHIFTING ONE LEFT BY 1
DEC R3 ; DECREMENT COUNT
BNF 2# ; IF LOOP INCOMPLETE, BRANCH
5#:
L10054: TRAP C#ESUB

```



49	112150	012700	000340		MOV	0PRI07,R0		
	112154	104441			TRAP	C#SPRI		
50	112156	005064	000002		CLR	2(R4)		
51	112162	012702	000144		MOV	0100.,R2		; WRITE SECOND STEP TO UDA
52	112166	005302		4\$:	DEC	R2		; SET UP DELAY SO WE KNOW WE'RE INTERRUPTED
53	112170	001376			BNE	4\$		; DECREMENT COUNT
54	112172	012701	000007		MOV	07.,R1		; IF INCOMPLETE, BRANCH
55	112176			5\$:				; R1 IS PROCESS PRIORITY LEVEL
	112176	010146			MOV	R1,-(SP)		
56	112200	012702	000005		MOV	05.,R2		; PUSH R1 ON STACK
57	112204	006301		6\$:	ASL	R1		; SET UP FOR SHIFTING PRIORITY
58	112206	005302			DEC	R2		; SHIFT PRIORITY
59	112210	001375			BNE	6\$		; DECREMENT SHIFT COUNT
60	112212	010100			MOV	R1,R0		; IF INCOMPLETE, BRANCH
	112214	104441			TRAP	C#SPRI		
61	112216	012601			MOV	(SP)+,R1		; POP STACK INTO R1
62	112220	005737	064454		TST	INTRCV		; SEE IF INTERRUPT RECEIVED
63	112224	001007			BNE	8\$		; IF SO, BRANCH
64	112226	005301			DEC	R1		; DECREMENT PRIORITY LEVEL
65	112230	100362			BPL	5\$		; IF ALL LEVELS UNTESTED, BRANCH
66	112232			7\$:				
	112232	104455			TRAP	C#ERDF		
	112234	000034			.WORD	28		
	112236	000000			.WORD	0		
	112240	074672			.WORD	ERRO23		
67	112242	000420			BR	10\$		; BRANCH
68								
69	112244			8\$:				
	112244	012700	000000		MOV	0PRI00,R0		
	112250	104441			TRAP	C#SPRI		
70	112252	005201			INC	R1		; SO PRIORITY = BR LEVEL
71	112254	023701	106652		CMF	BRLEV,R1		; SEE IF BR LEVEL MATCHES PRIORITY
72	112260	001405			BEQ	9\$		; IF SO, BRANCH
73	112262	104455			TRAP	C#ERDF		
	112264	000035			.WORD	29		
	112266	000000			.WORD	0		
	112270	074704			.WORD	ERRO23		
74	112272	000404			BR	10\$		; BRANCH
75	112274			9\$:				
	112274	004137	075704		JSR	R1,LPNTX		; CALL LPNTX PRINT ROUTINE
	112300	065437			.WORD	INTST1		; ADDRESS OF ASCII STRING
	112302	000000			.WORD	ARG,CT		; ARGUMENT COUNT + 2
76	112304	016503	000004	10\$:	MOV	C,VEC(R5),R3		; GET VECTOR ADDRESS
77	112310	042703	177000		BIC	0#CCT,VEC,R3		; CLEAR UNUSED BITS
78	112314	010300			MOV	R3,R0		
	112316	104436			TRAP	C#CVEC		
79	112320			L10055:				
	112320	104403			TRAP	C#ESUB		

G5

TEST 1: UNIBUS ADDRESSING TEST

1  
2 112322  
112322 104402  
3 112324 005004  
4 112326 004737 104612  
5 112332  
112332 104403

T1.5:  
TRAP C#BSUB  
CLR R4  
CALL UDAINT  
L10056:  
TRAP C#ESUB

; INITIALIZE UDA WITH SMALLEST  
; RING BUFFER AND INTERRUPTS DISABLED

H5

1						
2	112334			T1.6:		
	112334	104402				
3	112336	012704	126400		TRAP	C#BSUB
4	112342	004737	104612		MOV	*<SA.STP+<5*SA.MS1>+<5*SA.CM1>>,R4
5					CALL	UDAIN
6						; INITIALIZE UDA WITH RING BUFFER
7						; LARGE ENOUGH TO COVER NORMAL
						; HOST COMM AREA PACKET AND BUFFER
						; SPACE (A 5 IN MESSAGE LENGTH AND
						; A 5 IN COMMAND LENGTH)
8	112346			L10057:		
	112346	104403			TRAP	C#ESUB

TEST 1: UNIBUS ADDRESSING TEST

1					
2	112350			T1.7:	
	112350	104402			TRAP C\$BSUB
3	112352	013746	064412		MOV FFREE, -(SP) ; PUSH FFREE ON STACK
	112356	013746	064414		MOV FSIZE, -(SP) ; PUSH FSIZE ON STACK
4	112362	012701	000001		MOV #1, R1 ; RUN DM PROGRAM IN
5	112366	004737	076532		CALL RUNDM ; ONE CONTROLLER ONLY
6	112372	001402			BEQ 1\$
7	112374	004737	076626		CALL RESPDM
8	112400			1\$:	
	112400	012637	064414		MOV (SP)+, FSIZE ; POP STACK INTO FSIZE
	112404	012637	064412		MOV (SP)+, FFREE ; POP STACK INTO FFREE
9	112410			L10060:	
	112410	104403			TRAP C\$ESUB
10	112412	000137	111402		JMP T1SKIP
27					.EVEN
28					
29	112416			L10051:	
	112416	104401			TRAP C\$ETST
30					

```

1
2          .SBTTL TEST 2: DISK RESIDENT DIAGNOSTIC TEST
3 112420
8 112420 012701 000002          T2::      MOV     #2,R1          ;INITIALIZE TEST PARAMETERS
9 112424 004737 076336          CALL    TINIT
10 112430 013737 064424 064430  MOV     CTABS,TSTTAB    ;GET ADDRESS TO 1ST CONTROLLER TABLE
11
12 112436 004737 106260          1$:     CALL    RESET          ;RESET ALL UDA'S
13 112442 013746 064412          MOV     FFREE,-(SP)     ;;PUSH FFREE ON STACK
    112446 013746 064414          MOV     FSIZE,-(SP)    ;;PUSH FSIZE ON STACK
14 112452 012701 000001          MOV     #1,R1          ;RUN DM PROGRAM IN
15 112456 004737 076532          CALL    RUNDM          ; ONE CONTROLLER ONLY
16 112462 001402
17 112464 004737 076626          BEQ     2$
    112470 012637 064414          CALL    RESPDM
18 112470 012637 064414          2$:     MOV     (SP)+,FSIZE    ;;POP STACK INTO FSIZE
    112474 012637 064412          MOV     (SP)+,FFREE    ;;POP STACK INTO FFREE
19 112500 062737 000046 064430  ADD     #C.SIZE,TSTTAB ;MOVE TO NEXT CONTROLLER
20 112506 005777 151716          TST     @TSTTAB        ;CHECK IF ANY MORE CONTROLLER TABLES
21 112512 001351
38          BNE     1$
39          .EVEN
40 112514          L10061
    112514 104401          TRAP   C$ETST
41

```

