Micro Fiche Scan

Name of device(s) tested:

    UDA50,KDA50,RA60/80/81/82

Test description:

    UDA/KDA SUBSYS EXER

MAINDEC Number or Package Identifier (after SEP 1977):

    CZUDJA0

Fiche Document Part Number:

    AH-T937A-MC

Fiche preparation date unknown, using copyright year:

    1985

Image resolution:

    8-bit gray levels, max. quality for archiving

IDENTIFICATION

PRODUCT CODE: AC-T936A-MC

PRODUCT NAME: CZUDJAO UDA/KDA SUBSYS EXER

PRODUCT DATE: 2-OCT-1985

MAINTAINER: RON BOWSER

AUTHOR: JOHN MERTZ

1
2
3

.REM @

CONTENTS

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 2

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 4

1 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

The PDP-11 UDA50-A/KDA50-Q SUBSYSTEM EXERCISER is a standalone diagnostic written in MACRO-11 that runs under the control of version 2.1 of the PDP-11 Diagnostic Supervisor (XXDP+). This diagnostic exercises UDA50-A or KDA50-Q disk subsystems and verifies that the subsystems function properly. Though it will provide the operator all relevant information returned in the MSCP (Mass Storage Control Protocol) response packet when a failure occurs, the operator may be required to use other diagnostics to isolate the FRU (Field Replaceable Unit) that failed.

The UDA50-A/KDA50-Q SUBSYSTEM EXERCISER uses the MSCP interface to the KDA50-Q or UDA50-A disk controller to perform extensive input/output operations on all selected SDI (Standard Disk Interface) compatible disk drives and selected controllers.

The UDA50-A/KDA50-Q SUBSYSTEM EXERCISER consists of three tests: a controller verification test (TEST 1), a subsystem verification test (TEST 2), and a subsystem exerciser (TEST 3). TEST 1 will support at least two controllers. TESTS 2 and 3 are multi-drive tests that will support at least two controllers with a maximum of four drives on each controller. The only test parameters applicable to TEST 1 and TEST 2 are the hardware test parameters. Both the software and hardware test parameters are applicable to TEST 3. The default operation of this diagnostic program is to run TEST 1, then run TEST 2, and then run TEST 3. Since this diagnostic can only access the customer data area on the drives under test, any operations requiring a write to the drive should only be used if the customer data has been adequately backed up or, on drives with removable media, a scratch pack is put in the drive.

TEST 1 (the controller verification test) tests each of the controllers' ability to initialize correctly and pass their self tests. The memory in the controller and the data path is verified by performing extensive maintenance read and write operations to each controller. The test will also verify the ability of the host to set each controller's characteristics using MSCP. This test does not perform any operations that use the disk drives.

TEST 2 (the subsystem verification test) initializes all of the controllers and drives under test. It verifies the host's ability to use MSCP to set the drives ONLINE, and AVAILABLE. It also verifies the subsystem's ability to perform read, seek, and data compare operations. On any drive that is writeable, this test verifies the

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 5

drive's ability to write and performs data pattern verification on the
data written.

TEST 3 (the subsystem exerciser) initializes the controllers and
brings the drives ONLINE. The exerciser consists of two phases, a
deterministic phase and an operator modifiable phase. The
deterministic phase exercises the subsystem in a manner fixed in the
diagnostic program. The operator modifiable phase exercises the
subsystem in a manner determined by the operator. Each phase will
perform extensive simultaneous I/O operations to all of the drives
under test. The amount of data transferred and the number of
outstanding I/O operations will be limited by the amount of available
memeory on the system.

This program is designed to handle all future disk drives that are
attached to the UDA50-A or KDA50-Q without modifying or rereleasing
the program. 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.

The following PDP-11 diagnostic programs are also provided for the
UDA50-A or KDA50-Q disk subsystem:

CZUDH - UDA50-A/KDA50-Q Basic Subsystem Diagnostic.

CZUDI - UDA50-A/KDA50-Q Disk Drive Exerciser.

CZUDK - UDA50-A/KDA50-Q Disk Drive Formatter.

CZUDL - UDA50-A/KDA50-Q Bad Block Replacement Utility

CXDUB - UDA50-A/KDA50-Q DEC/X11 Module.

This diagnostic has been written for use with the Diagnostic Runtime
Services software (DRS supervisor). DRS provides the interface to the
operator and to the software environment. For a complete description
of the Diagnostic Runtime Services, refer to the XXDP+ User's Manual.
There is a brief description of the Diagnostic Runtime Services in the
OPERATING INSTRUCTIONS section of this document.

1.2 SYSTEM REQUIREMENTS

This program was designed using the XXDP version 2.1. The program
requires version 2.1 or later of XXDP. Run time environments are

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 6

     determined by the Diagnostic Runtime Services and may change as new
versions of DRS are developed. This program requires the following:

     PDP-11 processor
     28K words of memory (minimum)
     Console terminal
     XXDP+ load media containing this program
     One or more UDA50-A or KDA50-Q subsystems. The subsystem controller
         must be at the latest hardware and microcode revision level.
     Line clock - either Type L or P

The line clock is used for all timed loops in the program. If a clock
exists on the system it should be enabled so it can interrupt the
processor. The diagnostic will run on a system with no clock or with
the clock disabled, 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).

The XXDP+ load device does not need to remain on-line during the
execution of this diagnostic.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 7

2 PRODUCT GOALS

2.1 PERFORMANCE GOALS

The UDA50-A/KDA50-Q SUBSYSTEM EXERCISER will exercise any SDI
compatible drive. The exerciser will build all MSCP packets needed to
perform I/O operations to the drives under test. All error recovery
and retries are performed by the controller automatically. ECC
correction and retries can be disabled by the operator. All error
reporting is based on information provided by the MSCP response
packet. TEST 3 will allow the operator to change the parameters of
the exercise, thereby isolating a possible problem to a specific
region of a drive's media. It is the goal of the subsystem exerciser
to generate as heavy an I/O load on the controller(s) and drive(s) as
possible.

2.2 FAILSOFT GOALS

When a read, write, or access operation fails, the exerciser will
display the following information: the elapsed time to failure, the
controller and drive that the failure occurred on, the LBN of the
failing sector, and the reason for the failure.

When the hard error limit for a drive has been reached, the drive is
dropped from testing and a message will be printed notifying the
operator that the drive has been dropped. If a device fatal error
occurs, the drive is immediately dropped. The error limit has no
effect on device fatal errors.

Unexpected traps will require the operator to restart the diagnostic.
The diagnostic will restart automatically after a power failure as
long as the contents of memory or the hardware configuration are
unchanged. Otherwise, the operator must re-boot the system.

2.3 RESTRICTIONS

The UDA50-A/KDA50-Q SUBSYSTEM EXERCISER will test at least two
controllers at a time, with a maximum of four drives per controller.
The number of sectors that can be transferred at a time and the number
of drives and controlelrs that can be simultaneously tested will be
limited by the available memory in the system.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 8

2.4 NON-GOALS

This diagnostic will not replace bad blocks.

This diagnostic will not isolate failing FRUs.

This diagnostic is not intended to be perform disk media verification.

3 PROGRAM DESCRIPTION

The UDA50-A/KDA50-Q SUBSYSTEM EXERCISER consists of three tests: the
controller verification test (TEST 1), the subsystem verification test
(TEST 2), and the subsystem exerciser (TEST 3). TEST 1 will support
at least two controllers. TESTS 2 and 3 are multi-drive tests that
will support at least two controllers with a maximum of four drives on
each controller. TEST 3 will exercise all of the drives under test in
parallel unless restricted by memory. In all of the tests, the size
of the data transfers, the number of devices simultaneously tested,
and the number of outstanding MSCP commands will be limited by the
amount of host memory available. The default operation of this
diagnostic program is to run TEST 1, then run TEST 2, and then run
TEST 3. All hardware, software, and manual intervention questions are
asked prior to running the tests. The diagnostic fully supports all
of the diagnostic supervisor (DRS) commands, switches, and flags
(except LOE) which are listed in Appendix A of this document. Since
this diagnostic can only access the customer data area on the drives
under test, any operations requiring a write to the drive should only
be used if the customer data has been adequately backed up or, on
drives with removable media, a scratch pack is put in the drive.

To run the complete exerciser, the operator loads XXDP+, then types:

    R ZUDJAO
      or
    L ZUDJAO
    S

and then at the diagnostic supervisor prompt (DR>), types:

    START

If any of the drives are designated as writeable, the following
warning message will be printed when the exerciser is started:

    CUSTOMER DATA WILL BE DESTROYED ON:
        UNIT   CONTROLLER   DRIVE
        xx   xxxxxx   xxx

Unless this diagnostic program is running in unattended mode, the
warning message will be followed with the following question.

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

An answer of 'N', the default, will halt the diagnostic and allow the

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 10

operator to change the hardware and software parameters. An answer of
'Y' will allow the diagnostic program to continue.

3.1 CONTROLLER VERIFICATION TEST (TEST 1)

TEST 1 is designed to verify that the controllers to be exercised are
functional. This test does not perform any operations that use the
disk drives. If any errors occur, the controller on which the error
occurred and all associated drives will be dropped from further
testing. Only the hardware test parameter questions are applicable to
this test.

First, TEST 1 initializes all of the controllers to be tested and
causes all internal controller diagnostics to be run. During STEP 1
of the initialization, the wrap mode bit will be set in the SA
register and wrap mode testing will be performed. The wrap mode
testing will use a shifting inversions data pattern. At the end of
the wrap mode testing, the controllers will be re-initialized. In
STEP 3 of the initialization, both the PURGE/POLL test and the HOST
MEMORY test will be performed by the controllers. In STEP 4 of the
initialization, the LAST FAIL bit will be cleared and the GO bit will
be set in the SA register. The model number and microcode version
number will be obtained from the SA register in STEP 4.

After initialization, TEST 1 performs extensive maintenance write and
read operations to the controller to verify the integrity of the
controller's memory and the data path to the controller.   The test
will write the entire controller memory 34 times using a shifting
inversions data pattern then read back the contents of memory and
perform a data compare on the data read. The test will transfer 512
bytes of data in each write or read operation.

Finally, TEST 1 will re-initialize the controllers and issue a MSCP
SET CONTROLLER CHARACTERISTICS command to each of them. This command
will enable "attention messages", "miscellaneous error log messages",
and "this host's error log messages". The host access timeout will be
set to 0. The controller identifier, hardware revision number, and
software revision number will be obtained from the end packet.

To run only TEST 1 of the diagnostic program, the operator loads XXDP+
and runs ZUDJAO, as described in the PROGRAM DESCRIPTION section. At
the diagnostic supervisor prompt (DR>), the operator types:

STA/TES:1

This will cause the diagnostic program to ask the hardware and
software questions. Once the questions have been asked, TEST 1 is
run.

3.2 SUBSYSTEM VERIFICATION TEST (TEST 2)

TEST 2 is designed to verify that the disk subsystems to be exercised
are functional. If any device fatal errors occur, the drive on which
the error occurred will be dropped from further testing. Controller
errors will result in the controller and associated drives being
dropped from testing. Only the hardware test parameter questions are
applicable to this test.

First, TEST 2 initializes the controllers without performing the wrap
mode test. After initialization, a MSCP SET CONTROLLER
CHARACTERISTICS command is issued to each of the controllers. This
command will enable "attention messages", "miscellaneous error log
messages", and "this host's error log messages". The host access
timeout will be set to 0. The controller identifier, hardware
revision number, and software revision number will be obtained from
the end packet.

TEST 2 then sets all of the drives under test online with the MSCP
ONLINE command with the ENABLE SET WRITE PROTECT modifier. For each
drive a MSCP SET UNIT CHARACTERISTICS command is issued to suppress
caching. Drives that are not to be written are also software write
protected.

TEST 2 will then test the data path to and from the drive. The test
will first use a MSCP READ with DATA COMPARE command to read the first
logical block on each disk. It will then issue a MSCP ACCESS command
to access 20 blocks on each disk. While the ACCESS command is
outstanding, a MSCP GETCOMMAND STATUS command will be issued to check
the status of the ACCESS command.

On any drive that is writeable, a MSCP ERASE command is issued for the
first LBN on each disk. When this completes, MSCP WRITE with DATA
COMPARE commands are used to write one block of each of the 16
standard data patterns to each drive under test. After performing the
write operations, TEST 2 will perform a MSCP COMPARE HOST DATA command
on the last block written.

Once the I/O operations have completed, this test will issue the MSCP
AVAILABLE command for each drive. When the test completes, all drives
will be available, but not spun down.

To run only TEST 2 of the diagnostic program, the operator loads XXDP+
and runs ZUDJAO, as described in the PROGRAM DESCRIPTION section. At
the diagnostic supervisor prompt (DR>), the operator types:

STA/TES:2

This will cause the diagnostic program to ask the hardware and
software questions.   Once the questions have been asked, TEST 2 is
run.

3.3 SUBSYSTEM EXERCISER (TEST 3)

TEST 3 is designed to extensively exercise the disk subsystem. To the
extent possible, TEST 3 will perform input and output operations to
all of the drives under test simultaneously. This test will always
attempt to maintain as high an input/output load on the drives and
controllers as possible. If any errors occur, the action of this test
will be determined by the setting of the software test parameters.
All hardware, software, and manual intervention questions apply to
this test.

First, TEST 3 initializes the controllers without performing the wrap
mode test.   This test will use command and response rings with 4
entries each.   After initialization, a MSCP SET CONTROLLER
CHARACTERISTICS command is issued to each of the controllers. This
command will enable "attention messages", "miscellaneous error log
messages", and "this host's error log messages". The host access
timeout will be set to 0.   The controller identifier, hardware
revision number, and software revision number will be obtained from
the end packet.

TEST 3 then sets all of the drives under test online with the MSCP
ONLINE command with the ENABLE SET WRITE PROTECT modifier. The unit
identifier, media type, and unit size will be obtained for each drive
from the end message. For each drive a MSCP SET UNIT CHARACTERISTICS
command is issued. Drives that are not to be written are also
software write protected.

TEST 3 then enters the exerciser section of the test which consists of
two phases, a deterministic phase and an operator modifiable phase.
The deterministic phase exercises the subsystem in a manner fixed in
the diagnostic program. The operator modifiable phase exercises the
subsystem in a manner determined by the operator.

On drives that are writeable, the deterministic phase will

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 13

sequentially write 128 block segments containing each of the 16 standard data patterns to the drive.

On all drives, the deterministic phase will then sequentially read the first 512 blocks of data from the disk. Data compare operations will randomly be performed on the data read. The MSCP ACCESS command will be used to read the remaining blocks of data on the disk.

The operator configurable phase of TEST 3 consists of alternating read and (if enabled) write operations until the test limits are reached. Controller data compare operations are randomly performed on read and write operations. The characteristics of the read and write operations, and the test limits can be modified by the operator by using the software parameters and manual intervention questions. TEST 3 performs the operator configurable phase until the maximum run time expires, the error limit is reached on all selected drives, or the operator types CONTROL-C. When the exerciser completes, a summary message will be printed indicating any errors that occurred.

To run only TEST 3 of the diagnostic program, the operator loads XXDP+ and runs ZUDJAO, as described in the PROGRAM DESCRIPTION section. At the diagnostic supervisor prompt (DR>), the operator types:

STA/TES:3

This will cause the diagnostic program to ask the hardware, software, and, if selected, the manual intervention questions. Once the questions have been asked, TEST 3 is run.

4 OPERATING INSTRUCTIONS

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

4.1 COMMANDS

This section lists the Diagnostic Runtime Services 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 a test that was interrupted (after †C)
PROCEED	Continue from an error halt
EXIT	Return to XXDP+ Monitor
REDIRECT	Redirect program output to a disk file or line printer.
TIME	Set the time of day or display current time
ADD	Activate a unit for testing (all units are considered to be active at start time)
DROP	Deactivate a unit
PRINT	Print statistical information
DISPLAY	Type a list of all device information
FLAGS	Type the state of all flags
ZFLAGS	Clear all flags

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

4.2 SWITCHES

There are several switches which can be used to modify supervisor
operation.   These switches are appended to the DRS commands. All of
the available switches are listed 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

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 15

                        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.
/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).
/DEV:DEVICE   Specify the device to which program output
                        will be redirected. Output will be placed
                        in a file called COLECT.DAT.
/LPT   Redirect program output to the lineprinter.

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	DEV	LPT
START	X	X	X	X	X		
RESTART	X	X	X	X	X		
CONTINUE		X	X	X			
PROCEED		X					
DROP					X		
ADD					X		
REDIRECT						X	X
TIME							
PRINT							
DISPLAY					X		
FLAGS							
ZFLAGS							
EXIT							

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 16

4.3 FLAGS

Flags are used to set up certain operational parameters such as halting on error. Flags are 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 (causes program to loop on test)
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
EVL	Execute diagnostic evaluation code

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

4.4 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 are testing multiple drives on the same controller, this becomes tedious since most of the answers are repetitious.

To illustrate a more efficient method, suppose you are testing a UDA50-A with 4 disk drives attached to it. These units are numbered 0 through 3. There is one hardware parameter that can vary among units, the drive number. This drive number may be 0 through 254. Below is a simple way to build a table for one UDA50-A with four units.

```
# UNITS (D) ?  4<CR>

UNIT 1
CSR ADDRESS OF CONTROLLER (0) 172150 ?  <CR>
DRIVE # (D) 0 ?  <CR>
WRITE ON CUSTOMER DATA AREA (L) N ? Y<CR>

UNIT 2
CSR ADDRESS OF CONTROLLER (0) 172150 ?  <CR>
DRIVE # (D) 0 ?  1<CR>
WRITE ON CUSTOMER DATA AREA (L) Y ? Y<CR>

UNIT 3
CSR ADDRESS OF CONTROLLER (0) 172150 ?  <CR>
DRIVE # (D) 1 ?  2<CR>
WRITE ON CUSTOMER DATA AREA (L) Y ? Y<CR>

UNIT 4
CSR ADDRESS OF CONTROLLER (0) 172150 ?  <CR>
DRIVE # (D) 2 ?  3<CR>
WRITE ON CUSTOMER DATA AREA (L) Y ? Y<CR>
```

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. Also, notice that the default values change when a non-default response is given.

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

Example 1:

```
# UNITS (D) ?  4<CR>

UNIT 1
CSR ADDRESS (0) 172150 ?  <CR>
DRIVE # (D) 0 ?  0-3<CR>
WRITE ON CUSTOMER DATA AREA (L) N ? Y<CR>
```

Example 2:

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 18

```
# UNITS (D) ?  4<CR>

UNIT 1
CSR ADDRESS (O) 172150 ?  <CR>
DRIVE # (D) 0 ?  0,1,2,3<CR>
WRITE ON CUSTOMER DATA AREA (L) N ? Y<CR>
```

As you can see in the above dialogue, the supervisor will build as
many entries as it can with the information given in any one pass
through the questions. In each example, four entries are built since
four drive numbers were specified. The supervisor assumes that the
CSR address is 172150 for each entry since it was specified only once.
In the first example, the "-" construct tells the supervisor to
increment the data from the first number to the second. In this case,
drive numbers 0, 1, 2, and 3 were specified.

4.5 TEST QUESTIONS

The following questions are asked by this diagnostic program in order
to set up the parameters for the tests. The symbol in the parenthesis
will indicate to the operator the type of response required.

    D - Decimal number
    O - Octal number
    L - logical (Y/N)

The value following the response type indication will indicate the
default value. The default value may be chosen by entering a carriage
return <CR> to the question. The defaults given here are the defaults
that would appear if the program was just loaded. Should any answer
be changed, the new value will become the default until the diagnostic
is re-loaded. In the SOFTWARE and MANUAL INTERVENTION questions, the
new value will be displayed as the default in any subsequent passes
through the test questions.