```

1
2          .SBTTL TEST 3: DISK FUNCTION TEST
3 112516
8 112516 012701 000003
9 112522 004737 076336
10 112526 013737 064424 064430
11 112534 013701 064426
12 112540 004737 076532
13 112544 001402
14 112546 004737 076626
15 112552
32
33
34 112552
112552 104401
35

T3::
MOV      03,R1          ;INITIALIZE TEST PARAMETERS
CALL     TINIT
MOV      CTABS,TSTTAB  ;GET ADDRESS OF 1ST TABLE ADDRESS
MOV      CTRLRS,R1    ;RUN DM PROGRAM ON ALL CONTROLLERS
CALL     RUNDM
BEQ      1$           ; AT ONCE
CALL     RESPDM

1$:
.EVEN

L10062:
TRAP     C4ETST

```

```

3
4      .SBTTL TEST 4: DISK EXERCISER
5 112554
13 112554 022737 000004 064444 T4::      CMP      #4,TNUM      ;CHECK IF TEST 4 WAS IN PROGRESS
15 112562 001053      BNE      T4STRT     ;BRANCH IF NOT
16 112564 022737 000002 064440      CMP      #ICONT,IFLAGS ;CHECK IF HERE BY CONTINUE COMMAND
17 112572 001047      BNE      T4STRT     ;BRANCH IF NOT
18 112574 005037 064440      CLR      IFLAGS     ;CLEAR FLAGS FOR NEXT TIME HERE
19 112600 013704 064650      MOV      LBUFS,R4   ;GET LOG BUFFER POINTER
20 112604 001423      BEQ      LOGCHK     ; IF ZERO, NONE EXISTS
21 112606 004137 075664      JSR      R1,LPNTF   ;CALL LPNTF PRINT ROUTINE
      112612 066543      .WORD   LOGM1      ;ADDRESS OF ASCIZ STRING
      112614 000000      .WORD   ARG.CT     ;ARGUMENT COUNT * 2
22 112616 005037 064650      CLR      LBUFS     ;CLEAR START ADDRESS TO ERASE BUFFER
23 112622 012405      LOGOUT: MOV      (R4)+,R5 ;GET CONTROLLER TABLE ADDRESS
24 112624 004737 103250      CALL    PNTERR     ;PRINT ERROR REPORT
25 112630 062704 000104      ADD     #<HC.BSZ-2>,R4 ;BUMP POINTER TO NEXT ENTRY
26 112634 020437 064652      CMP     R4,LBUFN   ;CHECK IF AT END
27 112640 103770      BLO LOGOUT        ;PRINT ALL ENTRIES
28 112642 004137 075664      JSR     R1,LPNTF   ;CALL LPNTF PRINT ROUTINE
      112646 066575      .WORD   LOGM2      ;ADDRESS OF ASCIZ STRING
      112650 000000      .WORD   ARG.CT     ;ARGUMENT COUNT * 2
29 112652 000410      BR     T4CON
30
31 112654 032737 001000 064400 LOGCHK: BIT     #SM.LOG,SFPTBL+50.BIT ;CHECK IF LOG ENABLED
32 112662 001404      BEQ     T4CON
33 112664 004137 075664      JSR     R1,LPNTF   ;CALL LPNTF PRINT ROUTINE
      112670 066622      .WORD   LOGM3      ;ADDRESS OF ASCIZ STRING
      112672 000000      .WORD   ARG.CT     ;ARGUMENT COUNT * 2
34 112674 005737 064450      T4CON: TST     URNING ;CHECK IF ANY CONTROLLERS STILL RUNNING
35 112700 001404      BEQ     T4STRT     ;RESTART IF NOT
36 112702 004737 076626      CALL    RESPDM    ;CONTINUE BY RESPONDING TO REQUESTS
37 112706 000137 113240      JMP     T4WAIT     ;END OF TEST WHEN DONE

```

```

1
2
3
; START TEST
10 112712 012701 000004 T4STRT: MOV #4,R1 ; INITIALIZE TEST PARAMETERS
12 112716 004737 076336 CALL TINIT
13 112722 032737 000014 064440 BIT #ISTRT!IREST,IFLAGS ; HERE FROM OPERATOR COMMAND?
14 112730 001521 BEQ T4RUN ; RUN WITH PREVIOUS PARAMETERS IF NEW PASS
15 112732 032737 000200 064400 BIT #SM.MAN,SFPTBL+SO.BIT ; MANUAL INTERVENTION MODE?
16 112740 001463 BEQ T4DEF ; IF NOT, SET UP DEFAULT PARAMETERS
17 112742 104450 TRAP C#MANI
18 112744 103055 BCC T4DEFW
20 112746 012701 000004 MOV #4,R1 ; R1 = T4QUEST FILE NUMBER
21 112752 020137 064442 CMP R1,FNUM ; IS IT ALREADY LOADED?
22 112756 001406 BEQ 1$ ; IF SO, BRANCH
23 112760 005005 CLR R5 ; ELSE R5 = ADJUSTED ADDRESS
24 112762 004737 105630 CALL RDREC ; READ IN FILE
25 112766 103002 BCC 1$ ; IF OK, BRANCH
26 112770 000137 076424 JMP TINITE ; ELSE, ERROR
28
29 ; INPUT PARAMETERS
30
31 112774 005037 064452 1$: CLR UCNT ; CLEAR COUNT OF UNITS USING PATTERN 16
32 113000 013705 064424 MOV CTABS,R5 ; GET ADDRESS OF 1ST CONTROLLER TABLE
33 113004 012702 000010 T4PRM1: MOV #8.,R2 ; GET COUNT OF DRIVE TABLES
34 113010 010504 MOV R5,R4 ; GET FIRST DRIVE TABLE POINTER
35 113012 062704 000020 ADD #C.DRO,R4
36 113016 012403 T4PPM2: MOV (R4)+,R3 ; GET DRIVE TABLE ADDRESS
37 113020 001416 BEQ T4PRM4 ; GO TO NEXT CONTROLLER IF NONE
38 113022 032763 100000 000002 BIT #DT.AVL,D.UNIT(R3) ; SEE IF TO BE TESTED
39 113030 001010 BNE T4PRM3
41 113032 004737 002122 CALL STORAG ; ASK QUESTIONS
45 113036 022763 000020 000006 CMP #16.,D.PAT(R3)
46 113044 001002 BNE T4PRM3
47 113046 005237 064452 JNC UCNT
48 113052 005302 T4PRM3: DEC R2 ; COUNT DRIVE TABLES
49 113054 001360 BNE T4PRM2 ; GO LOOK AT NEXT
50 113056 062705 000046 T4PRM4: ADD #C.SIZE,R5 ; GO TO NEXT CONTROLLER
51 113062 005715 TST (R5) ; IF THERE IS ONE
52 113064 001347 BNE T4PRM1
53 113066 012701 064554 MOV #PAT16C,R1 ; R1 -> PAT16C FOR INPUT
55 113072 004737 002124 CALL STORAG+2 ; ASK LAST QUESTIONS
60 113076 000436 BR T4RUN

```





2  
13  
14  
42  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
55  
56  
57  
58  
59  
60  
61  
62  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
75  
76  
77  
79

113264 000032  
113266  
000000  
000002  
000004  
000006  
000010  
000012  
020000  
113266 000031  
113270 113352  
113272 160000  
113274 177774  
113276 001031  
113300 113400  
113302 000004  
113304 000774  
113306 002052  
113310 113407  
113312 177777  
113314 000004  
113316 000007  
113320 003052  
113322 113420  
113324 177777  
113326 000000  
113330 000077  
113332 004052  
113334 113442  
113336 177777  
113340 000000  
113342 000377  
113344 005130  
113346 113452  
113350 020000