4.5.1 HARDWARE QUESTIONS -

When the diagnostic is started, hardware description tables must be
created for each drive under test. In order to do this the supervisor
will ask the following hardware questions.

CHANGE HW (L) ?

This question is always asked. If the answer is 'Y' then the other
HARDWARE QUESTIONS will be asked. If the answer is 'N', the
previously defined hardware tables will be used. The hardware tables
can be pre-defined with the SETUP utility of XXDP+. If the SETUP
utility has not been run, the default values listed below can be used
to create the hardware table.

OF UNITS (D) ?

If only TEST 1 is selected, enter the number of controllers to be
tested.   If TEST 2, TEST 3 or all of the tests (the default) are
selected, enter the number of drives to be tested. There is no
default for this question.

The next 3 questions are asked for each unit selected:

CSR ADDRESS OF CONTROLLER (O) 172150 ?

Enter the CSR address for controller as addressed by the processor
with memory management turned off (i.e. an even 16-bit address in the
range 160000 to 177774). The default CSR used is 172150.

DRIVE # (D) 0 ?

Enter the unit number of the drive to be tested. The answer must be
in the range of 0 through 254. The default unit number is 0. If only
TEST 1 is being used, the default should be taken for this question.

WRITE ON CUSTOMER DATA AREA (L) N ?

This questions is asked to minimize the risk of destroying customer
data.   The default answer of 'N' to this question causes the drive to
be write-protected and no write operations will be performed on the
drive.   An answer of 'Y' will designate the drive as writeable. The
default should be selected when only TEST 1 is selected.

4.5.2 SOFTWARE QUESTIONS -

The software questions allow the operator to tailor the operation of

SEQ 0021

the diagnostic. The software questions only apply to TEST 3.

CHANGE SW (L) ?

This question is always asked. If the answer is 'Y' then the other
software questions will be asked. If the answer is 'N' then all of
the default values for the test parameters will be used and the test
will begin. If the SETUP utility is used to pre-define the test
parameters, the pre-defined parameters will be used instead of the
normal defaults.

The following message will be printed if the operator chooses to
change the software parameters:

THE FOLLOWING QUESTIONS APPLY ONLY TO TEST 3:

ENTER MANUAL INTERVENTION MODE (L) N ?

An answer of 'Y' causes the manual intervention questions to be asked
for TEST 3. An answer of 'N', the default, causes the manual
intervention questions to be skipped. See the MANUAL INTERVENTION
QUESTIONS section for more information on these questions.

HARD ERROR LIMIT (D) 1 ?

Enter the number of hard errors that will cause a drive to be dropped.
Responding to this question with a zero requests that a drive never be
dropped from testing due to exceeding the hard error limit. This
parameter has no effect on device fatal errors. One device fatal
error will cause a drive to be dropped. By default, a drive is
dropped from testing after 1 hard error. A number in the range 0 to
65535 will be accepted.

EXERCISER TIME LIMIT IN MINUTES (D) 60 ?

Enter the time in minutes to run the exerciser. Responding to this
question with a zero indicates that the test will only complete if the
error limit is exceeded on all drives, or the operator types a
CONTROL-C. A number in the range 0 to 65535 will be accepted. The
default is 60 minutes.

MINUTES BETWEEN STATISTICAL REPORTS (D) 15 ?

Enter the time in minutes that will elapse between statistical
reports. Responding to this question with a zero indicates that the
statistical reports are not printed except when the operator types
CONTROL-C and types PRINT or the test completes. A number in the
range 0 to 65535 will be accepted. the default is 15 minutes. See
the EXERCISE SUMMARY section for a description of the statistical
reports.

PRINT SOFT ERROR MESSAGES (L) N ?

Enter 'Y' if a message should be printed whenever a soft error occurs.
Soft errors include reporting of MSCP Error Log messages. An answer
of 'N' will cause only messages for hard errors, data compare errors
and device errors to be printed.

DO DATA PATTERN VERIFICATION ON READS (L) Y ?

An answer of 'Y' will cause the diagnostic to compare data read from
the disk with data in host memory. The data is first read into host
memory. Using the number in the first word of the block read, the
host builds a block containing the expected data pattern and compares
that with what was read. This pattern verification will fail if the
drive has not been previously written with the standard diagnostic
data patterns listed in the TEST DATA PATTERNS section. If selected,
this data comparison will be randomly performed on read operations.
By default data pattern verification on read operations is performed.

DO DATA PATTERN VERIFICATION ON WRITES (L) N ?

An answer of 'Y' will cause the diagnostic to compare data written to
the disk with data in host memory. The data is first written to the
disk and then read back by the controller and compared to the host
memory buffer. If selected, this data comparison will be randomly
performed on write operations. By default no data pattern
verification on write operations is performed since selecting this
option degrades the data transfer rate.

USE VARIABLE LENGTH TRANSFERS (L) Y ?

Answer 'Y' if the buffers written and/or read from the drives should
be a variable length. Answer 'N' if the buffers should be fixed in

size.   The default is variable length buffers.

MAXIMUM TRANSFER SIZE IN BLOCKS (D) 8 ?

Enter the maximum number of sectors to be read/written on one I/O
operation.   If fixed length transfers are selected, the number of
sectors written and read will be equal to the buffer size except when
a read or write of that length is impossible -- such as doing the last
read or write to the end of data area. If variable length transfers
are selected, the buffer size is a maximum, so the actual number of
sectors will vary from 1 to the buffer size randomly.   The maximum
value for the buffer size is limited by the amount of available memory
in the system. The default maximum transfer size is 8 blocks.   A
number in the range 1 to 128 will be accepted.

ENABLE ERROR RETRIES (L) Y ?

An answer of 'Y', the default, will allow retries of I/O operations
that result in data errors. An answer of 'N' will disable retries and
log the error as an uncorrectable data error.

ENABLE ECC DATA CORRECTION (L) Y ?

Enter 'Y' if ECC should be used to correct data errors during the
test.   An answer of 'N' will cause ECC errors to be treated as hard
errors and retries will be attempted if enabled. By default, ECC is
used for data correction.

RANDOMLY ACCESS DRIVE (L) Y ?

Enter 'N' to sequentially transfer data to or from the disk.
Sequential access begins I/O at the lowest LBN in the first BEGIN/END
set and continues sequentially to the highest LBN in the last
BEGIN/END set.   Enter 'Y' to transfer data randomly throughout the
BEGIN/END sets. By default I/O is performed randomly.

DO DETERMINISTIC PHASE (L) Y ?

The deterministic phase writes the standard data patterns to the
entire disk and then reads back the first 256 KB from the disk. The
remainder of the disk is then ACCESSed. The deterministic phase is
only executed in the first pass through TEST 3. By default the

deterministic phase of TEST 3 is performed.

4.5.3 MANUAL INTERVENTION QUESTIONS -

If MANUAL INTERVENTION FOR TEST 3 is selected, the following set of
questions will be asked when TEST 3 begins. These questions determine
the parameters that must be set up on a per-drive basis and are not
applicable to the normal operation of this diagnostic program. The
answers to these questions cannot be supplied using the SETUP utility.
If this diagnostic program is run in unattended mode, the defaults
will be used.

At the beginning of the questions the following message will be
printed:

TEST 3 MANUAL INTERVENTION QUESTIONS:

DATA PATTERN TO USE (D) 0 ?

This question only applies to drives that are writeable.   Enter the
pattern, 0 through 16, to be used for testing. If pattern number zero
is selected, the test will randomly write each of the sixteen data
patterns.   See the TEST DATA PATTERNS section for a description of
data patterns 1 through 15 and default pattern 16.   An operator
defined pattern can be specified for pattern 16. The default data
pattern is 0, all patterns randomly selected.

If PATTERN 0 or PATTERN 16 is selected for the drives under test, the
following questions are asked:

MODIFY DATA PATTERN 16 (L) N?

Enter 'Y' to modify data pattern 16. If data pattern 16 is modified,
the deterministic phase of TEST 3 should be run so pattern
verification errors don't occur. By default the standard pattern is
used.

HOW MANY WORDS IN PATTERN 16 (D) 1?

This question and the questions pertaining to the data words in
pattern 16 will only be asked if the user desires to modify pattern
16. Enter the number of words in the pattern. The pattern may not
consist of more than 16 words. The following questions will be asked

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 24

until all of the words in the pattern are defined:

```
DATA WORD (0) 000000 ?
DATA WORD (0) 000000 ?
DATA WORD (0) 000000 ?
        .
        .
        .
```

For each drive selected, the following questions will be asked:

THESE QUESTIONS REFER TO UNIT xx CONTROLLER AT xxxxxx DRIVE xxx

TEST OVER THE ENTIRE DISK (L) Y ?

Enter 'Y', the default, if the test is to be run over the entire customer data section of the drives under test or 'N' if testing should be limited to specific BEGIN/END sets. The following questions will be asked if the answer is 'N':

NUMBER OF BEGIN/END SETS TO USE (D) 1 ?

Enter the number of BEGIN/END sets to be used in the test. All testing will be done within the bounds of the BEGIN/END sets. The default is 1 BEGIN/END set consisting of the entire customer data area. The maximum number of BEGIN/END sets that can be specified for each drive is four. For each BEGIN/END set selected, the following question will be asked:

STARTING LBN (D) 0 ?

ENDING LBN (D) 0 ?

The valid LBN range is 0 to the maximum LBN in the customer data area of the disk. Since the number of LBNs on the disk is not known to the program until the drive is brought online, bounds checking is not done until the drive is brought online.The ending LBN must be greater than or equal to the starting LBN.

4.6 START-UP PROCEDURE

To run this program:

1. Boot XXDP+

2. Type "R ZUDJAO"

3. Type "STA"

4. If the hardware parameters need to be changed, answer the
   "CHANGE HW" question with "Y" and answer all of the hardware
   questions. Otherwise, answer the "CHANGE HW" question with "N".

5. If the software parameters need to be changed, answer the
   "CHANGE SW" question with "Y" and answer all of the software
   questions. Otherwise, answer the "CHANGE SW" question with "N".

Below is a sample of terminal dialogue to test two disks on one
controller.

DR>STA/FLA:PNT/PAS:1

CHANGE HW (L) ? Y

UNITS (D) ? 2

UNIT 0
CSR ADDRESS OF CONTROLLER (O) 172150 ?
DRIVE # (D) 0? 0,1
WRITE ON CUSTOMER DATA AREA (L) N ? Y<CR>

CHANGE SW (L) ? Y

THE FOLLOWING QUESTIONS APPLY ONLY TO TEST 3:

ENTER MANUAL INTERVENTION MODE (L) N ?
HARD ERROR LIMIT (D) 1 ?
EXERCISER TIME LIMIT IN MINUTES (D) 60 ?
MINUTES BETWEEN STATISTICAL REPORTS (D) 15 ?
PRINT SOFT ERROR MESSAGES (L) N ?
ENABLE ERROR LOG (L) N ?
DO DATA PATTERN VERIFICATION ON READS (L) Y ?
DO DATA PATTERN VERIFICATION ON WRITES (L) N ?
USE VARIABLE LENGTH TRANSFERS (L) Y ?
MAXIMUM TRANSFER SIZE IN BLOCKS (D) 16 ?
ENABLE ERROR RETRIES (L) Y ?
ENABLE ECC DATA CORRECTION (L) Y ?
RANDOMLY ACCESS DRIVE (L) Y ?

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 26

DO DETERMINISTIC PHASE (L) Y ?

TST: 001
TST: 002
TST: 003

CZUDJ end of pass 1
    number of errors 0

DR>

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 27

5 ERROR INFORMATION

5.1 ERROR REPORT FORMAT

The general error format is shown below. Note that informational
messages do not follow this general format, because they are not
errors. In the example below, all words in lower case letters
indicate values that are generated at run time.

CZUDJO severity err_num ON UNIT unit TST test SUB subtest PC: pc
test_descr CONTROLLER AT csr DRIVE drive RUNTIME hh:mm:ss
error_text
additional_information

severity:
    Identifies the severity of error. The four levels of severity
    are: 'SFT ERR' for soft errors, 'HRD ERR' for hard errors, 'DEV
    FTL' for device fatal errors, and 'SYS FTL' for system fatal
    errors.

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

    Device fatal errors (DVC FTL ERR) are used to report errors that
    are fatal to the device (may be either a UDA50-A or KDA50-Q or
    disk drive). Testing stops on that device for the remainder of
    the current pass.

    Hard errors (HRD ERR) are unrecoverable non-fatal errors.
    Testing will normally continue after the printing of the error
    unless the hard error limit has been reached.

    Soft errors (SFT ERR) indicate a problem was encountered that was
    resolved by re-trying the operation or by using ECC.

errnum:
    This is the number (decimal) of the error that occured.

unit:
    This is the number assigned to each unit under test by the
    diagnostic supervisor. Any adds or drops of units should refer
    to this number.

test:

This is the test number of the currently running test.

subtest:
This is the number of the currently running subtest.

pc:
This is the diagnostic program's program counter at the time of
the error.

testdescr:
This is a brief description of the test being performed when the
error occurred.   The description will be one of the following:
TEST   INITIALIZATION,   CONTROLLER   VERIFICATION,   SUBSYSTEM
VERIFICATION, or SUBSYSTEM EXERCISER.

csr:
This is the bus address of the controller's IP register. This is
only printed if the error involves the controller.

drive:
This is the drive's unit plug number which is on the front panel
of the drive. This is only printed if the error involves a disk
drive.

hh:mm:ss:
This is how long the diagnostic program has been running.

errortext:
This is the error (in english) that occurred.

additionalinformation:
This is information to help the operator diagnose the cause of
the problem. The additional information reported depends on the
type of error.

There are three levels of error messages that may be issued by a
diagnostic: general, basic and extended. General messages consist of
the first line of the error.   They indicate the name of the
diagnostic, the error type and number, the failing unit, the test and
subtest being executed, and the memory address of the error call in
the diagnostic. General error messages are always printed unless the
"IER" flag is set.

Basic error messages consist of at least two lines of text describing
the error.   The first line reports the name of the test, the
controller (if any) currently being tested, the drive (if any) being
tested, and the elapsed time since the program was started. The other

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 29

lines report the nature of the error. These are always printed unless
the "IER" or "IBE" flags are set. 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
basic error messages.

Sample error message:

CZUDJ HRD ERR 00038 ON UNIT 00 TST 003 SUB 000 PC: 045416
SUBSYSTEM EXERCISER - CONTROLLER AT 172150, DRIVE 5 - RUNTIME 0:20:12
DATA COMPARE ERROR DURING DATA PATTERN VERIFICAION
LBN: 000123, BYTE OFFSET: 123
DATA EXPECTED: 000000, DATA READ: 121212

Some informational messages are also printed by this program.   They
are one or two lines in length and consist of a time stamp and the
message text. These messages are for informational purposes only and
their contents should be self explanitory.

5.2 SPECIFIC ERROR MESSAGES

Following is a list of the error messages that can be printed by the
diagnostic program.   In the list, some of the numbers that may vary
with execution or program version are shown as lower case words.
These include program counters and runtime. Other numbers, such as
unit number, drive number, UDA50-A or KDA50-Q address and data in
registers are filled with sample numbers. Additional information
about the error follows the error message.

5.2.1 00002 - TWO UNITS SELECT THE SAME DRIVE -

CZUDJ SYS FTL ERR 00002 ON UNIT unit TST tst SUB sub PC: hostpc
TEST INITIALIZATION - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
INVALID ANSWERS GIVEN TO THE HARDWARE QUESTIONS
TWO UNITS SELECT THE SAME DRIVE

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

5.2.2 00003 MORE THAN 4 DRIVES SELECTED ON THIS CONTROLLER -

CZUDJ SYS FTL ERR 00003 ON UNIT unit TST tst SUB sub PC: hostpc
TEST INITIALIZATION - CONTROLLER AT csr - RUNTIME hh:mm:ss
INVALID ANSWERS GIVEN TO THE HARDWARE QUESTIONS
MORE THAN 4 DRIVES SELECTED ON THIS CONTROLLER

Up to four physical disk drives can be attached to a UDA50-A or
KDA50-Q at one time.   The program is aborted and returns to the
supervisor prompt so that you can change the hardware questions.

5.2.3 00004 NOT ENOUGH MEMORY TO TEST THE UNITS SELECTED -

```
CZUDJ SYS FTL ERR  00004 ON UNIT unit TST tst SUB sub PC: hostpc
TEST INITIALIZATION - RUNTIME hh:mm:ss
NOT ENOUGH MEMORY TO TEST THE UNITS SELECTED
RESTART PROGRAM 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.   The amount of memory
available to the program is inadequate for the the number of units
selected. Start program over and select fewer units.

5.2.4 00005 INVALID BEGIN/END SET SPECIFIED -

```
CZUDJ SYS FTL ERR  00005 ON UNIT unit TST tst SUB sub PC: hostpc
SUBSYSTEM EXERCISER - CONTROLLER AT csr , DRIVE drive - RUNTIME hh:mm:ss
INVALID BEGIN/END SET SPECIFIED
BEGIN/END SET SPECIFIES OUT-OF-RANGE LBN.
MAXIMUM LBN: xxxxxx
```

A begin/end set specified in the manual intervention questions of test
3 specifies a LBN that does not exist on the drive under test.
Re-start the program and specify a valid LBN range.

5.2.5 00006 - DATA STRUCTURE CORRUPT -

```
CZUDJ SYS FTL ERR  00006 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - RUNTIME hh:mm:ss
DATA STRUCTURE CORRUPT AT LOCATION xxxxxx. RE-LOAD PROGRAM
```

A number of data structures are created by the diagnostic program. If
the program detects a corruption of one of these data structures, this
error will be printed. Please report this error to the maintaining
group since this usually indicates a program bug. Often, the program
will run without error once it has been re-loaded.

5.2.6 00014 - CONTROLLER IS NOT SUPPORTED BY THIS DIAGNOSTIC PROGRAM

CZUDJ SYS FTL ERR 00014 ON UNIT unit TST tst SUB sub PC: hostpc
TEST INITIALIZATION - CONTROLLER AT csr - RUNTIME hh:mm:ss
CONTROLLER IS NOT SUPPORTED BY THIS DIAGNOSTIC PROGRAM.
THIS PROGRAM REQUIRES A UQSSP DISK TYPE CONTROLLER
CONTROLLER REPORTED MODEL CODE xx

This diagnostic program was designed to test any UQSSP disk type
controller though due to the differences in the controllers, only the
UDA50-A (modules M7435-6) and KDA50-Q (modules M7164-5) controllers
are fully supported. The diagnostic will attempt to test the RC25 and
RQDX1, but the rusults are not guarenteed to be 100% correct.   No
other controllers can be tested by this diagnostic.

5.2.7 00021 - FATAL ERROR REPORTED BY CONTROLLER -

CZUDJ DVC FTL ERR 00021 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
FATAL ERROR REPORTED BY CONTROLLER
SA REGISTER CONTAINS: xxxxxx
REPLACE CONTROLLER module MODULE

The controller resident diagnostic detected a failure. The error is
displayed in the SA register. The program will attempt to determine
which of the controller boards failed and report it in the last line
of the error message. A list of the possible error codes and their
meaning are contained in an appendix to this document.

5.2.8 00022 - STEP BIT DID NOT SET IN SA REGISTER DURING INITIALIZATION -

CZUDJ DVC FTL ERR 00022 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
STEP BIT DID NOT SET IN SA REGISTER DURING INITIALIZATION
STEP BIT EXPECTED: xxxxxx
SA REGISTER CONTAINS: xxxxxx
REPLACE CONTROLLER PROCESSOR MODULE

The controller did not respond as expected during the initialization
sequence which communicates using data in the SA register. A normal
response from the controller 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.

5.2.9 00023 - CONTROLLER DID NOT CLEAR RING STRUCTURE IN HOST MEMORY

```
CZUDJ DVC FTL ERR  00023 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
CONTROLLER DID NOT CLEAR PORT COMMUNICATIONS AREA DURING INITIALIZATION
x WORDS WERE TO BE CLEARED STARTING AT ADDRESS xxxxxx
THE FOLLOWING WORDS NOT CLEARED:
    ADDRESS     CONTENTS
    xxxxxx      xxxxxx
    xxxxxx      xxxxxx
    xxxxxx      xxxxxx
REPLACE CONTROLLER PROCESSOR MODULE
```

The controller is to clear the ring structure (a communications area
used by the controller to talk to the host) in host memory before Step
4 of initialization. If the controller 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 controller to clear each word
indicates a fault in the address interface to the bus.

5.2.10 00024 - SA REGISTER DID NOT GO TO ZERO AFTER STEP 3 -

```
CZUDJ DVC FTL ERR  00024 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
SA REGISTER DID NOT GO TO ZERO AFTER STEP 3 WRITE OF INITIALIZATION
PURGE/POLL DIAGNOSTICS WERE REQUESTED
SA REGISTER CONTAINS: xxxxxx
```

During step 3 of the controller initialization, the host can test the
PURGE and POLL mechanism of the controller. To do so the host sets
bit15 of the step 3 data and sends the data to the controller.   The
controller must go to zero and wait for the adapter purge completion
and polling to begin. If the controller never goes to zero, then
error message 00024 is displayed.   The controller may have a bad
processor module or the bus maybe broken.

5.2.11 00025 - INCORRECT DATA RETURNED IN SA REGISTER -

```
CZUDJ DVC FTL ERR  00025 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
INCORRECT DATA RETURNED IN SA REGISTER DURING INITIALIZATION
DATA EXPECTED: xxxxxx
```

SA REGISTER CONTAINS: xxxxxx
REPLACE CONTROLLER PROCESSOR MODULE

For each step of initialization, specific data is expected to be
displayed in the SA register. If the contents of the SA register does
not match the expected data, then error message 00025 is displayed.
Replace controller processor module.

5.2.12 00026 - DATA COMPARISON ERROR DURING WRAP MODE TEST -

```
CZUDJ DVC FTL ERR  00026 ON UNIT unit TST tst SUB sub PC: hostpc
CONTROLLER VERIFICATION - CONTROLLER AT csr - RUNTIME hh:mm:ss
DATA COMPARISON ERROR DURING WRAP MODE TEST
DATA EXPECTED: xxxxxx
SA REGISTER CONTAINS: xxxxxx
REPLACE CONTROLLER PROCESSOR MODULE
```

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

5.2.13 00027 - SA REGISTER DID NOT CHANGE AFTER WRITING TO IT -

```
CZUDJ DVC FTL ERR  00027 ON UNIT unit TST tst SUB sub PC: hostpc
CONTROLLER VERIFICATON - CONTROLLER AT csr - RUNTIME hh:mm:ss
SA REGISTER DID NOT CHANGE AFTER WRITING TO IT IN WRAP MODE TEST
SA REGISTER CONTAINS: xxxxxx
REPLACE CONTROLLER PROCESSOR MODULE
```

The controller can be put into a mode where the SA register acts as a
wrap port. While the controller is in this mode, any data being sent
to the SA register will be displayed in the SA register within a small
period of time. If, after the host program sends data to it while it
was in diagnostic wrap mode, the controller does not change the
contents of the SA register, error message 00027 is displayed.
Replace controller processor module.

5.2.14 00030 - FATAL ERROR REPORTED BY CONTROLLER -

```
CZUDJ DVC FTL ERR  00030 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
FATAL ERROR REPORTED BY CONTROLLER
SA REGISTER CONTAINS: xxxxxx
```

A message from the controller firmware reports an unexpected failure.
An error code is presented in the SA register. See the appendix for a
list of codes and their meaning.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 37

5.2.15 00031 - CONTROLLER TIMED OUT BY HOST -

```
CZUDJ DVC FTL ERR  00031 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
CONTROLLER TIMED OUT BY HOST, NO PROGRESS MADE ON COMMAND
```

The controller is required to either complete a command or make progress on the command within a controller specified timeout interval. If progress has not been made on the oldest outstanding command, the host assumes the controller has failed and drops it from testing.

5.2.16 00032 - PACKET RECEIVED FROM CONTROLLER WITH UNKNOWN ENDCODE -

```
CZUDJ DVC FTL ERR  00032 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
PACKET RECEIVED FROM CONTROLLER WITH UNKNOWN ENDCODE
ENDCODE: xxx
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
```

A message packet was received from the controller with an endcode unknown to the diagnostic program. The endcode is contained in byte 8 of the packet. If the endcode is valid, a problem report should be submitted to the diagnostic maintenance group.

5.2.17 00033 - PACKET RECEIVED FROM CONTROLLER WITH UNKNOWN STATUS -

```
CZUDJ DVC FTL ERR  00033 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
PACKET RECEIVED FROM CONTROLLER WITH UNKNOWN STATUS
STATUS: status
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
```

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 38

A message packet was received from the controller with status code
unknown to the diagnostic program. If the status code is valid, a
problem report should be submitted to the diagnostic maintenance
group.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 39

5.2.18 00034 - CONTROLLER DETECTED AN INVALID COMMAND PACKET -

```
CZUDJ DVC FTL ERR  00034 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
CONTROLLER DETECTED AN INVALID COMMAND PACKET
FIELD AT OFFSET xx IS BAD
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 controller has detected invalid data in a command packet received
from the host.   The byte offset of the invalid field is included in
the message. This error could be caused either by a program error or
a system hardware error. If the program appears to be generating an
invalid command packet, send a problem report to the diagnostic
maintenance group.

5.2.19 00035 - PACKET RECEIVED FROM CONTROLLER SPECIFIES UNKNOWN DRIVE -

```
CZUDJ DVC FTL ERR  00035 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
PACKET RECEIVED FROM CONTROLLER SPECIFIES UNKNOWN DRIVE
DRIVE UNIT NUMBER: xx
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 controller sent a message to the host referencing a drive which
doesn't have an entry in the host program's drive data tables. The
message sent to the host program is also included in the error
message.   The word at byte offset 4 contains the drive number
specified. This error can indicate either a software or hardware
error.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 40

5.2.20 00037 - MEMORY ERROR TRYING TO READ CONTROLLER REGISTERS -

CZUDJ DVC FTL ERR 00037 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
MEMORY ERROR TRYING TO READ CONTROLLER REGISTERS
EITHER THE CONTROLLER CSR ADDRESS WAS INCORRECTLY SPECIFIED
OR THERE IS A BUS OR CONTROLLER PROBLEM

A non-existant memory error occurred when the host program tried to
access the controller IP or SA registers.   The controller is at
another CSR address (check the bus selection switches) or the
processor module is broken or the bus is broken.

5.2.21 00038 - DATA COMPARE ERROR DURING DATA PATTERN VERIFICATION -

```
CZUDJ HRD ERR  00038 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
DATA COMPARE ERROR DURING DATA PATTERN VERIFICATION
LBN: xxxxxx,  BYTE OFFSET: xxxxxx
DATA EXPECTED: xxxxxx,  DATA READ: xxxxxx
```

If selected by the operator, data pattern verification is randomly
performed by the program during read operations. The first word of
the block read is checked for a pattern number. The pattern indicated
by the pattern number is generated by the program and the block read
is checked to make sure it contains the correct pattern. If the data
read is not the same as the data pattern generated byt the host, this
error will occur. This usually indicates a problem transferring data
between the host and the controller. If the standard diagnostic data
patterns have not been written to the disk by this program or another
RA series disk diagnostic, this test will fail.   The standard
diagnostic data patterns are listed later in this document.

5.2.22 00039 - DATA COMPARE ERROR DURING CONTROLLER MEMORY TEST -

```
CZUDJ DVC FTL ERR  00039 ON UNIT unit TST tst SUB sub PC: hostpc
CONTROLLER VERIFICATION - CONTROLLER AT csr - RUNTIME hh:mm:ss
DATA COMPARE ERROR DURING CONTROLLER MEMORY TEST
CONTROLLER MEMORY ADDRESS: xxxxxx
DATA WRITTEN: xxxxxx DATA READ: xxxxxx
```

The controller verification test performs a controller memory test by
writing known data patterns to the controller memory and then reading
the data back. If the data read is not the same as the data written
this error will occur. The data pattern used is a floating inversion
pattern. This usually indicates a bad memory location in the
controller, a controller memory addressing problem, or a data transfer
problem.

5.2.23 00041 - MSCP OPERATION FAILED -

```
CZUDJ DVC FTL ERR  00041 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
operation MSCP OPERATION FAILED
STATUS/EVENT CODE: status
BAD BLOCK REPORTED. LBN: xxxxxx
```

```
ADDITIONAL BAD BLOCKS UNREPORTED
ERROR LOG GENERATED
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
```

This error message indicates a MSCP operation has failed. The
operation that failed is indicated in the error message. The status
returned by the controller and the response packet are also displayed.
If bad blocks are detected or an error log message is generated, these
will also be reported to the user. For more information on the
information returned by the controller, see the MSCP architectural
specification and the device service manuals.

5.2.24 00042 - PREVIOUS FATAL CONTROLLER ERROR -

```
CZUDJ DVC FTL ERR  00042 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
PREVIOUS FATAL CONTROLLER ERROR CAUSED THE CONTROLLER AND
THE FOLLOWING DRIVES TO BE DROPPED: x, x, x, x,
```

This error message will be printed after a fatal controller error has
occurred to inform the operator that all of the drives have been
dropped from testing.

5.2.25 00043 - PREVIOUS FATAL DRIVE ERROR -

```
CZUDJ DVC FTL ERR  00043 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
PREVIOUS FATAL DRIVE ERROR CAUSED THIS DRIVE TO BE DROPPED
```

This error message will be printed after a fatal drive error has
occurred to inform the operator that the drive has been dropped from
testing.

5.2.26 00050 - ERROR LOG RECEIVED - CONTROLLER ERROR -

```
CZUDJ SFT ERR  00050 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
ERROR LOG RECEIVED - CONTROLLER ERROR
OPERATION status
STATUS/EVENT CODE: status
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
```

This error message indicates that a error log message was received
from the controller indicating a controller error has occurred. The
'OPERATION status' line of the message will tell whether the MSCP
operation that caused the error is continuing, was successful, or
failed. The event code for the message and the error log message are
reported to the operator. If the operation failed, a error message of
type 00041 will also be printed. Error log messages will only be
reported to the operator if the printing of soft error messages is

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 44

enabled in the software questions. More information on the contents
of the error log message can be found in the device's service manual
and in the MSCP architectural specification.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 45

5.2.27 00051 - ERROR LOG RECEIVED - HOST MEMORY ACCESS ERROR -

```
CZUDJ SFT ERR  00051 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr - RUNTIME hh:mm:ss
ERRCR LOG RECEIVED - HOST MEMORY ACCESS ERROR
HOST MEMORY ADDRESS: xxxxxx
OPERATION status
STATUS/EVENT CODE: status
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
```

This error message indicates that a error log message was received
from the controller indicating a host memory access error has
occurred.   The memory location being accessed is reported in decimal.
The 'OPERATION status' line of the message will tell whether the MSCP
operation that caused the error is continuing, was successful, or
failed.   The event code for the message and the error log message are
reported to the operator. If the operation failed, a error message of
type 00041 will also be printed. Error log messages will only be
reported to the operator if the printing of soft error messages is
enabled in the software questions. More information on the contents
of the error log message can be found in the device's service manual
and in the MSCP architectural specification.

5.2.28 00052 - ERROR LOG RECEIVED - DISK TRANSFER ERROR -

```
CZUDJ SFT ERR  00052 ON UNIT unit TST tst SUB sub PC: hostpc
test_descr - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
ERROR LOG RECEIVED - DISK TRANSFER ERROR
RETRY COUNT: xx  RECOVERY LEVEL: xx
OPERATION status
STATUS/EVENT CODE: status
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
```

This error message indicates that a error log message was received
from the controller indicating a disk transfer error has occurred.

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 46

The retry count and the recovery level are reported in decimal.   The
'OPERATION status' line of the message will tell whether the MSCP
operation that caused the error is continuing, was successful, or
failed.   The event code for the message and the error log message are
reported to the operator. If the operation failed, a error message of
type 00041 will also be printed. Error log messages will only be
reported to the operator if the printing of soft error messages is
enabled in the software questions. More information on the contents
of the error log message can be found in the device's service manual
and in the MSCP architectural specification.

5.2.29 00053 - ERROR LOG RECEIVED - SDI ERROR -

CZUDJ SFT ERR 00053 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTRGLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
ERROR LOG RECEIVED - SDI ERROR
SDI STATUS (R TO L): xxxxxx xxxxxx xxxxxx xxxxxx xxxxxx xxxxxx
OPERATION status
STATUS/EVENT CODE: status
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

This error message indicates that a error log message was received
from the controller indicating a SDI error has occurred. The 12 byte
SDI status is reported as 6 octal words. The low order byte is on the
right and the high order byte is on the left. The 'OPERATION status'
line of the message will tell whether the MSCP operation that caused
the error is continuing, was successful, or failed. The event code
for the message and the error log message are reported to the
operator. If the operation failed, a error message of type 00041 will
also be printed. Error log messages will only be reported to the
operator if the printing of soft error messages is enabled in the
software questions. More information on the contents of the error log
message can be found in the device's service manual and in the MSCP
architectural specification.

5.2.30 00054 - ERROR LOG RECEIVED - SMALL DISK ERROR -

```
CZUDJ SFT ERR  00054 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
ERROR LOG RECEIVED - SMALL DISK ERROR
CYLINDER: xxxxxx
OPERATION status
STATUS/EVENT CODE: status
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
```

This error message indicates that a error log message was received
from the controller indicating a small disk error has occurred. The
cylinder being accessed when the error occurred is reported in
decimal. The 'OPERATION status' line of the message will tell whether
the MSCP operation that caused the error is continuing, was
successful, or failed. The event code for the message and the error
log message are reported to the operator. If the operation failed, a
error message of type 00041 will also be printed. Error log messages
will only be reported to the operator if the printing of soft error
messages is enabled in the software questions. More information on
the contents of the error log message can be found in the device's
service manual and in the MSCP architectural specification.

5.2.31 00060 - AVAILABLE ATTENTION MESSAGE RECEIVED -

```
CZUDJ DVC FTL ERR  00060 ON UNIT unit TST tst SUB sub PC: hostpc
test descr - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
AVAILABLE ATTENTION MESSAGE RECEIVED
DRIVE UNDER TEST BECAME UNIT-AVAILABLE
```

This error message indicates that a drive that was selected for
testing unexpectedly became 'UNIT-AVAILABLE'. This indicates an
unexpected fault in either the controller or the disk drive. This
error will result in the drive being dropped from testing.

5.2.32 00061 - DUPLICATE UNIT ATTENTION MESSAGE RECEIVED -

```
CZUDJ DVC FTL ERR  00061 ON UNIT unit TST tst SUB sub PC: hostpc
```

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 48

        test_descr - CONTROLLER AT csr, DRIVE drive - RUNTIME hh:mm:ss
        DUPLICATE UNIT ATTENTION MESSAGE RECEIVED
        DUPLICATE UNIT NUMBER DETECTED BY CONTROLLER

        This error message indicates that the controller detected two or more
        drives with the same unit number. The controller will disable both
        drives and the unit will be dropped from testing.

6 PERFORMANCE AND PROGRESS REPORTS

A summary will be printed out whenever one of three conditions are met: the operator CONTROL-C's, then types 'PRINT', a test completes or the time limit between summaries expires.

The exercise summary will consist of a header and two tables, a SUBSYSTEM I/O SUMMARY and a DRIVE ERROR SUMMARY. The header will contain the time and date of the report, and the the elapsed time since the diagnostic was started.

The SUBSYSTEM I/O SUMMARY will contain the following information:

        Controller CSR
        Controller unique identifier
        Drive name
        Drive unique identifier
        Number of bytes written (controller and drive)
        Number of bytes read (controller and drive)
        Number of bytes accessed (drive only)
        Number of bytes that have completed data compares (drive only)
        Number of blocks containing ECC correctable data (drive only)

The DRIVE ERROR SUMMARY consists of the following information:

        Unit number
        Drive name
        Drive media serial number
        Drive status
        Number of hard/device fatal errors
        Number of data check errors
        Number of soft errors

UDA50-A/KDA50-Q Subsystem Exerciser User Guide   Page 50

The format for the exercise summary is as follows:

TEST x IN PROGRESS - RUNTIME: hhhh:mm:ss

SUBSYSTEM I/O SUMMARY:

CTRLR CSR	DRIVE NAME		UNIQUE IDENTIFIER	BYTES WRITTEN	BYTES READ	BYTES ACCESSED	BYTES COMPARED	ECC DATA
172150		01 13	1234567890123456	950K	923K			
	DU0	05 02	5678901234567890	456M	26M	456M	258K	0
	DU252	05 02	6543210987654321	456M	28M	456M	128K	1

DRIVE ERROR SUMMARY:

UNIT #	DRIVE NAME	VOLUME SERIAL #		HARD/FATAL ERRORS	DATA CHECK ERRORS	SOFT ERRORS
0	DU0			0	0	0
1	DU252	0000123456	(DROPPED)	1	8	0

Note drive DU252 which has been dropped -- when a drive has been
dropped, the operator will always be notified in the summary that the
drive has been dropped.

In the "BYTES" fields, if the byte count is from 0 through 999, the
field contains th actual number of bytes. If the byte count is from
1000 to 999,999, the byte count indicates the number of kilobytes,
which will be designated by a trailing "K". If the byte count is
1,000,000 or higher, the byte count indicates the number of megabytes,
which will be designated by a trailing "M".

If the operator requested an interval between summaries, the interval
is measured from the last summary. For example, if a 15 minute
interval is selected between summaries, and at 14 minutes the operator
CONTROL-C's then types SUMMARY, the next summary will not occur for 15
more minutes.

7 GLOSSARY

"CIS"

Commercial Instruction Set. An extension to the standard PDF-11
instruction set designed to increase string manipulation
performance.

"EIS"

Extended Integer Instructions. An extension to the standard
PDP-11 instruction set designed to improve the performance of
certain integer math functions. The instructions are ASH, ASHC,
DIV, and MUL.

"FRU"

Field Replaceable Unit. The smallest element of a device under
test that can be replaced in the field to resolve a device
problem.

"IP Register"

The IP register is one of two I/O page registers used for
communication between the host and the controller.   The IP
register allows the host to initiate a controller initialization
and allows the host to initiate polling by the controller.

"LBN"

Logical Block Number. Address of a sector on a disk drive where
the LBN can be in the range 0 - (number_of_sectors_on_drive - 1).

"MSCP"

Mass Storage and Control Protocol. This is a protocol that is
used by a host processor to communicate with a disk controller to
perform disk I/O operations and I/O control functions.

"Revectored Sector"

A sector that was found to be bad by either the surface checker
in manufacturing, the SDI formatter during formatting, or a disk
drive controller during normal use. The sector is marked as bad,
and a good sector assigned to replace it.

"SA Register"

One of two I/O page registers used for communication between the
controller and the host.   The SA register is used by the
controller to provide data, status, and error information to the
host.   It is also used by the host to provide parameter
information and bus adaptor purge status to the controller.

"SDI"

Standard Disk Interface.   The interface between the disk
controller and the disk drive.   The SDI defines the physical
connection (SI), protocol and functions that can be performed
using the connection.

"SI"

Storage Interconnect. One of four physical connections from a
disk controller to a disk drive.

"Targeted Unit Under Test"

The smallest unit farthest away from the processor to which a
diagnostic can be directed.

"UQSSP"

UNIBUS/QBUS Storage Systems Port.   UQSSP defines   the
initialization sequence for the controller and the transport
mechanism for the MSCP and DUP messages passed between the host
and the controller over the UNIBUS or Q-BUS.

APPENDIX A

DRS COMMAND SUMMARY

DRS COMMANDS

Only the first three letters are needed for each command, the rest of the
letters are ignored. This is shown above by lower case letters.
The description above is intended to be an introduction only, for
a detailed description of these commands, see the XXDP+/SUPR USERS
MANUAL CHQUS??.

STArt	start the diagnostic and initialize
REStart	start the diagnostic and do not initialize
CONtinue	continue diagnostic at test that was interrupted by a ↑C
PROceed	continue from an error halt
ADD	activate a unit for testing
DROp	deactivate a unit
DISplay	print a list of device information
rLAgs	print status of all flags
ZFLags	reset all flags
PRInt	print statistical information
EXIt	return to XXDP+ runtime monitor
REDirect	redirect output to specified device

DRS COMMAND SWITCHES

/TES:test-list	execute only the tests specified
/PAS:ddddd	execute ddddd passes (ddddd = 1 to 64000)
/FLA:flag-list	set specified flags
/EOP:ddddd	report end of pass after each ddddd passes (ddddd = 1 to 64000)
/UNI:unit-list	command will affect only specified units

DRS COMMAND SUMMARY   Page A-2