```
.TITLE PARAMETER CODING
.SBTTL  HARDWARE PARAMETER CODING SECTION

! **
! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
! WITH THE OPERATOR.
! --

        .WORD L10064-L$HARD/2
L$HARD::
H.UBA   = 0           ;UNIBUS ADDRESS
H.VEC   = 2           ;UDA VECTOR
H.BRI   = 4           ;BR LEVEL
H.BST   = 6           ;BURST RATE
H.DRV   = 10          ;DRIVE NUMBER
H.PRM   = 12          ;PROGRAM PARAMETERS

HM.CYL  == BIT13     ;TEST CUSTOMER DATA AREA
                        ;PRINT 'UNIBUS ADDRESS OF UDA?'

        .WORD  T#CODE
        .WORD  MSGUBA
        .WORD  T#LLOLIM
        .WORD  T#HILIM

                        ;PRINT 'VECTOR?'

        .WORD  T#CODE
        .WORD  MSGVEC
        .WORD  T#LLOLIM
        .WORD  T#HILIM

                        ;PRINT 'BR LEVEL?'

        .WORD  T#CODE
        .WORD  MSGBRI
        .WORD  -1
        .WORD  T#LLOLIM
        .WORD  T#HILIM

                        ;PRINT 'UNIBUS BURST RATE?'

        .WORD  T#CODE
        .WORD  MSGBST
        .WORD  -1
        .WORD  T#LLOLIM
        .WORD  T#HILIM

                        ;PRINT 'DRIVE #?'

        .WORD  T#CODE
        .WORD  MSGLDR
        .WORD  -1
        .WORD  T#LLOLIM
        .WORD  T#HILIM

                        ;PRINT 'EXERCISE ON CUSTOMER DATA AREA
                        ;      IN TEST 4?'

        .WORD  T#CODE
        .WORD  MSGCST
        .WORD  HM.CYL
        .EVEN
```

D6

80	113352				L10064:	
84	113352	125	116	111	MSGUBA:	.ASCIZ \UNIBUS ADDRESS OF UDA\
85	113400	126	105	103	MSGVEC:	.ASCIZ \VECTOR\
86	113407	102	122	040	MSGBRL:	.ASCIZ \BR LEVEL\
87	113420	125	116	111	MSGBST:	.ASCIZ \UNIBUS BURST RATE\
88	113442	104	122	111	MSGLDR:	.ASCIZ \DRIVE #\
90	113452	105	130	105	MSGCST:	.ASCIZ \EXERCISE ON CUSTOMER DATA AREA IN TEST 4\
92						.EVEN
96						
106						

```

1      .SBTTL SOFTWARE PARAMETER CODING SECTION
2
3
4      ;**
5      ; THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
6      ; THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
7      ; MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
8      ; INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
9      ; MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
10     ; WITH THE OPERATOR.
11     ;--
12     113524 000030      .WORD L10065-L$SOFT/2
13     113526
14     000000      L$SOFT::
15     000002      SO.EL = 0      ;ERROR LIMIT
16     000004      SO.XL = 2      ;DATA TRANSFER LIMIT (MEGABITS)
25     000004      SO.BIT = 4     ;SINGLE BIT ANSWERS
26     000200      SM.MAN == BIT07 ; MANUAL INTERVENTION MODE
27     000400      SM.SSF == BI108 ; SUPPRESS SOFT ERRORS
28     001000      SM.LOG == BIT09 ; ERROR LOG ENABLED
29     040000      SM.IW == BIT14  ; INITIAL WRITE

```

```

1
2
3
4 113526 002130      .WORD  T#CODE
   113530 113606      .WORD  S.MAN
   113532 000200      .WORD  SM.MAN
7
8
9 113534 000003      .WORD  T#CODE
   113536 113673      .WORD  S.MES
11
12 113540 000052      .WORD  T#CODE
   113542 113756      .WORD  S.EL
   113544 177777      .WORD  -1
   113546 000001      .WORD  T#LOLIM
   113550 177777      .WORD  T#HILIM
13
14
15 113552 001052      .WORD  T#CODE
   113554 113772      .WORD  S.XL
   113556 177777      .WORD  -1
   113560 000000      .WORD  T#LOLIM
   113562 177777      .WORD  T#HILIM
16
17 113564 002130      .WORD  T#CODE
   113566 114054      .WORD  S.SSF
   113570 000400      .WORD  SM.SSF
18
19 113572 002130      .WORD  T#CODE
   113574 114112      .WORD  S.IW
   113576 040000      .WORD  SM.IW
20
21 113600 002130      .WORD  T#CODE
   113602 114144      .WORD  S.LOG
   113604 001000      .WORD  SM.LOG
32
   113606                L10065:
33
37 113606      105      116      124  S.MAN:  .ASCIZ  \ENTER MANUAL INTERVENTION MODE FOR SPECIAL DIAGNOSIS\
40 113673      122      105      115  S.MES:  .ASCIZ  \REMAINING SOFTWARE QUESTIONS APPLY TO TEST 4 ONLY\
42 113755      000
43 113756      105      122      122  S.EL:   .ASCIZ  \ERROR LIMIT\
44 113772      122      105      101  S.XL:   .ASCIZ  \READ TRANSFER LIMIT IN MEGABYTES - 0 FOR NO LIMIT\
45 114054      123      125      120  S.SSF:  .ASCIZ  \SUPPRESS PRINTING SOFT ERRORS\
46 114112      104      117      040  S.IW:   .ASCIZ  \DO INITIAL WRITE ON START\
47 114144      105      116      101  S.LOG:  .ASCIZ  \ENABLE ERROR LOG\
56
60
69
84 114166
85                000050      $PATCH::
100 114306 114332      .REPT 40.
   114310 000010      .EVEN
   114312                .WORD  T#FREE
                        .WORD  T#SIZE
                        L#LAST::

```

G6