```
/DEV:device              device to which ouput is re-directed
/LPT                     redirect output to lineprinter

DRS FLAGS

HOE      halt on error - control is returned to DRS
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)
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
IDR      inhibit program dropping of units
ADR      execute autodrop code
LOT      loop on test
EVL      execute evaluation on error
```

APPENDIX B

SA REGISTER ERROR CODE SUMMARY (OCTAL)

100001 - Envelope/Packet read error (parity or timeout)
100002 - Envelope/Packet write error (parity or timeout)
100003 - Controller ROM and RAM parity error
100004 - Controller RAM parity error
100005 - Controller ROM parity error
100006 - Ring read error (parity or timeout)
100007 - Ring write error (parity of timeout)
100010 - Interrupt master failure
100011 - Host access timeout
100012 - Credit limit exceeded
100013 - Q-bus master error
100014 - Diagnostic controller fatal error
100015 - Instruction Loop timeout
100016 - Illegal virtual circuit ID
100017 - Interrupt write error
100020 - Maintenance read/write invalid region identifier
100021 - Maintenance write load to non-loadable controller
100022 - Controller RAM error (non-parity)
100023 - INIT sequence error
100024 - High-level protocol incompatibility error
100025 - Purge/poll hardware failure
100026 - Mapping register read error (parity or timeout)
100027 - Mapping option unsupported
104000 - Fatal sequencer error
104040 - D PROC ALU test error
104041 - D PROC ROM parity test error / Timeout test error
105102 - D PROC no board 2 error / D PROC control reg test error /
         D PROC RAM parity error
105105 - D PROC RAM buffer error
105152 - D PROC SDI error
105153 - D PROC write mode, wrap SERDES 16 error
105154 - D PROC read mode, SERDES 16, 10 RSGEN and ECC circuitry error

SA REGISTER ERROR CODE SUMMARY (OCTAL)   Page B-2

```
106040 -  U PROC ALU test error / DFAIL test error / Unexpected trap error
106041 -  U PROC control reg test error
106042 -  U PROC parity error set erroneously / CROM parity test error
106055 -  Unexpected trap - abnormal termination of diagnostics
106071 -  U PROC Log/Antilog RAM checksum error
106072 -  U PROC ROM parity test error
106200 -  STEP 1 data error (MSB wasn't set)
107103 -  U PROC RAM parity error
107107 -  U PROC RAM buffer error
112300 -  STEP 2 INIT error
122300 -  STEP 3 INIT error
122240 -  DMA test error
142300 -  STEP 4 INIT error
```

APPENDIX C

TEST DATA PATTERNS

In order for CZUDJO to be able to perform a data compare on a sector read,
it must be able to determine the pattern that was written in the sector.
If the sector was not previously written by CZUDJO, the standard diagnostic
data patterns must have been written to the disk by a previously run
diagnostic.

Following is a list of the data patterns used by CZUDJO in the read and
write subtests.   Each of the data patterns is preceeded by one word that
contains the pattern number replicated four times. The data patterns will
be replicated throughout remaining words in the sector.

PATTERN 0   This pattern number is used to indicate any
             pattern number 1 to 16 chosen at random.

PATTERN 1   Words in pattern sequence - 1

             Sequence (octal) 105613

             Sequence (binary) 1000101110001011

PATTERN 2   Words in pattern sequence - 1

             Sequence (octal) 31463

             Sequence (binary) 0011001100110011

PATTERN 3   Words in pattern sequence - 1

             Sequence (octal) 30221

             Sequence (binary) 0011000010010001

TEST DATA PATTERNS   Page C-2

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 (binary) 0000000000000001
                    0000000000000011
                    0000000000000111
                    0000000000001111
                    0000000000011111
                    0000000000111111
                    0000000001111111
                    0000000011111111
                    0000000111111111
                    0000001111111111
                    0000011111111111
                    0000111111111111
                    0001111111111111
                    0011111111111111
                    0111111111111111
                    1111111111111111

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 (binary) 1111111111111110
                    1111111111111100
                    1111111111111000
                    1111111111110000
                    1111111111100000
                    1111111111000000
                    1111111110000000
                    1111111100000000
                    1111111000000000
                    1111110000000000
                    1111100000000000
                    1111000000000000
                    1110000000000000
                    1100000000000000
                    1000000000000000

TEST DATA PATTERNS   Page C-3

                              C000000000000000

PATTERN 6   Words in pattern sequence - 16 (3-2-1-1-1)

              Sequence (octal) 000000, 000000, 000000, 177777,
                                 177777, 177777, 000000, 000000,
                                 177777, 177777, 000000, 177777,
                                 000000, 177777, 000000, 177777

              Sequence (binary) 0000000000000000
                                 0000000000000000
                                 0000000000000000
                                 1111111111111111
                                 1111111111111111
                                 1111111111111111
                                 0000000000000000
                                 0000000000000000
                                 1111111111111111
                                 1111111111111111
                                 0000000000000000
                                 1111111111111111
                                 0000000000000000
                                 1111111111111111
                                 0000000000000000
                                 1111111111111111

PATTERN 7   Words in pattern sequence - 1

              Sequence (octal) 133331

              Sequence (binary) 1011011011011001

PATTERN 8   Words in pattern sequence - 16 (3-2-1-1-1)

              Sequence (octal) 052525, 052525, 052525, 125252,
                                 125252, 125252, 052525, 052525,
                                 125252, 125252, 052525, 125252,
                                 052525, 125252, 052525, 125252

              Sequence (binary) 0101010101010101
                                 0101010101010101
                                 0101010101010101
                                 1010101010101010
                                 1010101010101010
                                 1010101010101010
                                 0101010101010101
                                 0101010101010101

TEST DATA PATTERNS   Page C-4

```
                              1010101010101010
                              1010101010101010
                              0101010101010101
                              1010101010101010
                              0101010101010101
                              1010101010101010
                              0101010101010101
                              1010101010101010
```

PATTERN 9   Words in pattern sequence - 1

              Sequence (octal) 155554

              Sequence (binary) 1101101101101100

PATTERN 10   Words in pattern sequence - 16 (3-2-1-1-1)

              Sequence (octal) 026455, 026455, 026455, 151322,
                                 151322, 151322, 026455, 026455,
                                 151322, 151322, 026455, 151322,
                                 026455, 151322, 026455, 151322

              Sequence (binary) 0010110100101101
                                 0010110100101101
                                 0010110100101101
                                 1101001011010010
                                 1101001011010010
                                 1101001011010010
                                 0010110100101101
                                 0010110100101101
                                 1101001011010010
                                 1101001011010010
                                 0010110100101101
                                 1101001011010010
                                 0010110100101101
                                 1101001011010010
                                 0010110100101101
                                 1101001011010010

PATTERN 11   Words in pattern sequence - 1

              Sequence (octal) 066666

              Sequence (binary) 0110110110110110

PATTERN 12   Words in pattern sequence - 16 (ripple one)
```

TEST DATA PATTERNS                                        Page C-5

Sequence  (octal)   000001, 000002, 000004, 000010,
                    000020, 000040, 000100, 000200,
                    000400, 001000, 002000, 004000,
                    010000, 020000, 040000, 100000

Sequence  (binary)  0000000000000001
                    0000000000000010
                    0000000000000100
                    0000000000001000
                    0000000000010000
                    0000000000100000
                    0000000001000000
                    0000000010000000
                    0000000100000000
                    0000001000000000
                    0000010000000000
                    0000100000000000
                    0001000000000000
                    0010000000000000
                    0100000000000000
                    1000000000000000

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  (binary)  1111111111111110
                    1111111111111101
                    1111111111111011
                    1111111111110111
                    1111111111101111
                    1111111111011111
                    1111111110111111
                    1111111101111111
                    1111111011111111
                    1111110111111111
                    1111101111111111
                    1111011111111111
                    1110111111111111
                    1101111111111111
                    1011111111111111
                    0111111111111111

PATTERN 14   Words in pattern sequence - 3

TEST DATA PATTERNS                                          Page C-6

Sequence  (octal)  155555,133333,155555

Sequence  (binary) 1101101101101101
                   1011011011011011
                   1101101101101101

PATTERN 15    Words in pattern sequence - 3

Sequence  (octal)  155555,133333,066666

Sequence  (binary) 1101101101101101
                   1011011011011011
                   0110110110110110

PATTERN 16    This is the operator selectable data pattern.
              See the TEST 2 SOFTWARE QUESTIONS section for more
              information.  As a default, when the program is
              started pattern 16 is defined at 1 word of zeroes.

Words in default pattern sequence - 1 (zeroes)

Sequence  (octal)  000000

Sequence  (binary) 0000000000000000

a

```
 1                              .NLIST BEX,CND
 2                              .SBTTL REVISION HISTORY
 3
 4                      ;++
 5                      ;       REVISION HISTORY:
 6                      ;       REV. AO - JFM - 2-OCT-85
 7                      ;               THIS PROGRAM IS A PDP-11 DISK SUBSYSTEM EXERCISER DESIGNED TO
 8                      ;               EXERCISE ANY KDA50-Q OR UDA50-A DISK SUBSYSTEM RESIDING ON A
 9                      ;               PDP-11 SYSTEM. THIS IS THE ORIGINAL VERSION OF THIS PROGRAM.
10                      ;--
11
12                      ;++
13                      ;       CONDITIONAL ASSEMBLY FLAGS
14                      ;--
15        000001        DEBUG=1                              ; 1=DEBUG MODE
16
17 000000              .ASECT
18                      ;       .ENABL  AMA                  ; THIS IS USED WITH LINKER
19 000000              .ENABL  ABS,AMA                       ; USE THIS WITHOUT LINKER
20        002000       .       =       2000                  ;
21
```

```
 23                              .SBTTL MACRO DEFINITION SECTION
 24
 25                      ;++
 26                      ;       PUSH - PUT DATA ON THE STACK            .
 27                      ;
 28                      ;       ARGUMENTS:
 29                      ;               A - DATA TO BE PUT ON THE STACK
 30                      ;--
 31
 32                      .MACRO  PUSH    A
 33                      .IRP    B,<A>
 34                              MOV     B,-(SP)         ; PUSH B ON STACK
 35                      .ENDM
 36                      .ENDM   PUSH
 37
 38                      ;++
 39                      ;       POP - REMOVE DATA FROM THE STACK
 40                      ;
 41                      ;       ARGUMENTS:
 42                      ;               A - LOCATION TO PUT THE DATA REMOVED FROM THE STACK
 43                      ;--
 44
 45                      .MACRO  POP     A
 46                      .IRP    B,<A>
 47                              MOV     (SP)+,B         ; POP STACK INTO B
 48                      .ENDM
 49                      .ENDM   POP
 50
 51                      ;++
 52                      ;       ASSUME - CHECK VALIDITY OF PROGRAM ASSUMPTIONS
 53                      ;--
 54                      .MACRO  ASSUME  FIRST,CONDITION,SECOND
 55                      .IF CONDITION <FIRST>-<SECOND>
 56                              .IFF
 57                              .ERROR  ;BAD ASSUME OF <FIRST> CONDITION <SECOND>
 58                      .ENDC
 59                      .ENDM   ASSUME
 60
 61                      ;++
 62                      ;       MACRO DEFINITIONS FOR GLOBAL EQUATES
 63                      ;
 64                      ;       THESE MACROS ARE USED TO DEFINE INDICES INTO A TABLE
 65                      ;
 66                      ;       CALLING SEQUENCE MUST BE
 67                      ;
 68                      ;               TABLE
 69                      ;               ITEM    NAME    BYTES   COMMENT
 70                      ;               ITEM    NAME    BYTES   COMMENT
 71                      ;               ITEM    NAME    BYTES   COMMENT
 72                      ;               END     SIZE
 73                      ;
 74                      ;       TABLE - DESIGNATES THAT A TABLE IS ABOUT TO BE DEFINED.
 75                      ;       END - TERMINATES THE DEFINITION.
 76                      ;       ITEM - ENTRY IN THE TABLE. ANY NUMBER OF ITEM LINES CAN APPEAR.
 77                      ;       NAME - THE NAME OF THE SYMBOL BEING EQUATED TO THE INDEX. THE INDEX
```

```
 78                              ;              ALWAYS STARTS AT ZERO.
 79                              ;       BYTES - THE SIZE OF THE VALUE TO BE STORED AT THAT INDEX IN BYTES.
 80                              ;       SIZE - (OPTIONAL) THE SIZE OF THE TABLE IN BYTES.
 81                              ;       TINDEX - KEEPS TRACK OF THE INDEX VALUE AND WILL BE EQUAL TO THE SIZE
 82                              ;              OF THE TABLE AFTER THE END STATEMENT.
 83                              ;--
 84
 85                              .MACRO  TABLE
 86                                      TINDEX = 0
 87                              .ENDM   TABLE
 88
 89                              .MACRO ITEM NAME BYTES COMMENT
 90                                      NAME=TINDEX                             ;COMMENT
 91                                      TINDEX=TINDEX+BYTES
 92                              .ENDM   ITEM
 93
 94                              .MACRO END SIZE COMMENT
 95                                      .IF NB SIZE
 96                                      SIZE=TINDEX                             ;COMMENT
 97                                      .ENDC
 98                              .ENDM   END
 99
```

```
101                                    .SBTTL  PROGRAM HEADER
102
103                                    .MCALL  SVC
104 002000                             SVC                                  ; INITIALIZE SUPERVISOR MACROS
105
106                            ;++
107                            ;       IF STRUCTURED MACROS ARE TO BE USED, ".MCALL STRUCT" AND "STRUCT"
108                            ;       MUST BE ADDED TO INITIALIZE THE STRUCTURED MACROS.
109
110        000000             SVCINS= 0                                    ; LIST INSTRUCTIONS, SHIFTED RIGHT
111        000000             SVCTST= 0                                    ; LIST TEST TAGS, SHIFTED RIGHT
112        000000             SVCSUB= 0                                    ; LIST SUBTEST TAGS, SHIFTED RIGHT
113        000000             SVCGBL= 0                                    ; LIST GLOBAL TAGS, SHIFTED RIGHT
114        000000             SVCTAG= 0                                    ; LIST OTHER TAGS, SHIFTED RIGHT
115
116                            ;       THE VALUES OF THE SVC... SYMBOLS ARE ZERO TO ALIGN THE MACRO CALLS
117                            ;       AND THEIR EXPANSIONS.  SETTING THE SYMBOLS TO BE MINUS-ONE WILL CAUSE
118                            ;       THE EXPANSIONS TO NOT BE LISTED.  THE SYMBOLS MAY BE CHANGED AT ANY
119                            ;       POINT IN THE PROGRAM.
120                            ;--
121
122
123                            ;++
124                            ;       THE PROGRAM HEADER IS THE INTERFACE BETWEEN
                               ;       THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
125                            ;
126                            ;       THE FOLLOWING ARE THE OPTIONAL 'HEADER' ARGUMENTS:
127                            ;
128                            ;       ARGUMENT          OPTION
129                            ;       --------          ------
130                            ;       BGNRPT            REPORT CODE
131                            ;       BGNSW             SOFTWARE TABLE
132                            ;       BGNSFT            SOFTWARE TABLE QUESTIONS
133                            ;       BGNAU             ADD UNIT CODE
134                            ;       BGNDU             DROP UNIT CODE
135                            ;       ERRTBL            ERROR TABLE
136                            ;       BGNSETUP          ASSEMBLED P-TABLES
137                            ;       ALL               ALL OF THE ABOVE
138                            ;       NONE              NONE OF THE ABOVE
139                            ;--
140
141 002000                            POINTER ALL
142
143
144                            ;++
145                            ;       THE "HEADER" ARGUMENTS ARE: NAME, REV, PATCH, LONGEST TEST
146                            ;       TIME, TYPE, PRIORITY, RESTORE, AND XM.
147                            ;       "TYPE" = 0 FOR SEQUENTIAL DIAGNOSTIC AND = 1 FOR EXERCISER.
148                            ;       "PRIORITY" INITIAL PROCESSOR PRIORITY OF THE DIAGNOSTIC (DEFAULT IS 0).
149                            ;       "RESTORE" = 0 TO RESTORE XM AND = 1 NOT TO RESTORE XM.
150                            ;       "XM" = 0 FOR SMALL MONITOR AND = 1 FOR EXTENDED MONITOR.
151                            ;--
152 002000                            HEADER  CZUDJ,A,0,0,1,PRIO7,1,1          ; TEST 5
    002000             L$NAME::                  ;DIAGNOSTIC NAME
    002000      103             .ASCII /C/
    002001      132             .ASCII /Z/
```

```
        002002      125                    .ASCII /U/
        002003      104                    .ASCII /D/
        002004      112                    .ASCII /J/
        002005      000                    .BYTE   0
        002006      000                    .BYTE   0
        002007      000                    .BYTE   0
        002010                     L$REV::                      ;REVISION LEVEL
        002010      101                    .ASCII  /A/
        002011                     L$DEPO::                     ;0
        002011      060                    .ASCII  /0/
        002012                     L$UNIT::                     ;NUMBER OF UNITS
        002012      000001                 .WORD   T$PTHV
        002014                     L$TIML::                     ;LONGEST TEST TIME
        002014      000000                 .WORD   0
        002016                     L$HPCP::                     ;POINTER TO H.W. QUES.
        002016      065214                 .WORD   L$HARD
        002020                     L$SPCP::                     ;POINTER TO S.W. QUES.
        002020      065344                 .WORD   L$SOFT
        002022                     L$HPTP::                     ;PTR. TO DEF. H.W. PTABLE
        002022      002134                 .WORD   L$HW
        002024                     L$SPTP::                     ;PTR. TO S.W. PTABLE
        002024      002144                 .WORD   L$SW
        002026                     L$LADP::                     ;DIAG. END ADDRESS
        002026      066462                 .WORD   L$LAST
        002030                     L$STA::                      ;RESERVED FOR APT STATS
        002030      000000                 .WORD   0
        002032                     L$CO::
        002032      000000                 .WORD   0
        002034                     L$DTYP::                     ;DIAGNOSTIC TYPE
        002034      000001                 .WORD   1
        002036                     L$APT::                      ;APT EXPANSION
        002036      000000                 .WORD   0
        002040                     L$DTP::                      ;PTR. TO DISPATCH TABLE
        002040      002124                 .WORD   L$DISPATCH
        002042                     L$PRIO::                     ;DIAGNOSTIC RUN PRIORITY
        002042      000340                 .WORD   PRIO7
        002044                     L$ENVI::                     ;FLAGS DESCRIBE HOW IT WAS SETUP
        002044      000000                 .WORD   0
        002046                     L$EXP1::                     ;EXPANSION WORD
        002046      000000                 .WORD   0
        002050                     L$MREV::                     ;SVC REV AND EDIT #
        002050      004                    .BYTE   C$REVISION
        002051      001                    .BYTE   C$EDIT
        002052                     L$EF::                       ;DIAG. EVENT FLAGS
        002052      000000                 .WORD   0
        002054      000000                 .WORD   0
        002056                     L$SPC::
        002056      000000                 .WORD   0
        002060                     L$DEVP::                     ; POINTER TO DEVICE TYPE LIST
        002060      007142                 .WORD   L$DVTYP
        002062                     L$REPP::                     ;PTR. TO REPORT CODE
        002062      054534                 .WORD   L$RPT
        002064                     L$EXP4::
        002064      000000                 .WORD   0
        002066                     L$EXP5::
```

```
          002066   000000              .WCRD    0
          002070                L$AUT::                      ;PTR. TO ADD UNIT CODE
          002070   062634              .WORD    L$AU
          002072                L$DUT::                      ;PTR. TO DROP UNIT CODE
          002072   062632              .WORD    L$DU
          002074                L$LUN::                      ;LUN FOR EXERCISERS TO FILL
          002074   000000              .WORD    0
          002076                L$DESP::                     ;POINTER TO DIAG. DESCRIPTION
          002076   007162              .WORD    L$DESC
          002100                L$LOAD::                     ;GENERATE SPECIAL AUTOLOAD EMT
          002100   104035              EMT      E$LOAD
          002102                L$ETP::                      ;POINTER TO ERRTBL
          002102   002156              .WORD    L$ERRTBL
          002104                L$ICP::                      ;PTR. TO INIT CODE
          002104   060364              .WORD    L$INIT
          002106                L$CCP::                      ;PTR. TO CLEAN-UP CODE
          002106   062612              .WORD    L$CLEAN
          002110                L$ACP::                      ;PTR. TO AUTO CODE
          002110   062610              .WORD    L$AUTO
          002112                L$PRT::                      ;PTR. TO PROTECT TABLE
          002112   060356              .WORD    L$PROT
          002114                L$TEST::                     ; TEST NUMBER
          002114   000003              .WORD    3
          002116                L$DLY::                      ;DELAY COUNT
          002116   000000              .WORD    0
          002120                L$HIME::                     ;PTR. TO HIGH MEM
          002120   000000              .WORD    0
     153
```

```
155                                    .SBTTL  DISPATCH TABLE
156
157                                 ;++
158                                 ;       THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
159                                 ;       IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST. THE
160                                 ;       ARGUMENT OF 'DISPATCH' INDICATES THE NUMBER OF HARDWARE TESTS
161                                 ;       IN THE DIAGNOSTIC.
162                                 ;--
163
164  002122                                 DISPATCH 3
     002122  000003                          .WORD   3
     002124                         L$DISPATCH::
     002124  062636                          .WORD   T1
     002126  063212                          .WORD   T2
     002130  063304                          .WORD   T3
165
```

```
167                                    .SBTTL   DEFAULT HARDWARE P-TABLE
168
169                           ;++
170                           ;        THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
171                           ;        THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
172                           ;        IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
173                           ;        AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
174                           ;        THE ACTUAL P-TABLE BUILT AT RUN TIME IS STORED IN SUPERVISOR
175                           ;        SPACE.
176                           ;--
177
178  002132                            BGNHW    DFPTBL
     002132   000003                   .WORD    L10000-L$HW/2
     002134            L$HW::
     002134            DFPTBL::
179
180  002134   172150                   .WORD    172150              ; CONTROLLER CSR ADDRESS
181  002136   000000                   .WORD    0.                  ; DRIVE PLUG NUMBER
182  002140   000000                   .WORD    0.                  ; WRITE ON DRIVE (0=NO)
183
184  002142                            ENDHW
     002142            L10000:
```

```
     186                              .SBTTL   SOFTWARE P-TABLE
     187
     188                          ;++
     189                          ;        THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
     190                          ;        PROGRAM AS OPERATIONAL PARAMETERS.   THESE PARAMETERS ARE
     191                          ;        SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
     192                          ;        AT RUN TIME. THIS TABLE, UNLIKE THE HARDWARE TABLE, WILL CONTAIN
     193                          ;        THE ACTUAL VALUES ENTERED BY THE OPERATOR.
     194                          ;--
     195
     196 002142                       BGNSW    SFPTBL
         002142   000005              .WORD    L10001-L$SW/2
         002144               L$SW::
         002144               SFPTBL::
     197
     198 002144   000001              .WORD    1.                      ; HARD ERROR LIMIT
     199 002146   000074              .WORD    60.                     ; EXERCISER TIME LIMIT
     200 002150   000017              .WORD    15.                     ; MINUTES BETWEEN REPORTS
     201 002152   000010              .WORD    8.                      ; MAXIMUM TRANSFER SIZE IN BLOCKS
     202 002154   000764              .WORD    ^B0000000111110100      ; SINGLE BIT QUESTIONS
     203
     204 002156                       ENDSW
         002156               L10001:
     205
```

```
207                              .SBTTL  GLOBAL EQUATES SECTION
208
209                              ;++
210                              ;       THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
211                              ;       ARE USED IN MORE THAN ONE TEST.
212                              ;--
213
214 002156                      EQUALS
                                 ;
                                 ; 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

        001000                   BIT9==  BIT09
        000400                   BIT8==  BIT08
        000200                   BIT7==  BIT07
        000100                   BIT6==  BIT06
        000040                   BIT5==  BIT05
        000020                   BIT4==  BIT04
        000010                   BIT3==  BIT03
        000004                   BIT2==  BIT02
        000002                   BIT1==  BIT01
        000001                   BIT0==  BIT00
                                 ;
                                 ; EVENT FLAG DEFINITIONS
                                 ;  EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION
                                 ;
                                 ;                               ; BIT POSITION IN SECOND STATUS WORD
        000040                   EF.START==      32.             ; (100000) START COMMAND WAS ISSUED
        000037                   EF.RESTART==    31.             ; (040000) RESTART COMMAND WAS ISSUED
        000036                   EF.CONTINUE==   30.             ; (020000) CONTINUE COMMAND WAS ISSUED
        000035                   EF.NEW==        29.             ; (010000) A NEW PASS HAS BEEN STARTED
        000034                   EF.PWR==        28.             ; (004000) A POWER-FAIL/POWER-UP OCCURRED
        000033                   EF.XM==         27.             ; (002000) DIAG IS GOOD OF EXTENDED ENVIROMENT
                                 ;
                                 ;
                                 ; PRIORITY LEVEL DEFINITIONS
                                 ;
        000340                   PRIO7== 340
        000300                   PRIO6== 300
```

```
        000240                  PRIO5== 240
        000200                  PRIO4== 200
        000140                  PRIO3== 140
        000100                  PRIO2== 100
        000040                  PRIO1== 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                  IXE==    4000
        010000                  IBE==   10000
        020000                  IER==   20000
        040000                  LOE==   40000
        100000                  HOE==  100000
220
```

```
222                                    .SBTTL  CONTROLLER BIT DEFINITIONS
223
224                            ;
225                            ;          SA REGISTER UNIVERSAL READ BITS
226                            ;
227        100000              SA.ERR  = 100000                      ; ERROR INDICATOR
228        040000              SA.S4   = 040000                      ; STEP 4 STATUS BIT
229        020000              SA.S3   = 020000                      ; STEP 3 STATUS BIT
230        010000              SA.S2   = 010000                      ; STEP 2 STATUS BIT
231        004000              SA.S1   = 004000                      ; STEP 1 STATUS BIT
232
233                            ;          SA REGISTER ERROR STATUS BITS
234
235        003777              SA.ERC  = 003777                      ; ERROR CODE
236
237                            ;          SA REGISTER STEP 1 SEND BITS
238
239        000177              SA.VEC  = 000177                      ; INTERRUPT VECTOR (DIVIDED BY 4)
240        000200              SA.INT  = 000200                      ; INTERRUPT ENABLE DURING INIT
241        003400              SA.MSG  = 003400                      ; MESSAGE RING LENGTH
242        034000              SA.CMD  = 034000                      ; COMMAND RING LENGTH
243        040000              SA.WRP  = 040000                      ; WRAP BIT
244        100000              SA.STP  = 100000                      ; STEP - MUST ALWAYS BE WRITTEN A ONE
245
246        000400              SA.MS1  = 000400                      ; LSB OF MESSAGE RING LENGTH
247        004000              SA.CM1  = 004000                      ; LSB OF COMMAND RING LENGTH
248                            ;
249                            ;          SA REGISTER STEP 1 RESPONSE BITS
250
251        002000              SA.NV   = 002000                      ; NON SETTABLE INTERRUPT VECTOR
252        001000              SA.QB   = 001000                      ; 22 BIT ADDRESS BUS
253        000400              SA.DI   = 000400                      ; ENHANCED DIAGNOSTICS
254        000100              SA.MP   = 000100                      ; MAPPING BIT
255        000040              SA.SM   = 000040                      ; SPECIAL MODE BIT FOR KDA50-Q
256                                      000377                      ; THESE BITS RESERVED
257                            ;
258                            ;          SA REGISTER STEP 2 SEND BITS
259                            ;
260        000001              SA.PRG  = 000001                      ; ENABLE VAX UBA PURGE INTERRUPT
261                            ;          177776                      ; LOW ORDER MESSAGE RING BYTE ADDRESS
262                            ;
263                            ;          SA REGISTER STEP 2 RESPONSE BITS
264
265        000007              SA.MSE  = 000007                      ; MESSAGE RING LENGTH ECHO
266        000070              SA.CME  = 000070                      ; COMMAND RING LENGTH ECHO
267                                      000100                      ; RESERVED
268        000200              SA.STE  = 000200                      ; STEP ECHO
269        003400              SA.CTP  = 003400                      ; CONTROLLER TYPE
270                            ;
271                            ;          SA REGISTER STEP 3 SEND BITS
272                            ;
273                                      077777                      ; HIGH ORDER MESSAGE RING BYTE ADDRESS
274        100000              SA.TST  = 100000                      ; PURGE POLL TEST ENABLE
275                            ;
276                            ;          SA REGISTER STEP 3 RESPONSE BITS
```

```
277          ;
278  000177         SA.VCE = 000177                    ; INTERRUPT VECTOR ECHO
279  000200         SA.INE = 000200                    ; INTERRUPT ENABLE ECHO
280  000400         SA.NVE = 000400                    ; VECTOR NOT PROGRAMMABLE
281                 ;        003000                    ; RESERVED
282          ;
283          ;              SA REGISTER STEP 4 SEND BITS
284          ;
285  000001         SA.GO  = 000001                    ; GO BIT TO START CONTROLLER FIRMWARE
286  000002         SA.LFC = 000002                    ; LAST FAILURE CODE REQUEST
287  000374         SA.BST = 000374                    ; BURST LEVEL
288          ;
289          ;              SA.REGISTER STEP 4 RESPONSE BITS
290          ;
291  000017         SA.MCV = 000017                    ; CONTROLLER MICROCODE VERSION
292  000360         SA.CNT = 000360                    ; CONTROLLER TYPE
293                 ;        003400                    ; RESERVED
```

```
295                                     .SBTTL  HOST COMMUNICATION AREA DEFINIIONS
296
297                                     ;
298                                     ;       COMMAND/MESSAGE RING BIT DEFINITIONS
299                                     ;
300     100000                  RG.OWN  = 100000                          ; SET WHEN CONTROLLER OWNS RING
301     040000                  RG.FLG  = 040000                          ; FLAG BIT
302                                     ;