```
1
14
16 114312 000000      .WORD 0
   114314 000006      .WORD L10070 ,/2-1
   114316                                L10066:
17 114316 172150      .WORD 172150      ; UNIBUS ADDRESS
18 114320 000154      .WORD 154         ; VECTOR ADDRESS
19 114322 000005      .WORD 5           ; BR LEVEL
20 114324 000077      .WORD 63         ; UNIBUS BURST RATE
21 114326 000000      .WORD 0           ; DRIVE NUMBER
22 114330 000000      .WORD 0           ; COSTUMER DATA AREA
23 114332
25                                L10070:
                                .END
```

ADR	=	000020	G	CALR5	102142	C#DU	=	000053	C.UNIT	=	000002	D.DC	=	000002		
ALOCM	=	075422		CALR6	102220	C#EDIT	=	000003	C.VEC	=	000004	D.DCA	=	000001		
APRINT	=	104472		CALR7	102234	C#ERDF	=	000055	DDEF	=	011012	D.DCY	=	020000		
APRIZ	=	104506		CALR8	102252	C#ERHR	=	000056	DFPTBL	=	064356	D.DRV	=	000000		
ARG.CT	=	000000		CALR9	102270	C#ERRO	=	000060	DIAG	=	177777	D.ECC	=	010000		
ASS	=	100000		CF.ATN	=	000200	C#ERSF	=	000054	DIAGMC	=	000000	D.ECCC	=	000176	
ASSEMB	=	000010		CF.MSC	=	000100	C#ERSO	=	000057	DIVIDE	=	102704	D.ECYL	=	000160	
A1	=	000001		CF.OTH	=	000040	C#ESCA	=	000010	DIV10	=	102742	D.END1	=	000120	
A2	=	000002		CF.SHD	=	000002	C#ESEG	=	000005	DLL	=	064656	D.END2	=	000130	
A3	=	000020		CF.THS	=	000020	C#ESUB	=	000003	DLLADR	=	064666	D.END3	=	000140	
A4	=	000040		CF.576	=	000001	C#ETST	=	000001	DLLDR	=	064660	D.END4	=	000150	
BAS	=	067047		CLOG	=	105550	C#EXIT	=	000032	DLLNAM	=	064674	D.HERR	=	000170	
BASLN	=	067050		CLOSEF	=	105612	C#GEID	=	000026	DLLR	=	064664	D.IW	=	040000	
BASL1	=	066775		CLRBFL	=	104304	C#GETW	=	000027	DLLSIZ	=	064672	D.PAT	=	000006	
BASL2	=	067013		CLRBUF	=	104256	C#GMAN	=	000043	DLLV	=	064662	D.PRM	=	000004	
BASL3	=	067032		CON.A	=	076062	C#GPHR	=	000042	DMFRST	=	001000	D.RET	=	001000	
BASNO	=	066651		CON.A1	=	076066	C#GPLD	=	000030	DMMAIN	=	000040	D.RO	=	004000	
BASN1	=	066670		CON.A2	=	076110	C#GPRI	=	000040	DMOVRL	=	000004	D.SEEK	=	000174	
BASN2	=	066714		CON.D	=	076112	C#INIT	=	000011	DMPROG	=	064432	D.SEQ	=	000100	
BASN3	=	066734		CON.H	=	076124	C#INLP	=	000020	DMTRLN	=	000000	D.SERN	=	000200	
BASN4	=	066754		CON.N	=	076150	C#INPI	=	000050	DONE	=	101722	D.SERR	=	000172	
BELL	=	000007	G	CON.N1	=	076154	C#MEM	=	000031	DSPSIZ	=	000017	D.SIZE	=	000206	
BIT0	=	000001	G	CON.O	=	076136	C#MSG	=	000023	DTABS	=	064422	D.TR	=	000020	
BIT00	=	000001	G	CON.QU	=	076042	C#OPEN	=	000034	DT.AVL	=	100000	D.UNIT	=	000002	
BIT01	=	000002	G	CON.QX	=	076060	C#PNTB	=	000014	DT.UNT	=	000077	D.WC	=	000010	
BIT02	=	000004	G	CON.R	=	076172	C#PNTF	=	000017	DUP	=	001000	D.WCA	=	000004	
BIT03	=	000010	G	CON.R1	=	076214	C#PNTS	=	000016	DU.DFL	=	020000	D.WO	=	002000	
BIT04	=	000020	G	CON.S	=	076230	C#PNTX	=	000015	DU.FTL	=	050000	D.XFRR	=	000166	
BIT05	=	000040	G	CON.S1	=	076234	C#QIO	=	000377	DU.INF	=	030000	D.XFRW	=	000164	
BIT06	=	000100	G	CR	=	000015	C#RDBU	=	000007	DU.QUE	=	010000	D.ZERO	=	140200	
BIT07	=	000200	G	CRLF	=	065042	C#REFG	=	000047	DU.SPC	=	060000	EF.BBR	=	000200	
BIT08	=	000400	G	CTABER	=	111212	C#RESE	=	000033	DU.TER	=	040000	EF.BBU	=	000100	
BIT09	=	001000	G	CTABS	=	064424	C#REVI	=	000003	D.BB	=	000010	EF.CON	=	000036	G
BIT1	=	000002	G	CTRLRS	=	064426	C#RFLA	=	000021	D.BB01	=	000012	EF.LOG	=	000040	
BIT10	=	002000	G	CT.AVL	=	100000	C#RPT	=	000025	D.BB02	=	000016	EF.NEW	=	000035	G
BIT11	=	004000	G	CT.BRL	=	007000	C#SEFG	=	000046	D.BB03	=	000022	EF.PWR	=	000034	G
BIT12	=	010000	G	CT.CMD	=	000004	C#SPRI	=	000041	D.BB04	=	000026	EF.RES	=	000037	G
BIT13	=	020000	G	CT.MSG	=	000010	C#SVEC	=	000037	D.BB05	=	000032	EF.SEX	=	000020	
BIT14	=	040000	G	CT.REQ	=	000020	C#TPRI	=	000013	D.BB06	=	000036	EF.STA	=	000040	G
BIT15	=	100000	G	CT.RN	=	000002	C.BST	=	000006	D.BB07	=	000042	EN	=	040000	
BIT2	=	000004	G	CT.UNT	=	000077	C.DR0	=	000020	D.BB08	=	000046	EO	=	140000	
BIT3	=	000010	G	CT.U50	=	000040	C.DR1	=	000022	D.BB09	=	000052	ERRBLK	=	064410	G
BIT4	=	000020	G	CT.VEC	=	000777	C.DR2	=	000024	D.BB10	=	000056	ERRC	=	076304	
BIT5	=	000040	G	C#AU	=	000052	C.DR3	=	000026	D.BB11	=	000062	ERRD	=	076316	
BIT6	=	000100	G	C#AUT0	=	000061	C.DR4	=	000030	D.BB12	=	000066	ERRLIM	=	065245	
BIT7	=	000200	G	C#BRK	=	000022	C.DR5	=	000032	D.BB13	=	000070	ERRMC	=	101470	
BIT8	=	000400	G	C#BSEG	=	000004	C.DR6	=	000034	D.BB14	=	000076	ERRMES	=	101250	
BIT9	=	001000	G	C#BSUB	=	000002	C.DR7	=	000036	D.BB15	=	000102	ERRME1	=	065101	
BLDCMD	=	104124		C#CEFG	=	000045	C.FLG	=	000014	D.BB16	=	000106	ERRMSG	=	064406	G
BLDC0	=	104164		C#CLCK	=	000062	C.HCOM	=	000016	D.BCYL	=	000154	ERRMSL	=	101356	