303                                     ;       CONNECTION ID & MESSAGE TYPE NUMBERS
304                                     ;
305     000000                  DISK    = 0                               ; 0 - DISK CIRCUIT, SEQUENTIAL MSG
306     000400                  TAPE    = 400                             ; 1 - TAPE CIRCUIT, SEQUENTIAL MSG
307     001000                  DUP     = 1000                            ; 2 - DUP CIRCUIT, SEQUENTIAL MSG
308     177760                  MAINT   = 177760                          ; 177760 - MAINT. CIRCUIT, MAINT. MSG
```

```
310                                    .SBTTL  COMMAND PACKET OPCODES DEFINITIONS
311
312
313                                    ;++
314                                    ;       NOTE: END PACKET OPCODES (ALSO CALLED ENDCODES) ARE FORMED BY ADDING
315                                    ;       THE END PACKET FLAG TO THE COMMAND OPCODE. FOR EXAMPLE, A READ
316                                    ;       COMMAND'S END PACKET CONTAINS THE VALUE OP.RD+OP.END IN ITS OPCODE
317                                    ;       FIELD. THE INVALID COMMAND END PACKET CONTAINS JUST THE END PACKET
318                                    ;       FLAG (I.E., OP.END) IN ITS OPCODE FIELD. THE SERIOUS EXCEPTION END
319                                    ;       PACKET CONTAINS THE SUM OF THE END PACKET FLAG PLUS THE SERIOUS
320                                    ;       EXCEPTION OPCODE SHOWN ABOVE (I.E., OP.SEX+OP.END) IN ITS OPCODE FIELD.
321
322                                    ;       COMMAND OPCODE BITS 3 THROUGH 5 INDICATE THE COMMAND CLASS, WHICH IS
323                                    ;       ENCODED AS FOLLOWS:
324                                    ;       000   IMMEDIATE COMMANDS
325                                    ;       001   SEQUENTIAL COMMANDS
326                                    ;       010   NON-SEQUENTIAL COMMANDS THAT DO NOT INCLUDE A BUFFER DESCRIPTOR
327                                    ;       100   NON-SEQUENTIAL COMMANDS THAT DO INCLUDE A BUFFER DESCRIPTOR
328                                    ;--
329
330       000001                       OP.ABO = 01                            ; ABORT COMMAND
331       000020                       OP.ACC = 20                            ; ACCESS COMMAND
332       000010                       OP.AVL = 10                            ; AVAILABLE COMMAND
333       000021                       OP.CCD = 21                            ; COMPARE CONTROLLER DATA COMMAND
334       000040                       OP.CMP = 40                            ; COMPARE HOST DATA COMMAND
335       000013                       OP.DAP = 13                            ; DETERMINE ACCESS PATHS COMMAND
336       000022                       OP.ERS = 22                            ; ERASE COMMAND
337       000023                       OP.FLU = 23                            ; FLUSH COMMAND
338       000002                       OP.GCS = 02                            ; GET COMMAND STATUS COMMAND
339       000003                       OP.GUS = 03                            ; GET UNIT STATUS COMMAND
340       000011                       OP.ONL = 11                            ; ONLINE COMMAND
341       000041                       OP.RD  = 41                            ; READ COMMAND
342       000024                       OP.RPL = 24                            ; REPLACE COMMAND
343       000004                       OP.SCC = 04                            ; SET CONTROLLER CHARACTERISTICS COMMAND
344       000012                       OP.SUC = 12                            ; SET UNIT CHARACTERISTICS COMMAND
345       000042                       OP.WR  = 42                            ; WRITE COMMAND
346       000030                       OP.MRD = 30                            ; MAINTENANCE READ COMMAND
347       000031                       OP.MWR = 31                            ; MAINTENANCE WRITE COMMAND
348       000200                       OP.END = 200                           ; END PACKET FLAG
349       000007                       OP.SEX = 7                             ; SERIOUS EXCEPTION END PACKET
350       000100                       OP.AVA = 100                           ; AVAILABLE ATTENTION MESSAGE
351       000101                       OP.DUP = 101                           ; DUPLICATE UNIT NUMBER ATTENTION MESSAGE
352       000102                       OP.SHC = 102                           ; SHADOW COPY COMPLETE ATTENION MESSAGE
353                                    ;
354                                    ;              COMMAND MODIFIERS
355                                    ;
356       020000                       MD.CSE = 020000                        ; CLEAR SERIOUS EXCEPTION
357       040000                       MD.CMP = 040000                        ; COMPARE
358       100000                       MD.EXP = 100000                        ; EXPRESS REQUEST
359       010000                       MD.ERR = 010000                        ; FORCE ERROR
360       004000                       MD.SCH = 004000                        ; SUPPRESS CACHING (HIGH SPEED)
361       002000                       MD.SCL = 002000                        ; SUPPRESS CACHING (LOW SPEED)
362       000100                       MD.SEC = 000100                        ; SUPPRESS ERROR CORRECTION
363       000400                       MD.SER = 000400                        ; SUPPRESS ERROR RECOVERY
364       000200                       MD.SSH = 000200                        ; SUPPRESS SHADOWING
          000100                       MD.WBN = 000100                        ; WRITE-BACK (NON-VOLATILE)
```

```
365        000040              MD.WBV = 000040              ; WRITE BACK (VOLATILE)
366        000020              MD.SEQ = 000020              ; WRITE SHADOW SET ONE UNIT AT A TIME
367        000002              MD.ALL = 000002              ; ALL CLASS DRIVERS
368        000001              MD.SPD = 000001              ; SPIN-DOWN
369        000001              MD.FEU = 000001              ; FLUSH ENTIRE UNIT
370        000002              MD.VOL = 000002              ; VOLATILE ONLY
371        000001              MD.NXU = 000001              ; NEXT UNIT
372        000001              MD.RIP = 000001              ; ALLOW SELF DESTRUCTION
373        000002              MD.IMF = 000002              ; IGNORE MEDIA FORMAT ERROR
374        000010              MD.CWB = 000010              ; CLEAR WRITE-BACK DATA LOST
375        000004              MD.SWP = 000004              ; SET WRITE PROTECT
376        000020              MD.SHD = 000020              ; SHADOW UNIT SPECIFIED
377        000001              MD.PRI = 000001              ; PRIMARY REPLACEMENT BLOCK
378                            ;
379                            ;          END PACKET FLAGS
380                            ;
381        000200              EF.BBR = 000200              ; BAD BLOCK REPORTED
382        000100              EF.BBU = 000100              ; BAD BLOCK UNREPORTED
383        000040              EF.LOG = 000040              ; ERROR LOG GENERATED
384        000020              EF.SEX = 000020              ; SERIOUS EXCEPTION
385                            ;
386                            ;          CONTROLLER FLAGS
387                            ;
388        000200              CF.ATN = 000200              ; ENABLE ATTENTION MESSAGES
389        000100              CF.MSC = 000100              ; ENABLE MISC. ERROR LOG MESSAGES
390        000040              CF.OTH = 000040              ; ENABLE OTHER HOST'S ERROR LOG MESSAGES
391        000020              CF.THS = 000020              ; ENABLE THIS HOST'S ERROR LOG MESSAGES
392        100000              CF.RPL = 100000              ; CNTRLR INITIATED BAD BLOCK REPLACEMENT
393        000002              CF.SHD = 000002              ; SHADOWING
394        000001              CF.576 = 000001              ; 576 BYTE SECTORS
395                            ;
396                            ;          UNIT FLAGS
397                            ;
398        000001              UF.CMR = 000001              ; COMPARE READS
399        000002              UF.CMW = 000002              ; COMPARE WRITES
400        100000              UF.RPL = 100000              ; CNTRLR INITIATED BAD BLOCK REPLACEMENT
401        040000              UF.INA = 040000              ; INACTIVE SHADOW SET UNIT
402        000200              UF.RMV = 000200              ; REMOVABLE MEDIA
403        004000              UF.SCH = 004000              ; SUPPRESS CACHING (HIGH SPEED)
404        002000              UF.SCL = 002000              ; SUPPRESS CACHING (LOW SPEED)
405        000100              UF.WBN = 000100              ; WRITE-BACK (NON-VOLATILE)
406        020000              UF.WPH = 020000              ; WRITE PROTECT (HARDWARE)
407        010000              UF.WPS = 010000              ; WRITE PROTECT (SOFTWARE OR VOLUME)
408        000004              UF.576 = 000004              ; 576 BYTE SECTORS
```

```
410                                    .SBTTL  COMMAND PACKET OFFSETS
411
412                            ;
413                            ;        GENERIC COMMAND PACKET OFFSETS
414                            ;
415        000000             P.CRF  = 0.                          ; COMMAND REFERENCE NUMBER
416        000004             P.UNIT = 4.                          ; UNIT NUMBER
417        000010             P.OPCD = 8.                          ; OPCODE
418        000012             P.MOD  = 10.                         ; MODIFIERS
419        000014             P.BCNT = 12.                         ; BYTE COUNT
420        000020             P.BUFF = 16.                         ; BUFFER DESCRIPTOR
421        000020             P.ADBA = 16.                         ;   BASE ADDRESS
422        000022             P.ADEA = 18.                         ;   EXTENDED ADDRESS
423        000034             P.LBN  = 28.                         ; LOGICAL BLOCK NUMBER
424                            ;
425                            ;        ABORT AND GET COMMAND STATUS COMMAND PACKET OFFSETS
426                            ;
427        000014             P.OTRF = 12.                         ; OUTSTANDING REFERENCE NUMBER
428                            ;
429                            ;        ONLINE AND SET UNIT CHARACTERISTICS COMMAND PACKET OFFSETS
430                            ;
431        000016             P.UNFL = 14.                         ; UNIT FLAGS
432        000034             P.DVPM = 28.                         ; DEVICE DEPENDENT PARAMETERS
433        000040             P.SHUN = 32.                         ; SHADOW UNIT
434        000042             P.CPSP = 34.                         ; COPY SPEED
435                            ;
436                            ;        REPLACE COMMAND PACKET OFFSETS
437                            ;
438        000014             P.RBN  = 12.                         ; REPLACEMENT BLOCK NUMBER
439                            ;
440                            ;        SET CONTROLLER CHARACTERISTICS COMMAND PACKET OFFSETS
441                            ;
442        000014             P.VRSN = 12.                         ; MSCP VERSION
443        000016             P.CNTF = 14.                         ; CONTROLLER FLAGS
444        000020             P.HTMO = 16.                         ; HOST TIMEOUT
445        000024             P.TIME = 20.                         ; QUAD-WORD TIME AND DATE
446                            ;
447                            ;        MAINTENANCE READ AND MAINTENANCE WRITE COMMAND PACKET OFFSETS
448                            ;
449        000034             P.RGID =        28.                  ; REGION ID
450        000040             P.RGOF =        32.                  ; REGION OFFSET
```

```
452                                    .SBTTL END PACKET OFFSETS
453
454                             ;
455                             ;          GENERIC END PACKET OFFSETS
456                             ;
457        000000              P.CRF   = 0.                          ; COMMAND REFERENCE NUMBER
458        000004              P.UNIT  = 4.                          ; UNIT NUMBER
459        000010              P.OPCD  = 8.                          ; OPCODE (ALSO CALLED ENDCODE)
460        000011              P.FLGS  = 9.                          ; END PACKET FLAGS
461        000012              P.STS   = 10.                         ; STATUS
462        000014              P.BCNT  = 12.                         ; BYTE COUNT
463        000034              P.FBBK  = 28.                         ; FIRST BAD BLOCK
464                             ;
465                             ;      ABORT AND GET COMMAND STATUS END PACKET OFFSETS
466                             ;
467        000014              P.OTRF  = 12.                         ; OUTSTANDING REFERENCE NUMBER
468        000020              P.CMST  = 16.                         ; COMMAND STATUS
469                             ;
470                             ;          GET UNIT STATUS END PACKET OFFSETS
471                             ;
472        000014              P.MLUN  = 12.                         ; MULTI-UNIT CODE
473        000016              P.UNFL  = 14.                         ; UNIT FLAGS
474        000024              P.UNTI  = 20.                         ; UNIT IDENTIFIER
475        000034              P.MEDI  = 28.                         ; MEDIA TYPE IDENTIFIER
476        000040              P.SHUN  = 32.                         ; SHADOW UNIT
477        000042              P.SHST  = 34.                         ; SHADOW STATUS
478        000044              P.TRKS  = 36.                         ; TRACK SIZE
479        000046              P.GRPS  = 38.                         ; GROUP SIZE
480        000050              P.CYLS  = 40.                         ; CYLINDER SIZE
481        000052              P.USVR  = 42.                         ; UNIT SOFTWARE VERSION
482        000053              P.UHVR  = 43.                         ; UNIT HARDWARE VERSION
483        000054              P.RCTS  = 44.                         ; RCT TABLE SIZE
484        000056              P.RBNS  = 46.                         ; RBNS / TRACK
485        000057              P.RCTC  = 47.                         ; RCT COPIES
486
487                             ;      ONLINE AND SET UNIT CHARACTERISTICS END PACKET AND AVAILABLE
488                             ;      ATTENTION MESSAGE OFFSETS
489                             ;
490        000014              P.MLUN  = 12.                         ; MULTI-UNIT CODE
491        000016              P.UNFL  = 14.                         ; UNIT FLAGS
492        000024              P.UNTI  = 20.                         ; UNIT IDENTIFIER
493        000034              P.MEDI  = 28.                         ; MEDIA TYPE IDENTIFIER
494        000040              P.SHUN  = 32.                         ; SHADOW UNIT
495        000042              P.SHST  = 34.                         ; SHADOW STATUS
496        000044              P.UNSZ  = 36.                         ; UNIT SIZE
497        000050              P.VSER  = 40.                         ; VOLUME SERIAL NUMBER
498                             ;
499                             ;      SET CONTROLLER CHARACTERISTICS END PACKET OFFSETS
500                             ;
501        000014              P.VRSN  = 12.                         ; MSCP VERSION
502        000016              P.CNTF  = 14.                         ; CONTROLLER FLAGS
503        000020              P.CTMO  = 16.                         ; CONTROLLER TIMEOUT
504        000022              P.CSVR  = 18.                         ; CONTROLLER SOFTWARE VERSION
505        000023              P.CHVR  = 19.                         ; CONTROLLER HARDWARE VERSION
506        000024              P.CNTI  = 20.                         ; CONTROLLER ID
```

```
507                                    ;
508                                    ;         STATUS AND EVENT CODE DEFINITIONS
509                                    ;
510         000037              ST.MSK  = 37              ; STATUS / EVENT CODE MASK
511         000040              ST.SUB  = 40              ; SUB-CODE MULTIPLIER
512         000000              ST.SUC  = 0               ; SUCCESS
513         000001              ST.CMD  = 1               ; INVALID COMMAND
514         000002              ST.ABO  = 2               ; COMMAND ABORTED
515         000003              ST.OFL  = 3               ; UNIT-OFFLINE
516         000004              ST.AVL  = 4               ; UNIT-AVAILABLE
517         000005              ST.MFE  = 5               ; MEDIA FORMAT ERROR
518         000006              ST.WPR  = 6               ; WRITE PROTECTED
519         000007              ST.CMP  = 7               ; COMPARE ERROR
520         000010              ST.DAT  = 10              ; DATA ERROR
521         000011              ST.HST  = 11              ; HOST BUFFER ACCESS ERROR
522         000012              ST.CNT  = 12              ; CONTROLLER ERROR
523         000013              ST.DRV  = 13              ; DRIVE ERROR
524         000037              ST.DIA  = 37              ; MESSAGE FROM AN INTERNAL DIAGNOSTIC
```

```
526                                      .SBTTL ERROR LOG MESSAGE OFFSETS
527
528
529                                  ;            ERROR LOG MESSAGE OFFSETS
530
531          000000          L.CRF   =          0.              ; COMMAND REFERENCE NUMBER
532          000004          L.UNIT  =          4.              ; UNIT NUMBER
533          000006          L.SEQ   =          6.              ; SEQUENCE NUMBER
534          000010          L.FMT   =          8.              ; FORMAT
535          000011          L.FLGS  =          9.              ; ERROR LOG MESSAGE FLAGS
536          000012          L.EVNT  =          10.             ; EVENT CODE
537          000014          L.CNTI  =          12.             ; CONTROLLER ID
538          000024          L.CSVR  =          20.             ; CONTROLLER SOFTWARE VERSION
539          000025          L.CHVR  =          21.             ; CONTROLLER HARDWARE VERSION
540          000026          L.MLUN  =          22.             ; MULTI-UNIT CODE
541          000030          L.UNTI  =          24.             ; UNIT ID
542          000030          L.BADR  =          24.             ; BUS ADDRESS
543          000040          L.USVR  =          32.             ; UNIT SOFTWARE VERSION
544          000041          L.UHVR  =          33.             ; UNIT HARDWARE VERSION
545          000042          L.LVL   =          34.             ; LEVEL
546          000042          L.SCYL  =          34.             ; CYLINDER
547          000043          L.RTRY  =          35.             ; RETRY
548          000044          L.VSER  =          36.             ; VOLUME SERIAL NUMBER
549          000050          L.HDCD  =          40.             ; HEADER CODE
550          000054          L.SDI   =          44.             ; SDI INFORMATION
551
552                                  ;            ERROR LOG FLAGS
553
554          000200          LF.SUC  =          000200          ; OPERATION SUCCESSFUL
555          000100          LF.CON  =          000100          ; OPERATION CONTINUING
556          000001          LF.SNR  =          000001          ; SEQUENCE NUMBER RESET
557                                  ;
558                                  ;            SUBCODES FOR ST.SUC
559
560          000040          SC.SPI  =          40              ; SPIN-DOWN IGNORED
561          000100          SC.STC  =          100             ; STILL CONNECTED
562          000200          SC.DUP  =          200             ; DUPLICATE UNIT NUMBER
563          000400          SC.AOL  =          400             ; ALREADY ONLINE
564          001000          SC.SOL  =          1000            ; STILL ONLINE
565          010000          SC.ROV  =          10000           ; READ ONLY (VOLUME FORMAT)
566
567                                  ;            SUBCODES FOR ST.OFL
568
569          000000          SC.UNK = 0                         ; UNIT UNKNOWN OR ONLINE OTHER CTRLR
570          000040          SC.NVL = 40                        ; NO VOLUME MOUNTED OR DRIVE DISABLED
571          000100          SC.IOP = 100                       ; UNIT INOPERATIVE
572          000400          SC.DIS = 400                       ; UNIT DISABLED BY FIELD SERVICE
573          000200          SC.DUP = 200                       ; DUPLICATE UNIT NUMBER
574                                  ;
575                                  ;            SUBCODES FOR ST.MFE
576
577          000240          SC.576  =          240             ; DISK FORMATTED WITH 576 BYTE SECTORS
578          000300          SC.FCT  =          300             ; FCT CORRUPT
579          000400          SC.RCT  =          400             ; RCT CORRUPT
580          000440          SC.RBN  =          440             ; NO RBN AVAILABLE
```

```
581
582                              ;          SUBCODES FOR ST.WPR
583                              ;
584        020000                SC.HWP  =        20000           ; UNIT IS HARDWARE WRITE PROTECTED
585        010000                SC.SWP  =        10000           ; UNIT IS SOFTWARE WRITE PROTECTED
586
587                              ;          SUBCODES FOR ST.DAT
588                              ;
589        000000                SC.FOR  =        0               ; FORCED ERROR
590        000100                SC.HDR  =        100             ; INVALID HEADER
591        000140                SC.DSY  =        140             ; DATA SYNC NOT FOUND
592        000200                SC.CEC  =        200             ; CORRECTABLE ECC ERROR
593        000340                SC.UEC  =        340             ; UNCORRECTABLE ECC ERROR
594        000400                SC.1EC  =        400             ; 1 SYMBOL ECC ERROR
595        000440                SC.2EC  =        440             ; 2 SYMBOL ECC ERROR
596        000500                SC.3EC  =        500             ; 3 SYMBOL ECC ERROR
597        000540                SC.4EC  =        540             ; 4 SYMBOL ECC ERROR
598        000600                SC.5EC  =        600             ; 5 SYMBOL ECC ERROR
599        000640                SC.6EC  =        640             ; 6 SYMBOL ECC ERROR
600        000700                SC.7EC  =        700             ; 7 SYMBOL ECC ERROR
601        000740                SC.8EC  =        740             ; 8 SYMBOL ECC ERROR
602                              ;
603                              ;          SUBCODES FOR ST.HST
604                              ;
605        000040                SC.OTA  =        40              ; ODD TRANSFER ADDRESS
606        000100                SC.OBC  =        100             ; ODD BYTE COUNT
607        000140                SC.NXM  =        140             ; NON-EXISTANT MEMORY
608        000200                SC.HMP  =        200             ; HOST MEMORY PARITY ERROR
609        000240                SC.IPT  =        240             ; INVALID PAGE TABLE ENTRY
610                              ;
611                              ;          SUBCODES FOR ST.CNT
612                              ;
613        000040                SC.SDS  =        40              ; SERDES OVERRUN OR UNDERRUN
614        000100                SC.EDC  =        100             ; EDC ERROR
615        000140                SC.IIC  =        140             ; INCONSISTANT INTERNAL CONTROL
616        000200                SC.IEE  =        200             ; INTERNAL EDC ERROR
617        000240                SC.LPI  =        240             ; LESI ADAPTOR CARD PARITY ERROR INPUT
618        000300                SC.LPO  =        300             ; LESI ADAPTOR CARD PARITY ERROR,OUTPUT
619        000340                SC.CIP  =        340             ; "CABLE IN PLACE" NOT ASSERTED
620        000400                SC.COU  =        400             ; CONTROLLER OVERRUN OR UNDERRUN
621        000440                SC.CME  =        440             ; CONTROLLER MEMORY ERROR
622                              ;
623                              ;          SUBCODES FOR ST.DRV
624                              ;
625        000040                SC.CTO  =        40              ; DRIVE COMMAND TIME OUT
626        000100                SC.XME  =        100             ; CONTROLLER DETECTED XMISSION ERROR
627        000140                SC.POE  =        140             ; POSITIONER ERROR
628        000200                SC.LRW  =        200             ; LOST RESD/WRITE READY
629        000240                SC.DCD  =        240             ; DRIVE CLOCK DROPOUT
630        000300                SC.LRR  =        300             ; LOST RECEIVER READY
631        000340                SC.DDE  =        340             ; DRIVE DETECTED ERROR
632        000400                SC.PSE  =        400             ; PULSE OR STATE PARITY ERROR
633        000500                SC.PRO  =        500             ; CONTROLLER DETECTED PROTOCOL ERROR
634        000550                SC.DFI  =        550             ; DRIVE FAILED INITIALIZATION
635        000600                SC.DII  =        600             ; DRIVE IGNORED INITIALIZATION
```