BLDC1	=	104172		C#CLEA	=	000012	C.JAD	=	000012	D.BE	=	000040	ERRMSX	=	101464	
BOE	=	000400	G	C#CLOS	=	000035	C.JSR	=	000010	D.BEC	=	000112	ERRNBR	=	064404	G
BRLEV	=	106652		C#CLP1	=	000006	C.REF	=	000044	D.BGN1	=	000114	ERRTYP	=	064402	G
CALR1	=	101726		C#CVEC	=	000036	C.SIZE	=	000046	D.BGN2	=	000124	ERRVEC	=	000004	G
CALR2	=	101754		C#DCLN	=	000044	C.TO	=	000040	D.BGN3	=	000134	ERR. ?	=	000011	
CALR3	=	102052		C#DODU	=	000051	C.TOH	=	000042	D.BGN4	=	000144	ERR.TB	=	076252	
CALR4	=	102066		C#DRPT	=	000024	C.UADR	=	000000	D.CYL	=	000400	ERR.TN	=	075250	G

ERR001	074252	G	F#PROT=	000021	H.DRV =	000010	KW11I	106230	G	L#MPCP	002016	G
ERR002	074270	G	F#PWR =	000017	H.PRM =	000012	LBUFE	064654		L#HPTP	002022	G
ERR003	074306	G	F#RPT =	000012	H.UBA =	000000	LBUFN	064652		L#HW	064356	G
ERR004	074324	G	F#SEG =	000003	H.VEC =	000002	LBUFS	064650		L#ICP	002104	G
ERR005	074336	G	F#SOFT=	000005	IBE =	010000	LDDM	076554		L#INIT	107636	G
ERR006	074400	G	F#SRV =	000010	ICONT =	000002	LDNEXT	076606		L#LADP	002026	G
ERR007	074350	G	F#SUB =	000002	IDU =	000040	LF =	000012	G	L#LAST	114312	G
ERR008	074412	G	F#SW =	000014	IER =	020000	LOAD	104006		L#LOAD	002100	G
ERR014	074362	G	F#TEST=	000001	IFLAGS	064440	LOADB	103600		L#LUN	002074	G
ERR021	074430	G	GETCDN	102430	INITBL	105276	LOADDM	103410		L#MREV	002050	G
ERR022	074470	G	GETCNT	102364	INITWA	065454	LOADER	104120		L#NAME	002000	G
ERR023	074516	G	GETCNX	102370	INITWB	065560	LOADER1	104110		L#PRIO	002042	G
ERR024	074602	G	GETCXX	102436	INITWC	064763	LOADT1	103674		L#PROT	107630	G
ERR025	074616	G	GTORVT	102274	INITXX	111170	LOE =	040000	G	L#PRT	002112	G
ERR026	074634	G	GTT452	076500	INIT1	110140	LOG =	000001		L#REPP	002062	G
ERR027	074654	G	G#CNTD=	000200	INIT2	110326	LOGCHK	112654		L#REV	002010	G
ERR028	074672	G	G#DELM=	000372	INIT3	110364	LOGM1	066543		L#RPT	106654	G
ERR029	074704	G	G#DISP=	000003	INIT4	110420	LOGM2	066575		L#SOFT	113526	G
ERR030	074720	G	G#EXCP=	000400	INIT5	111006	LOGM3	066622		L#SPC	002056	G
ERR031	074734	G	G#HILI=	000002	INTRCV	064454	LOGOUT	112622		L#SPCP	002020	G
ERR032	074746	G	G#LOLI=	000001	INTSRV	106252	LOT =	000010	G	L#SPTP	002024	G
ERR033	074764	G	G#NO =	000000	INTST0	065342	LPNT	075722		L#STA	002030	G
ERR034	074772	G	G#OFFS=	000400	INTST1	065437	LPNTB	075674		L#SW	064374	G
ERR035	075000	G	G#OF SI=	000376	IPADRS	064534	LPNTF	075664		L#TEST	002114	G
ERR036	075016	G	G#PRMA=	000001	IREST =	000004	LPNTS	075714		L#TIML	002014	G
ERR037	075030	G	G#PRMD=	000002	ISR =	000100	LPNTX	075704		L#UNIT	002012	G
ERR038	075044	G	G#PRML=	000000	ISTRT =	000010	LT1L1	103716		L10000	064372	
ERR23A	074536		G#RADA=	000140	ISTRTH=	000020	LT1L2	103744		L10001	064402	
ERR23B	074562		G#RADB=	000000	IXE =	004000	LT11	103730		L10002	074266	
ERR23C	074570		G#RADD=	000040	I#AU =	000041	L#ACP	002110	G	L10003	074304	
EVL =	000004	G	G#RADL=	000120	I#AUTO=	000041	L#APT	002036	G	L10004	074322	
E#END =	002100		G#RADO=	000020	I#CLN =	000041	L#AU	111336	G	L10005	074334	
E#LOAD=	000035		G#XFER=	000004	I#DU =	000041	L#AUT	002070	G	L10006	074346	
FDATA	064472		G#YES =	000010	I#HRD =	000041	L#AUTO	111300	G	L10007	074360	
FFREE	064412	G	HCOMM	075446	I#INIT=	000041	L#CCP	002106	G	L10010	074376	
FILOPN	064474		HC.BF1=	000100	I#MOD =	000041	L#CLEA	111302	G	L10011	074410	
FHEM	064416		HC.BF2=	000206	I#MSG =	000041	L#CO	002032	G	L10012	074426	
FHEMS	064420		HC.BSZ=	000106	I#PROT=	000040	L#DEPO	002011	G	L10013	074466	
FHERR	075410		HC.CCT=	000012	I#PTAB=	000041	L#DESC	064724	G	L10014	074514	
FNAME	064456		HC.CEV=	000014	I#PWR =	000041	L#DESP	002076	G	L10015	074600	
FNUM	064442		HC.CMD=	000010	I#RPT =	000041	L#DEVP	002060	G	L10016	074614	
FRMTT	065037		HC.CPK=	000020	I#SEG =	000041	L#DISP	064344	G	L10017	074632	
FS =	100000		HC.ESZ=	000004	I#SETU=	000041	L#DLY	002116	G	L10020	074652	
F#SIZE	064414	G	HC.INT=	000000	I#SFT =	000041	L#DTP	002040	G	L10021	074670	
F#WORD	106150		HC.ISZ=	000004	I#SRV =	000041	L#DTP	002034	G	L10022	074702	
F#AU =	000015		HC.MCT=	000006	I#SUB =	000041	L#DU	111330	G	L10023	074716	
F#AUTO=	000020		HC.MEV=	000014	I#TST =	000041	L#DUT	002072	G	L10024	074732	
F#BGN =	000040		HC.MPK=	000020	J#JMP =	000167	L#DVTY	064700	G	L10025	074744	
F#CLEA=	000007		HC.MSG=	000004	KTBASE	064434	L#EF	002052	G	L10026	074762	
F#DU =	000016		HC.PSZ=	000060	KTBASO	064436	L#ENVI	002044	G	L10027	074770	
F#END =	000041		HC.RSZ=	000004	KTMEM	064640	L#ERRT	064402	G	L10030	074776	
F#HARD=	000004		HC.SIZ=	000314	KW.BRL	064620	L#ETP	002102	G	L10031	075014	
F#HW =	000013		HELP =	000000	KW.CSR	064616	L#EXP1	002046	G	L10032	075026	
F#INIT=	000006		MM.CYL=	020000	KW.EL	064626	L#EXP4	002064	G	L10033	075042	
F#JMP =	000050		HOE =	100000	KW.HZ	064624	L#EXP5	002066	G	L10034	075054	
F#MOD =	000000		H.BRL =	000004	KW.OUT=	000105	L#HARD	113266	G	L10035	075406	
F#MSG =	000011		H.BST =	000006	KW.VEC	064622	L#HIME	002120	G	L10036	104516	

J6

L10037	104526	MSGPKT	075150	PAT16C	064554	P.SHST	000042	RSP.CK	105366
L10040	106250	MSGUBA	113352	PAT16W	064556	P.SHUN	000040	RSP.S1	105314
L10041	106256	MSGVEC	113400	PB	075556	P.STS	000012	RSP.S2	105322
L10042	107626	MX	000252	PF	075542	P.TIME	000024	RSP.S3	105342
L10044	111276	MXFERE	101142	PNT	001000 G	P.TRKS	000044	RSP.S4	105360
L10045	111300	MXFERP	065170	PNTERR	103250	P.UADR	000020	RUNDM	076532
L10046	111326	MXFERX	101140	PNTNUM	102442	P.UNFL	000016	RWORDT	106110
L10047	111334	M1	000004	PNTNUS	102450	P.UNIT	000004	RWRDE1	106216
L10050	111342	M2	000010	PNTPKL	075106	P.UNSZ	000044	SAMVEC	111262
L10051	112416	M3	000100	PNTPKT	075056	P.UNTI	000024	SA.A2	001000
L10052	111522	M4	000200	PRI	002000 G	P.USEF	000022	SA.BST	000374
L10053	111564	NCON	075374	PRINTC	075506	P.VRSN	000014	SA.CMD	034000
L10054	111720	NCONF	076024	PRI00	000000 G	P.VSER	000050	SA.CME	000070
L10055	112320	NCJNS	076002	PRI01	000040 G	RDDAT	106004	SA.CM1	001000
L10056	112332	NOCLOC	066466	PRI02	000100 G	RDDL	105570	SA.CNT	000360
L10057	112346	NXMAD	064636	PRI03	000140 G	RDERR	106172	SA.CTP	003400
L10060	112410	NXMI	104510 G	PRI04	000200 G	RDREC	105630	SA.DI	000400
L10061	112514	ONEFIL	000001	PRI05	000240 G	RDEST	105672	SA.ERC	003777
L10062	112552	OP.ABO	000001	PRI06	000300 G	RDSTS	105670	SA.ERR	100000
L10063	113262	OP.ACC	000020	PRI07	000340 G	RESET	106260	SA.GO	000001
L10064	113352	OP.AVA	000100	PS	075636	RESPCT	076640	SA.INE	000200
L10065	113606	OP.AVL	000010	PTYPE	075770	RESPDM	076626	SA.INT	000200
L10066	114316	OP.CCD	000021	PX	075612	RG.FLG	040000	SA.LFC	000002
L10070	114332	OP.CMP	000040	P.BCNT	000014	RG.OWN	100000	SA.MCV	000017
MC	000314	OP.DUP	000101	P.BUFF	000020	RNTIM	065045	SA.MSE	000007
MD	000125	OP.ELP	000003	P.CHST	000020	RNTIME	106362	SA.MSG	003400
MD.CMP	040000	OP.END	000200	P.CMCL	000022	RNTIMX	106542	SA.MS1	000400
MD.CMB	000010	OP.ERS	000022	P.CNTF	000016	RNTIM1	065070	SA.NV	002000
MD.ERR	010000	OP.ESP	000002	P.CNTI	000024	RNTIM2	065076	SA.NVE	000400
MD.EXP	100000	OP.FLU	000023	P.CPSP	000042	RNT4DM	076436	SA.PRG	000001
MD.FEU	000001	OP.GCS	000002	P.CRF	000000	RPTCT	106764	SA.STE	000200
MD.IMF	00000?	OP.GSS	000001	P.CTMO	000020	RPTCTN	107232	SA.STP	100000
MD.NXU	000001	OP.GUS	000003	P.CYLS	000050	RPTDT	107004	SA.S1	004000
MD.PRI	000001	OP.MRD	000030	P.DEXT	000014	RPTDTN	107226	SA.S2	010000
MD.RIP	000001	OP.MWR	000031	P.DFLG	000017	RPTMD2	107603	SA.S3	020000
MD.SCH	001000	OP.ONL	000011	P.DMDT	000024	RPTMSD	107534	SA.S4	040000
MD.SCL	002000	OP.RD	000041	P.DPRG	000020	RPTMSG	107262	SA.TST	100000
MD.SEC	000100	OP.RLC	000103	P.DTMO	000024	RPTMSH	107316	SA.VCE	000177
MD.SEQ	000020	OP.RPL	000024	P.ELGF	000034	RPTXX	107242	SA.VEC	000177
MD.SER	000400	OP.RSD	000005	F.FBBK	000034	RSPDRP	077036	SA.WRP	040000
MD.SPD	000001	OP.SCC	000004	P.FLGS	000011	RSPDSP	077360	SEKERE	101000
MD.SSH	000200	OP.SEX	000007	P.GRPS	000046	RSPERR	077122	SET00	104542
MD.SWP	000004	OP.SHC	000102	P.HSTI	000020	RSPIN	077054	SET01	104550
MD.VOL	000002	OP.SSD	000004	P.HTMO	000020	RSPNWR	077074	SET02	104556
MD.WBN	000100	OP.SUC	000012	P.LBN	000034	RSPNRP	077022	SETTO	104530
MD.WBV	000400	OP.WR	000042	P.MEDI	000034	RSPNT0	076766	SFPTBL	064374 G
MEMFIL	077426	OSTRE	076040	P.MLUN	000014	RSPNXT	076770	SM.IW	040000 G
MESSAG	101610	OSTRNG	075772	P.MOD	000012	RSPQU	077142	SM.LOG	001000 G
MESSG	065713	O\$APTC	000000	P.OPCD	000010	RSPQU2	077256	SM.MAN	000200 G
MLDRER	111226	O\$AU	000000	P.OTRF	000014	RSPQU3	077324	SM.SSF	000400 G
MM	000377	O\$BGNR	000001	F.OVRL	000034	RSPPT0	077332	SNOCHD	104210
MSCP	000000	O\$BGNS	000001	P.RBN	000014	RSPPT1	077134	SNO.S1	105300
MSGBRL	113407	O\$DU	000000	P.RBNS	000056	RSPPT2	077144	SNO.S2	105304
MSGBST	113420	O\$ERRT	000001	P.RCTC	000057	RSPPT3	077214	SNO.S3	105310
MSGCST	113452	O\$GNSW	000001	P.RCTS	000054	RSPRPT	077020	SO.BIT	000004
MSGLDR	113442	O\$POIN	000001	P.RGID	000034	RSPTM	076724	SO.EL	000000
MSGPKL	075174	O\$SETU	000001	P.RGIF	000040	RSPTM0	076754	SO.XL	000002

SSTEP4 105406	T\$COJE= 002130	T1GOOD 111522	T4BB2 100704	UTOT2 101214
STIME 064632	T\$ERRN= 000035	T1MSIZ 077416	T4BB2E 100732	UTOT3 101236
STLDDM 076542	T\$EXCP= 000000	T1NEXT 111362	T4CON 112674	UTOT4 101244
STORAG 002122	T\$FLAG= 000040	T1NOW 111422	T4DEF 113110	U52EXT 064526