636   000640   SC.RRC =   640   ; RECEIVER READY COLLISION

```
638                                   .SBTTL   CONTROLLER TABLE DEFINITIONS
639
640                            ;++
641                            ;        CONTROLLER TABLE DEFINITIONS
642                            ;
643                            ;        ONE TABLE WILL BE SET UP BY INITIALIZATION SECTION FOR EACH CONTROLLER
644                            ;        SELECTED FOR TESTING.  TABLES ARE CONTIGUOUS.  THE END OF THE TABLES IS
645                            ;        MARKED BY A WORD OF ZEROS.
646                            ;
647                            ;        THE FIRST TABLE IS POINTED TO BY THE CONTENTS OF CTABS.
648                            ;        THE NUMBER OF TABLES IS CONTAINED IN CTRLRS.
649                            ;
650                            ;        BELOW IS A DESCRIPTION OF THE HOST COMMUNICATIONS AREA WITH ONE
651                            ;        DESCRIPTOR IN EACH RING. THE DESCRIPTION OF THE COMMAND PACKETS
652                            ;        IS CONTAINED IN THE MESSAGE ENVELOPE BUFFER DESCRITPTION LATER
653                            ;        IN THIS PROGRAM.
654                            ;
655                            ;        +----------------------------------------------------+
656                            ;   C.CINT )       COMMAND INTERRUPT INDICATOR              )   2 BYTES
657                            ;        +----------------------------------------------------+
658                            ;   C.RINT )       RESPONSE INTERRUPT INDICATOR             )   2 BYTES
659                            ;        +----------------------------------------------------+
660                            ;   C.RRNG )       MESSAGE (RESPONSE) RING                  )   16 BYTES
661                            ;        )                                                  )
662                            ;        +----------------------------------------------------+
663                            ;   C.CRNG )       COMMAND RING                             )   16 BYTES
664                            ;        )                                                  )
665                            ;        +----------------------------------------------------+
666                            ;
667                            ;        NOTE: BYTES ARE GIVEN IN DECIMAL
668                            ;
669                            ;--
670   000077                  CT.UNT = 000077                         ; LOGICAL UNIT NUMBER MASK
671   000777                  CT.VEC = 000777                         ; VECTOR ADDRESS MASK
672   007000                  CT.BRL = 007000                         ; BR LEVEL MASK
673
674   100000                  CT.AVL =        BIT15                   ; SET WHEN NOT AVAILABLE FOR TESTING
675   040000                  CT.DUN =        BIT14                   ; SET IF DONE GENERATING PACKETS
676   020000                  CT.MSG =        BIT13                   ; SET IF INTERRUPT RECEIVED
677   010000                  CT.IOC =        BIT12                   ; SET IF TESTING I/O COMPLETED
678   004000                  CT.DI  =        BIT11                   ; SET IF CONTROLLER HAS UQSSP DIAGS
679   002000                  CT.DET =        BIT10                   ; SET IF DETERMINISTIC PHASE NOT DONE
680   000400                  CT.MRW =        BIT08                   ; MAINTENANCE OPERATIONS IN PROGRESS
681
682 002156                    TABLE                                  ; START A TABLE DEFINITION
683 002156                    ITEM C.IPR      2       <BUS ADDRESS OF IP REGISTER>
684 002156                    ITEM C.UNIT     2       <UNIT NUMBER TO TEST>
685 002156                    ITEM C.VEC      2       <VECTOR ADDRESS>
686 002156                    ITEM C.CTO      2       <CONTROLLER TIMEOUT PERIOD>
687 002156                    ITEM C.JSR      2       <INTERRUPT SERVICE ROUTINE FOR CONTROLLER>
688 002156                    ITEM C.JAD      2       <THESE TWO WORDS LOADED WITH [JSR R0    CNTSRV]>
689 002156                    ITEM C.FLG      2       <FLAGS>
690 002156                    ITEM C.DR0      2       <POINTER TO DRIVE TABLES>
691 002156                    ITEM C.DR1      2       <IF ZERO, NO DRIVE TABLE EXISTS>
692 002156                    ITEM C.DR2      2
```

```
693 002156          ITEM C.DR3      2
694 002156          ITEM C.TYPE     2    <CONTROLLER MODEL NUMBER>
695 002156          ITEM C.STEP     2    <CURRENT TEST STEP>
696 002156          ITEM C.UCNT     2    <COUNT OF DRIVES UNDER TEST>
697 002156          ITEM C.MSIZ     2    <CONTROLLER MEMORY SIZE>
698 002156          ITEM C.ROFF     2    <REGION OFFSET OF LAST I/O>
699 002156          ITEM C.TPAT     2    <DATA PATTERN CURRENTLY USED>
700 002156          ITEM C.NEXT     2    <NEXT DRIVE TO TEST>
701 002156          ITEM C.XFLW     2    <BYTES WRITTEN COUNT - LOW WORD>
702 002156          ITEM C.XFMW     2    <BYTES WRITTEN COUNT - MID WORD>
703 002156          ITEM C.XFHW     2    <BYTES WRITTEN COUNT - HIGH WORD>
704 002156          ITEM C.XFLR     2    <BYTES READ COUNT - LOW WORD>
705 002156          ITEM C.XFMR     2    <BYTES READ COUNT - MID WORD>
706 002156          ITEM C.XFHR     2    <BYTES READ COUNT - HIGH WORD>
707 C02156          ITEM C.UID      8.   <UNIT IDENTIFIER>
708 002156          ITEM C.CPTR     2    <COMMAND RING POINTER>
709 002156          ITEM C.CBAS     2    <BASE ADDRESS OF COMMAND RING>
710 002156          ITEM C.CEND     2    <FIRST ADDRESS AFTER THE COMMAND RING>
711 002156          ITEM C.RPTR     2    <RESPONSE RING POINTER>
712 002156          ITEM C.RBAS     2    <BASE ADDRESS OF RESPONSE RING>
713 002156          ITEM C.REND     2    <FIRST ADDRESS AFTER THE REPONSE RING>
714 002156          ITEM C.RHDR     2    <POINTER TO RESPONSE ENVELOPE RING>
715 002156          ITEM C.CINT     2    <COMMAND INTERRUPT INDICATOR>
716 002156          ITEM C.RINT     2    <RESPONSE INTERRUPT INDICATOR>
717 002156          ITEM C.RRNG     16.  <RESPONSE RING>
718 002156          ITEM C.CRNG     16.  <COMMAND RING>
719 002156          ITEM C.CRED     2    <NUMBER OF AVAILABLE CREDITS>
720 002156          ITEM C.TO       2    <CONTROLLER TIMEOUT COUNTER>
721 00215C          ITEM C.TOH      2    <  (TWO WORDS)>
722 002156          ITEM C.CMFL     2    <COMMAND LIST FORWARD LINK>
723 002156          ITEM C.CMBL     2    <COMMAND LIST BACK LINK>
724 002156          ITEM C.TBSF     2    <TO BE SENT LIST FORWARD LINK>
725 002156          ITEM C.TBSB     2    <TO BE SENT LIST BACK LINK>
726
727 002156          END C.SIZE           <SIZE OF CONTROLLER TABLE IN BYTES>
728
```

```
 730                                   .SBTTL  DRIVE TABLE DEFINITIONS
 731
 732                                   ;++
 733                                   ;       DRIVE TABLE DEFINITIONS
 734                                   ;
 735                                   ;       ONE DRIVE TABLE WILL BE SET UP BY THE INITIALIZE SECTION FOR EACH
 736                                   ;       DRIVE SELECTED FOR TESTING.  EACH TABLE IS POINTED TO BY A
 737                                   ;       WORD IN THE CONTROLLER TABLE ON WHICH THE DRIVE EXISTS.
 738                                   ;
 739                                   ;       THE FIRST TABLE IS POINTED TO BY THE CONTENTS OF DTABS.
 740                                   ;--
 741
 742      000077                       DT.UNT = 000077                          ; LOGICAL UNIT NUMBER OF DRIVE
 743
 744      100000                       DT.AVL = BIT15                           ; SET WHEN NOT AVAILABLE FOR TESTING
 745      040000                       DT.DUN = BIT14                           ; SET IF DONE SUBTEST
 746      020000                       DT.BES = BIT13                           ; SET IF TESTING ENTIRE DISK
 747      010000                       DT.WRT = BIT12                           ; SET IF DRIVE IS WRITEABLE
 748      004000                       DT.CMP = BIT11                           ; SET IF COMPARE HOST DATA PENDING
 749      002000                       DT.EOM = BIT10                           ; SET IF END OF MEDIA REACHED
 750      001000                       DT.ONL = BIT09                           ; SET IF DRIVE IS ONLINE
 751
 752      147000                       DT.CLR = DT.AVL!DT.DUN!DT.CMP!DT.EOM!DT.ONL ; FLAGS TO CLEAR AT EOP
 753      020000                       DT.PRM = DT.BES                          ; DEFAULT TEST PARAMETERS
 754
 755 002156                            TABLE                                    ; START A TABLE DEFINITION
 756 002156                            ITEM D.DRV     2      <DRIVE NUMBER>
 757 002156                            ITEM D.UNIT    2      <LOGICAL UNIT NUMBER>
 758 002156                            ITEM D.BEC     2      <BEGIN/END SET COUNT>
 759 002156                            ITEM D.BGN1    4      <BEGIN BLOCK 1>
 760 002156                            ITEM D.END1    4      <END>
 761 002156                            ITEM D.BGN2    4      <BEGIN BLOCK 2>
 762 002156                            ITEM D.END2    4      <END>
 763 002156                            ITEM D.BGN3    4      <BEGIN BLOCK 3>
 764 002156                            ITEM D.END3    4      <END>
 765 002156                            ITEM D.BGN4    4      <BEGIN BLOCK 4>
 766 002156                            ITEM D.END4    4      <END>
 767 002156                            ITEM D.BES     2      <CURRENT BEGIN/END SET>
 768 002156                            ITEM D.BLKS    2      <BLOCKS LEFT IN TEST SEGMENT>
 769 002156                            ITEM D.LBN     4      <NEXT LBN TO ACCESS>
 770 002156                            ITEM D.WCNT    2      <# OF OUTSTANDING WRITES>
 771 002156                            ITEM D.CCNT    2      <# OF OUTSTANDING COMPARE HOST DATAS>
 772 002156                            ITEM D.XFLW    2      <BYTES WRITTEN COUNT - LOW WORD>
 773 002156                            ITEM D.XFMW    2      <BYTES WRITTEN COUNT - MID WORD>
 774 002156                            ITEM D.XFHW    2      <BYTES WRITTEN COUNT - HIGH WORD>
 775 002156                            ITEM D.XFLR    2      <BYTES READ COUNT - LOW WORD>
 776 002156                            ITEM D.XFMR    2      <BYTES READ COUNT - MID WORD>
 777 002156                            ITEM D.XFHR    2      <BYTES READ COUNT - HIGH WORD>
 778 002156                            ITEM D.XFLA    2      <BYTES ACCESSED COUNT - LOW WORD>
 779 002156                            ITEM D.XFMA    2      <BYTES ACCESSED COUNT - MID WORD>
 780 002156                            ITEM D.XFHA    2      <BYTES ACCESSED COUNT - HIGH WORD>
 781 002156                            ITEM D.XFLC    2      <BYTES COMPARED COUNT - LOW WORD>
 782 002156                            ITEM D.XFMC    2      <BYTES COMPARED COUNT - MID WORD>
 783 002156                            ITEM D.XFHC    2      <BYTES COMPARED COUNT - HIGH WORD>
 784 002156                            ITEM D.HERR    2      <HARD ERROR COUNTER>
```

```
785 002156                ITEM D.SERR      2      <SOFT ERROR COUNTER>
786 002156                ITEM D.CERR      2      <COMPARE ERROR COUNTER>
787 002156                ITEM D.ECCC      2      <ECC COUNTER>
788 002156                ITEM D.UID       8.     <UNIT IDENTIFIER>
789 002156                ITEM D.VSN       4      <VOLUME SERIAL NUMBER>
790
791 002156                END D.SIZE       <SIZE OF DRIVE TABLE IN BYTES>
792
```

```
794                              .SBTTL  GLOBAL DATA SECTION
795
796                         ;++
797                         ;       THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
798                         ;       IN MORE THAN ONE TEST.
799                         ;--
800
801 002156                         ERRTBL
    002156              L$ERRTBL::
    002156  000000      ERRTYP::        .WORD   0
    002160  000000      ERRNBR::        .WORD   0
    002162  000000      ERRMSG::        .WORD   0
    002164  000000      ERRBLK::        .WORD   0
802
803 002166              FFREE:  .BLKW   1               ; FIRST FREE WORD IN MEMORY
804 002170              FSIZE:  .BLKW   1               ; SIZE OF FREE MEMORY IN WORDS
805 002172              FMADR:  .BLKW   1               ; PERMANENT FREE MEM ADDR STORAGE
806 002174              FMSIZ:  .BLKW   1               ; PERMANENT FREE MEM SIZE STORAGE
807 002176              DTABS:  .BLKW   1               ; START OF DRIVE TABLE STORAGE
808 002200              CTABS:  .BLKW   1               ; START OF CONTROLLER TABLE STORAGE
809 002202              CTRLRS: .BLKW   1               ; COUNT OF CONTROLLERS IN PTABLES
810
811 002204  000000      IFLAGS: .WORD   0               ; FLAGS FROM INIT CODE
812
813         100000      IMMU    =       BIT15           ; MEMORY MANAGEMENT FLAG
814         040000      IDIE    =       BIT14           ; ABORT PROGRAM FLAG
815         020000      IERL    =       BIT13           ; ERROR LOG FLAG
816         010000      IDUN    =       BIT12           ; TEST DONE FLAG
817         000100      IPAT16  =       BIT6            ; PATTERN 16 MODIFIED FLAG
818         000040      ITEMP   =       BIT5            ; TEMPORARY FLAG
819         000010      ISTRT   =       BIT3            ; START FLAG
820         000004      IREST   =       BIT2            ; RESTART FLAG
821         000002      ICONT   =       BIT1            ; CONTINUE EVENT FLAG
822
823 002206  000000      TNUM:   .WORD   0               ; NUMBER OF CURRENT TEST
824 002210              UTEST:  .BLKW   1               ; NUMBER OF UNITS STILL RUNNING TEST
825
826 002212              TEMP:   .BLKW   12.             ; TEMPORY STORAGE FOR GMANI RESPONSES
827 002242              TSTOFF: .BLKW   1               ; MEMORY OFFSET OF BUFFER
828 002244              TSTOPC: .BLKW   1               ; OPCODE FOR CURRENT PACKET
829 002246              TSPGEN: .BLKW   1               ; ADDRESS OF PACKET GENERATION ROUTINE
830 002250              STIME:  .BLKW   2               ; STATISTICAL REPORT TIMER
831 002254              TSTIM:  .BLKW   2               ; TEST TIMER
832 002260              TS.PAT: .BLKW   1               ; TEST PATTERN TO USE
833
834 002262              VECSAV: .BLKW   1               ; STORAGE FOR COMPUTED VECTOR
835
836 002264              NXMAD:  .BLKW   1               ; SET TO -1 BY NON-EXISTANT ADDRESS
837
838 002266  000000      RNTHH:  .WORD   0               ; STORAGE FOR ELAPSED HOURS
839 002270  000000      RNTMM:  .WORD   0               ; STORAGE FOR ELAPSED MINUTES
840 002272  000000      RNTSS:  .WORD   0               ; STORAGE FOR ELAPSED SECONDS
841 002274              RPTLUN: .BLKW   0               ; STORAGE FOR LOGICAL UNIT NUMBER
842 002274              RPTWV:  .BLKW   1               ; STORAGE FOR BYTES WRITTEN
843 002276              RPTWF:  .BLKW   1               ; STORAGE FOR W/K FLAG
```

```
844 002300            RPTRV:  .BLKW   1              ; STORAGE FOR BYTES READ
845 002302            RPTRF:  .BLKW   1              ; STORAGE FOR M/K FLAG
846 002304            RPTAV:  .BLKW   1              ; STORAGE FOR BYTES ACCESSED
847 002306            RPTAF:  .BLKW   1              ; STORAGE FOR M/K FLAG
848 002310            RPTCV:  .BLKW   1              ; STORAGE FOR BYTES COMPARED
849 002312            RPTCF:  .BLKW   1              ; STORAGE FOR M/K FLAG
850
851 002314            MSCPOP: .BLKW   1              ; POINTER TO MSCP OPCODE MESSAGE
852 002316            STAMSO: .BLKW   1              ; POINTER TO MSCP ERROR
853 002320            STAMS1: .BLKW   1              ;   MESSAGE STRING
854                   ;
855                   ;          KW11 CLOCK CONTROL
856                   ;
857 002322  000000    KW.CSR: .WORD   0              ; CSR OF CLOCK
858 002324            KW.BRL: .BLKW   1              ; BR LEVEL
859 002326            KW.VEC: .BLKW   1              ; VECTOR
860 002330            KW.HZ:  .BLKW   1              ; HERTZ (50. OR 60.)
861 002332            KW.EL:  .BLKW   2              ; ELAPSED TIME
862 002336            KW.60:  .BLKW   1              ; HERTZ * 60
863                   ;
864                   ;          DATA TO BE SENT AND RECEIVED BY CONTROLLER INITIALIZATION
865                   ;
866
867 002340  053122    INITBL: .WORD   RSP.S1         ; 1ST WORD RESPONSE CHECK ROUTINE
868 002342  000000    SND.S1: .WORD   0              ; 1ST WORD TO SEND TO SA REGISTER
869 002344  053160            .WORD   RSP.S2         ; 2ND WORD RESPONSE CHECK ROUTINE
870 002346  000000    SND.S2: .WORD   0              ; 2ND WORD TO SEND TO SA REGISTER
871 002350  053204            .WORD   RSP.S3         ; 3RD WORD RESPONSE CHECK ROUTINE
872
873 002352  000000    SSTEP4: .WORD   0              ; LOCATION TO SAVE STEP 4 VALUE
874 002354  000000    CNTRSD: .WORD   0              ; LOCATION FOR STEP BIT MASK
875 002356            WCHNGD: .BLKW   1              ; SAVED LOOPBACK DATA
876 002360  034245    LOSEED: .WORD   34245          ; LO ORDER RANDOM NUMBER SEED
877 002362  061453    HISEED: .WORD   61453          ; HI ORDER RANDOM NUMBER SEED
878
879         032413    Q50MSZ  =       32413          ; KDA50-Q MEMORY SIZE
880         036413    U50MSZ  =       36413          ; UDA50-A MEMORY SIZE
881         000000    QDXMSZ  =       0              ; RQDX1 MEMORY SIZE
882         000000    C25MSZ  =       0              ; RUC25 MEMORY SIZE
883
884         000004    RNGLEN  =       4              ; # OF ENTRIES IN CMD/RSP RINGS
885         011000    RNGPWR  =       011000         ; RING LENGTH FOR STEP 1
886
887 002364            OBUFF:  .BLKW   256.           ; LOW MEMORY DATA BUFFER
888 003364            CBUFF:  .BLKW   256.           ; DATA COMPARE BUFFER
889 004364            PCGOOD: .BLKW   1              ; TEMPORARY STORAGE FOR
890 004366            PCBAD:  .BLKW   1              ;   DATA COMPARE ERROR MESSAGE
891 004370            BUFBA:  .BLKW   1              ; BUFFER DESCRIPTOR (LOW WORD)
892 004372            BUFEA:  .BLKW   1              ; BUFFER DESCRIPTOR (HIGH WORD)
893 004374            BUFBLK: .BLKW   1              ; BUFFER SIZE IN BLOCKS
894 004376            BUFSIZ: .BLKW   1              ; BUFFER SIZE IN WORDS
895 004400            MMBUF:  .BLKW   1              ; ADDRESS OF LOW MEMORY BUFFER
896 004402            MMSIZ:  .BLKW   1              ; SIZE OF LOW MEMORY BUFFER
897 004404            MMOFF:  .BLKW   1              ; MAPPED MEMORY OFFSET
898 004406            BCLO:   .BLKW   1              ; BYTE COUNT (LOW WORD)
```