STOSIZ= 062220	T\$FREE= 114332	T1SKIP 111402	T4DEFA 113114	WAITMS 104320
ST.ABO= 000002	T\$GMAN= 000000	T1.1 111426	T4DEFB 113126	WCHNG 106554
ST.AOL= 000400	T\$HILI= 177777	T1.2 111524	T4DEFC 113152	WCHNGD 106650
ST.AVL= 000004	T\$LAST= 000001	T1.3 111566	T4DEFD 113160	XFRU 074215
ST.CMD= 000001	T\$LOLI= 000000	T1.4 111722	T4DEFE 113164	XMSG1 073543
ST.CMP= 000007	T\$LSYM= 010000	T1.5 112322	T4DEFW 113100	XMSG2 073577
ST.CNT= 000012	T\$LTNO= 000004	T1.6 112334	T4EXIT 113254	XPKT1 073644
ST.DAT= 000010	T\$NEST= 177777	T1.7 112350	T4MPRM 100352	XPKT2 074135
ST.DIA= 000037	T\$NSO = 000000	T2 112420 G	T4MXFR 101002	XSA 074164
ST.DRV= 000013	T\$NS1 = 000005	T2CMD 077702	T4OPT7 064760	X\$ALWA= 000000
ST.HST= 000011	T\$NS2 = 000002	T2CMDE 100340	T4PRM1 113004	X\$FALS= 000040
ST.MFE= 000005	T\$PCNT= 000000	T2CMDM 077716	T4PRM2 113016	X\$OFFS= 000400
ST.MSK= 00003	T\$PTAB= 010067	T2CMDN 100262	T4PRM3 113052	X\$TRUE= 000020
ST.OFL= 000003	T\$PTHV= 000001	T2CMDQ 100074	T4PRM4 113056	X1 067160
ST.SUB= 000040	T\$PTNU= 000001	T2CMDR 100236	T4RUN 113174	X1A 067061
ST.SUC= 000000	T\$SAVL= 177777	T2CMDV 100162	T4SEEK 100762	X14 070061
ST.WPR= 000006	T\$SEGL= 177777	T2CMDW 100312	T4SOFT 100734	X2 067254
SVCGBL= 000000	T\$SIZE= 000010	T2CMDX 100334	T4STRT 112712	X2A 067061
SVCINS= 000000	T\$SUAN= 000000	T2CMD0 077736	T4UPRM 100374	X21 070650
SVCSUB= 000000	T\$TAGL= 177777	T2CMD2 100010	T4UPRX 100652	X22 070765
SVCTAG= 000000	T\$TAGN= 010071	T2CMD3 100172	T4WAIT 113240	X23A 071123
SVCTST= 000000	T\$TEMP= 000000	T2CMD9 077722	T4WARN 065605	X23B 071431
S\$LSYM= 010000	T\$TEST= 000004	T2CMS1 066056	UAM = 000200 G	X24 071445
S.EL 113756	T\$TSTM= 177777	T2CMS5 066445	UCNT 064452	X25 071635
S.IW 114112	T\$TSTS= 000001	T2DLL 077536	UDAINT 104612	X26 072006
S.LOG 114144	T\$AU = 010050	T2DR 064646	UDAIST 105070	X27 072174
S.MAN 113606	T\$AUT= 010045	T2GND1 103142	UDARSD 105546	X28 072321
S.MES 113673	T\$CLE= 010046	T2GND2 103144	UDARSP 105410	X29 072371
S.SSF 114054	T\$DAT= 010070	T2GND3 103204	UDASRV 104520 G	X3 067323
S.XL 113772	T\$DU = 010047	T2GNE 103240	UF.CMR= 000001	X3A 067061
TEMP 064476	T\$HAR= 010064	T2GNUM 103122	UF.CMW= 000002	X30 072662
TIEXIT 076422	T\$HW = 010000	T2GNX 103234	UF.INA= 040000	X31 072775
TINDEX= 000006	T\$INI= 010044	T2PNT 103050	UF.RPL= 100000	X32 073117
TINIT 076336	T\$MSG= 010035	T2PNTB 103040	UF.SCH= 004000	X35 073230
TINITE 076424	T\$PC = 000001	T2PNTD 103112	UF.SCL= 002000	X36 073310
TNAMES 076274	T\$PRO= 010043	T2PNT0 103070	UF.WBN= 000100	X37 073426
TNUM 064444	T\$PTA= 010067	T2PNTW 103010	UF.WPH= 020000	X38 070460
TOOMER 111244	T\$RPT= 010042	T2WARN 065757	UF.WPS= 001000	X4 067406
TSTTAB 064430	T\$SDF= 010065	T2WRO 064644	UF.576= 000004	X5 067725
TTYOUT 075662	T\$SRV= 010041	T2WRR 064642	URNING 064450	X6 067573
TYPCNT 064532	T\$SUB= 010060	T3 112516 G	URUN 064446	X7 067775
TY.U50= 000002	T\$SW = 010001	T4 112554 G	UTOTST 101144	X8 067660
TY.U52= 000001	T\$TES= 010063	T4BB1 100654	UTOT1 101156	X8A 067061
T\$ARGC= 000004	T1 111344 G	T4BB1E 100702	UTOT1A 101210	\$PATCH 114166 G

. ABS. 114332 000 (RW,I,GBL,ABS,OVR)  
 000000 001 (RW,I,LCL,REL,CON)  
 Errors detected: 0

\*\*\* Assembler statistics

L6

PARAMETER CODING MACRO V05.00 Wednesday 04-Jan-84 16:12 Page 221-5  
Symbol table

SFQ 0283

Work file reads: 396  
Work file writes: 367  
Size of work file: 29429 Words ( 115 Pages)  
Size of core pool: 17152 Words ( 67 Pages)  
Operating system: RT-11 (Under RSTS/E)

Elapsed time: 00:04:35.08  
ZUDCEO,ZUDCEO/C=SVC34R.MLB/P:1,ZUDCEO.DOC,ZUDCEO





C:GPMN	83-3740	190-26	193-7																			
C:GPLO	83-3740																					
C:GPRI	83-3740																					
C:INIT	83-3740	198-43																				
C:INLP	83-3740																					
C:MANI	83-3740	130-34	196-25	214-17																		
C:MEM	83-3740	191-30																				
C:MSG	83-3740	106-28	106-32	106-36	106-40	106-45	106-49	106-54	106-58	106-62	106-64	106-69	106-103	106-107								
		106-111	106-116	106-120	106-124	106-128	106-133	106-137	106-142	106-146	106-150	106-155	106-159	106-163	106-167							
		106-214																				
C:OPEN	83-3740	180-25																				
C:PNTB	83-3740	110-20																				
C:PNTF	83-3740	110-18	196-15																			
C:PNTS	83-3740	110-24	196-119	186-121																		
C:PNTX	83-3740	110-22																				
C:QIO	83-3740																					
C:ROBU	83-3740																					
C:REFG	83-3740	189-3	189-9	189-15	189-22																	
C:RESE	83-3740	83-3740																				
C:REVI	83-3740	83-427																				
C:RFLA	83-3740	138-50	140-38	142-45																		
C:RPT	83-3740	186-200																				
C:SEFG	83-3740																					
C:SPHI	83-3740	197-4	207-22	207-49	207-60	207-69																
C:SVEC	83-3740	162-36	175-28	182-15	191-20	204-5	207-21															
C:TPRI	83-3740																					
C:BST	97-300	174-100	193-24																			
C:DR0	97-350	139-20	154-16	186-84	190-7	195-5	196-9	214-35	216-7													
C:DR1	97-360																					
C:DR2	97-370																					
C:DR3	97-380																					
C:DR4	97-390																					
C:DR5	97-400																					
C:DR6	97-410																					
C:DR7	97-420																					
C:FLG	97-330	120-27*	121-10	121-13	121-15	122-23*	123-7	123-27	124-30*	124-31	124-38*	124-41*	138-54*	140-60*								
	143-19*	162-86*	162-87*	164-28*	167-21*	169-18	169-46*	194-28														
C:YCOM	97-340	106-184	109-18*	121-9	166-17	167-15																
C:JAD	97-320																					
C:JSR	97-310	162-35																				
C:REF	97-450	123-20	167-16*	167-17																		
C:SIZE	97-470	120-36	122-16	186-148	190-14	190-17	190-32	193-34	194-11	194-28	196-18	203-67	211-19	214-50								
	216-19																					
C:TO	97-430	121-44	124-46	169-15	169-39	176-23	176-35	184-8	184-19	207-33	207-45											
C:TOH	97-440	121-41	169-36	176-32	184-16	207-42																
C:UADR	97-270	106-201	175-25	193-14	206-8	207-8	207-19															
C:UNIT	97-280	120-29	120-30	121-12	161-18	162-56*	186-74	190-5*	190-11*	203-63	203-64											
C:VEC	97-290	162-32	193-22	193-28	207-9	207-76																
CALR1	115-4	145-70	147-12																			
CALR2	115-5	146-60																				
CALR3	115-6	147-110																				
CALR4	115-7	148-70																				
CALR5	115-8	149-70																				
CALR6	115-9	150-60																				
CALR7	115-10	151-60																				
CALR8	115-11	152-60																				



















PRE-PROGRAMMED SUBRO....B1	TEST 1: UNIBUS ADDRE....B5
PRE-PROGRAMMED SUBRO....C1	TEST 1: UNIBUS ADDRE....C5
PRE-PROGRAMMED SUBRO....D1	TEST 1: UNIBUS ADDRE....D5
PRE-PROGRAMMED SUBRO....E1	TEST 1: UNIBUS ADDRE....E5
PRE-PROGRAMMED SUBRO....F1	TEST 1: UNIBUS ADDRE....F5
PRE-PROGRAMMED SUBRO....G1	TEST 1: UNIBUS ADDRE....G5
PRE-PROGRAMMED SUBRO....H1	TEST 1: UNIBUS ADDRE....H5
PRE-PROGRAMMED SUBRO....I1	TEST 1: UNIBUS ADDRE....I5
PRE-PROGRAMMED SUBRO....J1	TEST 2: DISK RESIDEN....J5
PRE-PROGRAMMED SUBRO....K1	TEST 3: DISK FUNCTIO....K5
PRE-PROGRAMMED SUBRO....L1	TEST 4: DISK EXERCIS....L5
PRE-PROGRAMMED SUBRO....M1	TEST 4: DISK EXERCIS....M5
PRE-PROGRAMMED SUBRO....N1	TEST 4: DISK EXERCIS....N5

PRE-PROGRAMMED SUBRO....R2	TEST 4: DISK EXERCIS....B6
PRE-PROGRAMMED SUBRO....C2	TEST 4: DISK EXERCIS....C6
PRE-PROGRAMMED SUBRO....D2	HARDWARE PARAMETER C....D6
PRE-PROGRAMMED SUBRO....E2	SOFTWARE PARAMETER C....E6
PRE-PROGRAMMED SUBRO....F2	SOFTWARE PARAMETER C....F6
PRE-PROGRAMMED SUBRO....G2	SOFTWARE PARAMETER C....G6
PRE-PROGRAMMED SUBRO....M2	SYMBOL TABLE ....H6
PRE-PROGRAMMED SUBRO....T2	SYMBOL TABLE ....I6
PRE-PROGRAMMED SUBRO....J2	SYMBOL TABLE ....J6
PRE-PROGRAMMED SUBRO....K2	SYMBOL TABLE ....K6
PRE-PROGRAMMED SUBRO....L2	SYMBOL TABLE ....L6
PRE-PROGRAMMED SUBRO....M2	CROSS REFERENCE TABL....M6
PRE-PROGRAMMED SUBRO....N2	CROSS REFERENCE TABL....N6

PRE-PROGRAMMED SUBRO....B3	CROSS REFERENCE TABL....B7
PRE-PROGRAMMED SUBRO....C3	CROSS REFERENCE TABL....C7
PRE-PROGRAMMED SUBRO....D3	CROSS REFERENCE TABL....D7
PRE-PROGRAMMED SUBRO....E3	CROSS REFERENCE TABL....E7
PRE-PROGRAMMED SUBRO....F3	CROSS REFERENCE TABL....F7
PRE-PROGRAMMED SUBRO....G3	CROSS REFERENCE TABL....G7
REPORT CODING SECTIO....H3	CROSS REFERENCE TABL....H7
REPORT CODING SECTIO....I3	CROSS REFERENCE TABL....I7
REPORT CODING SECTIO....J3	CROSS REFERENCE TABL....J7
REPORT CODING SECTIO....K3	CROSS REFERENCE TABL....K7
INITIALIZE SECTION ....L3	CROSS REFERENCE TABL....L7
INITIALIZE SECTION ....M3	CROSS REFERENCE TABL....M7
INITIALIZE SECTION ....N3	CROSS REFERENCE TABL....N7

INITIALIZE SECTION ....B4	CROSS REFERENCE TABL....B8
INITIALIZE SECTION ....C4	CROSS REFERENCE TABL....C8
INITIALIZE SECTION ....D4	CROSS REFERENCE TABL....D8
INITIALIZE SECTION ....E4	CROSS REFERENCE TABL....E8
INITIALIZE SECTION ....F4	CROSS REFERENCE TABL....F8
INITIALIZE SECTION ....G4	CROSS REFERENCE TABL....G8
INITIALIZE SECTION ....H4	
INITIALIZE ERRORS ....I4	
AUTODROP SECTION ....J4	
CLEANUP CODING SECTI....K4	
DROP UNIT SECTION ....L4	
ADD UNIT SECTION ....M4	
HARDWARE TESTS ....N4	