899 004410   BCHI: .BLKW   1   ; BYTE COUNT (HIGH WORD)

```
901                          .SBTTL LINKED LIST STRUCTURES
902
903              ;++
904              ;       THE MAPPED MEMORY PACKETS CONTAIN INFORMATION ON THE I/O BUFFERS
905              ;       THAT ARE ALLOCATED IN MAPPED MEMORY. THE PROGRAM ALLOCATES MAPPED
906              ;       MEMORY IN 512 BYTE BLOCKS. EACH ALLOCATED SEGMENT OF MEMORY IS
907              ;       REFERENCED BY A MAPPED MEMORY PACKET. IF THE "NUMBER OF USERS"
908              ;       FIELD IS ZERO, THE SEGMENT OF MEMORY IS AVAILABLE FOR USE BY A
909              ;       NEW COMMAND. IF IT IS NON-ZERO, THE SEGMENT IS CURRENTLY IN USE.
910              ;       IF A SEGMENT IS REFERENCED BY THE MULTI-USE TABLE, IT CONTAINS
911              ;       ONE OF THE TEST DATA PATTERNS AND IS AVAILABLE FOR READ-ONLY USE
912              ;       BY ANY ADDITIONAL COMMAND. ONLY ONE SEGMENT OF EACH DATA PATTERN
913              ;       CAN BE REFERENCED BY THE MULTI-USE TABLE. SEGMENTS ARE ADDED TO
914              ;       THE MULTI-USE TABLE IF THEY ARE BEING USED IN A OUTPUT COMMAND AND
915              ;       THE TABLE ENTRY FOR THE PATTERN IS SMALLER THAN THE ALLOCATED SEGMENT
916              ;       OR THE ENTRY IS EMPTY. SEGMENTS ARE REMOVED FROM THE MULTI-USE TABLE
917              ;       IF THEY ARE REPLACED BY A LARGER SEGMENT OF THE SAME PATTERN, OR IF
918              ;       THEY ARE NOT BEING USED AND A MERGE IS DONE ON THE MAPPED MEMORY
919              ;       SEGMENTS.
920              ;
921              ;       THE MAPPED MEMORY PACKET FORMAT IS AS FOLLOWS:
922              ;
923              ;       +------------------------------------------------+
924              ;       )              PACKET FORWARD LINK              )   2 BYTES
925              ;       +------------------------------------------------+
926              ;       ) NUMBER OF USERS    ) SIZE OF SEGMENT (BLOCKS) )   2 BYTES
927              ;       +------------------------------------------------+
928              ;       )           MAPPED MEMORY OFFSET (BYTES)        )   2 BYTES
929              ;       +------------------------------------------------+
930              ;--
931
932      000006  PKTSIZ =  6.                     ; SIZE IN BYTES OF MEMORY MAP PACKET
933 004412      MUTOFF: .BLKW  34.                ; MULTI-USE TABLE, BYTE OFFSET
934 004516      MUTSIZ: .BLKW  34.                ; MULTI-USE TABLE, BUFFER SIZE (BLOCKS)
935 004622      M.HDR:  .BLKW  1                  ; IN USE MEMORY PACKET LIST HEADER
936 004624      F.HDR:  .BLKW  1                  ; AVAILABLE MEMORY PACKET LIST HEADER
937 004626      MMPKTS: .BLKW  384.               ; ALLOCATION OF MAPPED MEMORY PACKETS
938
939              ;++
940              ;       THE RESPONSE ENVELOPE BUFFER WILL CONTAIN DATA SENT BY THE CONTROLLER
941              ;       TO THE HOST. ONE BUFFER IS ALLOCATED TO EACH CONTROLLER TO AWAIT DATA.
942              ;       AS SOON AS A MESSAGE IS RECEIVED FROM THE CONTROLLER, THE FULL RESPONSE
943              ;       ENVELOPE IS REMOVED FROM THE RESPONSE RING AND REPLACED BY AN EMPTY
944              ;       RESPONSE ENVELOPE. AT THIS POINT THE DATA IN THE FULL RESPONSE
945              ;       ENVELOPE CAN BE PROCESSED.
946              ;
947              ;       THE RESPONSE ENVELOPE BUFFER FORMAT IS AS FOLLOWS:
948              ;
949              ;             +------------------------------------------------+
950              ;       RE.RFL )            RESPONSE FORWARD LINK             )   2 BYTES
951              ;             +------------------------------------------------+
952              ;       RE.RL  )            RESPONSE PACKET LENGTH            )   2 BYTES
953              ;             +------------------------------------------------+
954              ;       RE.RV  )  CONNECTION ID    ) MSG TYPE ) CREDITS )       2 BYTES
955              ;             +------------------------------------------------+
```

```
 956                              ;        RE.RP   }                    RESPONSE PACKET                     } 128 BYTES
 957                              ;                }
 958                              ;                +-----------------------------------------------------+
 959                              ;
 960                              ;        NOTE: BYTES ARE GIVEN IN DECIMAL
 961                              ;--
 962
 963         000000              RE.RFL  =        0.                          ; R.E. BUFFER FORWARD LINK
 964         000002              RE.RL   =        2.                          ; RESPONSE PACKET LENGTH
 965         000004              RE.RV   =        4.                          ; CIRCUIT ID, MSG TYPE, CREDITS
 966         000006              RE.RP   =        6.                          ; RESPONSE PACKET (128. BYTE)
 967
 968         000103              RSPSIZ  =        67.                         ; SIZE IN WORDS OF A RSP ENVELOPE
 969 006226                      RSPCNT: .BLKW    1                           ; NUMBER OF R.E. BUFFERS
 970 006230                      RSPSTR: .BLKW    1                           ; START OF R.E. BUFFER LIST
 971
 972                              ;++
 973                              ;        THE MESSAGE ENVELOPE BUFFER CONTAINS DATA ON EACH COMMAND THAT IS
 974                              ;        SENT TO THE CONTROLLER. THE BUFFERS ARE KEPT IN A 'TO BE SENT' LIST
 975                              ;        UNTIL THE COMMAND RING ENTRY BECOMES AVAILABLE. AT THAT POINT THE
 976                              ;        ADDRESS OF THE FIRST COMMAND PACKET (OFFSET ME.CP) IN THE TBS LIST
 977                              ;        IS LOADED INTO THE COMMAND RING. THE BUFFER IS THEN PLACED IN THE
 978                              ;        OUTSTANDING COMMAND LIST.
 979
 980                              ;        THE MESSAGE ENVELOPE BUFFER FORMAT IS AS FOLLOWS:
 981
 982                              ;        +-----------------------------------------------------+
 983                              ;        ME.CFL  }              COMMAND FLINK                  }  2 BYTES
 984                              ;        +-----------------------------------------------------+
 985                              ;        ME.CBL  }              COMMAND BLINK                  }  2 BYTES
 986                              ;        +-----------------------------------------------------+
 987                              ;        ME.CRF  }           COMMAND REFERENCE NUMBER          }  2 BYTES
 988                              ;        +-----------------------------------------------------+
 989                              ;        ME.CTO  }     COMMAND TIMEOUT INTERVAL (LOW WORD)     }  2 BYTES
 990                              ;        +-----------------------------------------------------+
 991                              ;        ME.CTH  }     COMMAND TIMEOUT INTERVAL (HIGH WORD)    }  2 BYTES
 992                              ;        +-----------------------------------------------------+
 993                              ;        ME.ST1  }         COMMAND STATUS (LOW WORD)           }  2 BYTES
 994                              ;        +-----------------------------------------------------+
 995                              ;        ME.ST2  }         COMMAND STATUS (HIGH WORD)          }  2 BYTES
 996                              ;        +-----------------------------------------------------+
 997                              ;        ME.CFG  }              COMMAND FLAGS                  }  2 BYTES
 998                              ;        +-----------------------------------------------------+
 999                              ;        ME.OFF  }   MAPPED MEMORY BYTE OFFSET TO I/O BUFFER   }  2 BYTES
1000                              ;        +-------------------------------------+---------------+
1001                              ;ME.PKT,ME.CL  }          COMMAND PACKET LENGTH                }  2 BYTES
1002                              ;        +-----------------------------------------------------+
1003                              ;        ME.CV   }   CONNECTION ID     } MSG TYPE  } CREDITS )  2 BYTES
1004                              ;        +-----------------------------------------------------+
1005                              ;        ME.CP   }              COMMAND PACKET                 } 60 BYTES
1006                              ;                }                                             }
1007                              ;        +-----------------------------------------------------+
1008
1009                              ;        NOTE: BYTES ARE GIVEN IN DECIMAL
1010                              ;--
```

```
1011
1012        100000              CF.FAIL =      BIT15           ; CMD FAILED FLAG
1013        040000              CF.CTO  =      BIT14           ; GCS ISSUED FLAG
1014        020000              CF.ERL  =      BIT13           ; ERROR LOG GENERATED
1015
1016        000000              ME.CFL  =      0.              ; CMD LIST FORWARD LINK
1017        000002              ME.CBL  =      2.              ; CMD LIST BACK LINK
1018        000004              ME.CRF  =      4.              ; CMD REFERENCE NUMBER
1019        000006              ME.CTO  =      6.              ; COMMAND TIMER (LOW WORD)
1020        000010              ME.CTH  =      8.              ; COMMAND TIMER (HIGH WORD)
1021        000012              ME.ST1  =      10.             ; COMMAND STATUS (LOW WORD)
1022        000014              ME.ST2  =      12.             ; COMMAND STATUS (HIGH WORD)
1023        000016              ME.CFG  =      14.             ; COMMAND FLAGS
1024        000020              ME.OFF  =      16.             ; MAPPED MEMORY OFFSET OF DATA BUFFER
1025        000022              ME.PKT  =      18.             ; START ADDRESS OF COMMAND PACKET
1026        000022              ME.CL   =      18.             ; COMMAND PACKET LENGTH
1027        000024              ME.CV   =      20.             ; CIRCUIT ID, MSG TYPE, CREDITS
1028        000026              ME.CP   =      22.             ; COMMAND PACKET (60 BYTE)
1029
1030        000051              CMDSIZ  =      41.             ; SIZE IN WORDS OF A ME BUFFER
1031 006232              MEMIN:  .BLKW   1                     ; MINIMUM SIZE ALLOCATION
1032 006234              TBSMIN: .BLKW   1                     ; MINIMUM NUMBER OF M.E.
1033 006236              TBSSIZ: .BLKW   1                     ; NUMBER OF AVAILABLE ME BUFFERS
1034 006240              TBSTRT: .BLKW   1                     ; START OF ME BUFFER LIST
1035 006242              T.HDR:  .BLKW   2                     ; FREE ME BUFFER LIST HEADER
```

```
1037                                    .SBTTL TEST DATA PATTERNS
1038
1039                              ;
1040                              ;         TEST DATA PATTERNS
1041                              ;
1042 006246  000000              PATC0:  .WORD   000000          ; PATTERN 0 ID WORD (MAPS TO PAT. 16)
1043 006250  000001                      .WORD   1               ; PATTERN SIZE
1044 006252  000000                      .WORD   000000          ; DATA PATTERN
1045 006254  000000                      .WORD   000000          ; DATA PATTERN
1046 006256  000000                      .WORD   000000          ; DATA PATTERN
1047 006260  000000                      .WORD   000000          ; DATA PATTERN
1048 006262  000000                      .WORD   000000          ; DATA PATTERN
1049 006264  000000                      .WORD   000000          ; DATA PATTERN
1050 006266  000000                      .WORD   000000          ; DATA PATTERN
1051 006270  000000                      .WORD   000000          ; DATA PATTERN
1052 006272  000000                      .WORD   000000          ; DATA PATTERN
1053 006274  000000                      .WORD   000000          ; DATA PATTERN
1054 006276  000000                      .WORD   000000          ; DATA PATTERN
1055 006300  000000                      .WORD   000000          ; DATA PATTERN
1056 006302  000000                      .WORD   000000          ; DATA PATTERN
1057 006304  000000                      .WORD   000000          ; DATA PATTERN
1058 006306  000000                      .WORD   000000          ; DATA PATTERN
1059 006310  000000                      .WORD   000000          ; DATA PATTERN
1060 006312  010421              PAT01:  .WORD   010421          ; PATTERN 1 ID WORD
1061 006314  000001                      .WORD   1               ; PATTERN SIZE
1062 006316  105613                      .WORD   105613          ; DATA PATTERN
1063 006320  021042              PAT02:  .WORD   021042          ; PATTERN 2 ID WORD
1064 006322  000001                      .WORD   1               ; PATTERN SIZE
1065 006324  031463                      .WORD   031463          ; DATA PATTERN
1066 006326  031463              PAT03:  .WORD   031463          ; PATTERN 3 ID WORD
1067 006330  000001                      .WORD   1               ; PATTERN SIZE
1068 006332  030221                      .WORD   030221          ; DATA PATTERN
1069 006334  042104              PAT04:  .WORD   042104          ; PATTERN 4 ID WORD
1070 006336  000020                      .WORD   16.             ; PATTERN SIZE
1071 006340  000001                      .WORD   000001          ; DATA PATTERN
1072 006342  000003                      .WORD   000003          ; DATA PATTERN
1073 006344  000007                      .WORD   000007          ; DATA PATTERN
1074 006346  000017                      .WORD   000017          ; DATA PATTERN
1075 006350  000037                      .WORD   000037          ; DATA PATTERN
1076 006352  000077                      .WORD   000077          ; DATA PATTERN
1077 006354  000177                      .WORD   000177          ; DATA PATTERN
1078 006356  000377                      .WORD   000377          ; DATA PATTERN
1079 006360  000777                      .WORD   000777          ; DATA PATTERN
1080 006362  001777                      .WORD   001777          ; DATA PATTERN
1081 006364  003777                      .WORD   003777          ; DATA PATTERN
1082 006366  007777                      .WORD   007777          ; DATA PATTERN
1083 006370  017777                      .WORD   017777          ; DATA PATTERN
1084 006372  037777                      .WORD   037777          ; DATA PATTERN
1085 006374  077777                      .WORD   077777          ; DATA PATTERN
1086 006376  177777                      .WORD   177777          ; DATA PATTERN
1087 006400  052525              PAT05:  .WORD   052525          ; PATTERN 5 ID WORD
1088 006402  000020                      .WORD   16.             ; PATTERN SIZE
1089 006404  177776                      .WORD   177776          ; DATA PATTERN
1090 006406  177774                      .WORD   177774          ; DATA PATTERN
1091 006410  177770                      .WORD   177770          ; DATA PATTERN
```

```
1092 006412  177760                .WORD    177760          ; DATA PATTERN
1093 006414  177740                .WORD    177740          ; DATA PATTERN
1094 006416  177700                .WORD    177700          ; DATA PATTERN
1095 006420  177600                .WORD    177600          ; DATA PATTERN
1096 006422  177400                .WORD    177400          ; DATA PATTERN
1097 006424  177000                .WORD    177000          ; DATA PATTERN
1098 006426  176000                .WORD    176000          ; DATA PATTERN
1099 006430  174000                .WORD    174000          ; DATA PATTERN
1100 006432  170000                .WORD    170000          ; DATA PATTERN
1101 006434  160000                .WORD    160000          ; DATA PATTERN
1102 006436  140000                .WORD    140000          ; DATA PATTERN
1103 006440  100000                .WORD    100000          ; DATA PATTERN
1104 006442  000000                .WORD    000000          ; DATA PATTERN
1105 006444  063146        PAT06:  .WORD    063146          ; PATTERN 6 ID WORD
1106 006446  000020                .WORD    16.              ; PATTERN SIZE
1107 006450  000000                .WORD    000000          ; DATA PATTERN
1108 006452  000000                .WORD    000000          ; DATA PATTERN
1109 006454  000000                .WORD    000000          ; DATA PATTERN
1110 006456  177777                .WORD    177777          ; DATA PATTERN
1111 006460  177777                .WORD    177777          ; DATA PATTERN
1112 006462  177777                .WORD    177777          ; DATA PATTERN
1113 006464  000000                .WORD    000000          ; DATA PATTERN
1114 006466  000000                .WORD    000000          ; DATA PATTERN
1115 006470  177777                .WORD    177777          ; DATA PATTERN
1116 006472  177777                .WORD    177777          ; DATA PATTERN
1117 006474  000000                .WORD    000000          ; DATA PATTERN
1118 006476  177777                .WORD    177777          ; DATA PATTERN
1119 006500  000000                .WORD    000000          ; DATA PATTERN
1120 006502  177777                .WORD    177777          ; DATA PATTERN
1121 006504  000000                .WORD    000000          ; DATA PATTERN
1122 006506  177777                .WORD    177777          ; DATA PATTERN
1123 006510  073567        PAT07:  .WORD    073567          ; PATTERN 7 ID WORD
1124 006512  000001                .WORD    1                ; PATTERN SIZE
1125 006514  133331                .WORD    133331          ; DATA PATTERN
1126 006516  104210        PAT08:  .WORD    104210          ; PATTERN 8 ID WORD
1127 006520  000020                .WORD    16.              ; PATTERN SIZE
1128 006522  052525                .WORD    052525          ; DATA PATTERN
1129 006524  052525                .WORD    052525          ; DATA PATTERN
1130 006526  052525                .WORD    052525          ; DATA PATTERN
1131 006530  125252                .WORD    125252          ; DATA PATTERN
1132 006532  125252                .WORD    125252          ; DATA PATTERN
1133 006534  125252                .WORD    125252          ; DATA PATTERN
1134 006536  052525                .WORD    052525          ; DATA PATTERN
1135 006540  052525                .WORD    052525          ; DATA PATTERN
1136 006542  125252                .WORD    125252          ; DATA PATTERN
1137 006544  125252                .WORD    125252          ; DATA PATTERN
1138 006546  052525                .WORD    052525          ; DATA PATTERN
1139 006550  125252                .WORD    125252          ; DATA PATTERN
1140 006552  052525                .WORD    052525          ; DATA PATTERN
1141 006554  125252                .WORD    125252          ; DATA PATTERN
1142 006556  052525                .WORD    052525          ; DATA PATTERN
1143 006560  125252                .WORD    125252          ; DATA PATTERN
1144 006562  114631        PAT09:  .WORD    114631          ; PATTERN 9 ID WORD
1145 006564  000001                .WORD    1                ; PATTERN SIZE
1146 006566  155554                .WORD    155554          ; DATA PATTERN
```

```
1147 006570 125252        PAT10:  .WORD   125252     ; PATTERN 10 ID WORD
1148 006572 000020                .WORD   16.        ; PATTERN SIZE
1149 006574 026455                .WORD   026455     ; DATA PATTERN
1150 006576 026455                .WORD   026455     ; DATA PATTERN
1151 006600 026455                .WORD   026455     ; DATA PATTERN
1152 006602 151322                .WORD   151322     ; DATA PATTERN
1153 006604 151322                .WORD   151322     ; DATA PATTERN
1154 006606 151322                .WORD   151322     ; DATA PATTERN
1155 006610 026455                .WORD   026455     ; DATA PATTERN
1156 006612 026455                .WORD   026455     ; DATA PATTERN
1157 006614 151322                .WORD   151322     ; DATA PATTERN
1158 006616 151322                .WORD   151322     ; DATA PATTERN
1159 006620 026455                .WORD   026455     ; DATA PATTERN
1160 006622 151322                .WORD   151322     ; DATA PATTERN
1161 006624 026455                .WORD   026455     ; DATA PATTERN
1162 006626 151322                .WORD   151322     ; DATA PATTERN
1163 006630 026455                .WORD   026455     ; DATA PATTERN
1164 006632 151322                .WORD   151322     ; DATA PATTERN
1165 006634 135673        PAT11:  .WORD   135673     ; PATTERN 11 ID WORD
1166 006636 000001                .WORD   1          ; PATTERN SIZE
1167 006640 066666                .WORD   066666     ; DATA PATTERN
1168 006642 146314        PAT12:  .WORD   146314     ; PATTERN 12 ID WORD
1169 006644 000020                .WORD   16.        ; PATTERN SIZE
1170 006646 000001                .WORD   000001     ; DATA PATTERN
1171 006650 000002                .WORD   000002     ; DATA PATTERN
1172 006652 000004                .WORD   000004     ; DATA PATTERN
1173 006654 000010                .WORD   000010     ; DATA PATTERN
1174 006656 000020                .WORD   000020     ; DATA PATTERN
1175 006660 000040                .WORD   000040     ; DATA PATTERN
1176 006662 000100                .WORD   000100     ; DATA PATTERN
1177 006664 000200                .WORD   000200     ; DATA PATTERN
1178 006666 000400                .WORD   000400     ; DATA PATTERN
1179 006670 001000                .WORD   001000     ; DATA PATTERN
1180 006672 002000                .WORD   002000     ; DATA PATTERN
1181 006674 004000                .WORD   004000     ; DATA PATTERN
1182 006676 010000                .WORD   010000     ; DATA PATTERN
1183 006700 020000                .WORD   020000     ; DATA PATTERN
1184 006702 040000                .WORD   040000     ; DATA PATTERN
1185 006704 100000                .WORD   100000     ; DATA PATTERN
1186 006706 156735        PAT13:  .WORD   156735     ; PATTERN 13 ID WORD
1187 006710 000020                .WORD   16.        ; PATTERN SIZE
1188 006712 177776                .WORD   177776     ; DATA PATTERN
1189 006714 177775                .WORD   177775     ; DATA PATTERN
1190 006716 177773                .WORD   177773     ; DATA PATTERN
1191 006720 177767                .WORD   177767     ; DATA PATTERN
1192 006722 177757                .WORD   177757     ; DATA PATTERN
1193 006724 177737                .WORD   177737     ; DATA PATTERN
1194 006726 177677                .WORD   177677     ; DATA PATTERN
1195 006730 177577                .WORD   177577     ; DATA PATTERN
1196 006732 177377                .WORD   177377     ; DATA PATTERN
1197 006734 176777                .WORD   176777     ; DATA PATTERN
1198 006736 175777                .WORD   175777     ; DATA PATTERN
1199 006740 173777                .WORD   173777     ; DATA PATTERN
1200 006742 167777                .WORD   167777     ; DATA PATTERN
1201 006744 157777                .WORD   157777     ; DATA PATTERN
```

```
1202 006746 137777                  .WORD   137777          ; DATA PATTERN
1203 006750 077777                  .WORD   077777          ; DATA PATTERN
1204 006752 167356          PAT14:  .WORD   167356          ; PATTERN 14 ID WORD
1205 006754 000003                  .WORD   3               ; PATTERN SIZE
1206 006756 155555                  .WORD   155555          ; DATA PATTERN
1207 006760 133333                  .WORD   133333          ; DATA PATTERN
1208 006762 155555                  .WORD   155555          ; DATA PATTERN
1209 006764 177777          PAT15:  .WORD   177777          ; PATTERN 15 ID WORD
1210 006766 000003                  .WORD   3               ; PATTERN SIZE
1211 006770 155555                  .WORD   155555          ; DATA PATTERN
1212 006772 133333                  .WORD   133333          ; DATA PATTERN
1213 006774 066666                  .WORD   066666          ; DATA PATTERN
1214
1215 006776 006246          PATDSP: .WORD   PAT00           ; POINTERS
1216 007000 006312                  .WORD   PAT01           ;   TO
1217 007002 006320                  .WORD   PAT02           ;     DATA
1218 007004 006326                  .WORD   PAT03           ;       PATTERNS
1219 007006 006334                  .WORD   PAT04           ;
1220 007010 006400                  .WORD   PAT05           ;
1221 007012 006444                  .WORD   PAT06           ;
1222 007014 006510                  .WORD   PAT07           ;
1223 007016 006516                  .WORD   PAT08           ;
1224 007020 006562                  .WORD   PAT09           ;
1225 007022 006570                  .WORD   PAT10           ;
1226 007024 006634                  .WORD   PAT11           ;
1227 007026 006642                  .WORD   PAT12           ;
1228 007030 006706                  .WORD   PAT13           ;
1229 007032 006752                  .WORD   PAT14           ;
1230 007034 006764                  .WORD   PAT15           ;
1231
1232 007036 000000          MNTPAT: .WORD   000000          ; POINTERS
1233 007040 177777                  .WORD   177777          ;   TO
1234 007042 000001                  .WORD   000001          ;     DATA
1235 007044 177776                  .WORD   177776          ;       PATTERNS
1236 007046 000002                  .WORD   000002          ;
1237 007050 177775                  .WORD   177775          ;
1238 007052 000004                  .WORD   000004          ;
1239 007054 177773                  .WORD   177773          ;
1240 007056 000010                  .WORD   000010          ;
1241 007060 177767                  .WORD   177767          ;
1242 007062 000020                  .WORD   000020          ;
1243 007064 177757                  .WORD   177757          ;
1244 007066 000040                  .WORD   000040          ;
1245 007070 177737                  .WORD   177737          ;
1246 007072 000100                  .WORD   000100          ;
1247 007074 177677                  .WORD   177677          ;
1248 007076 000200                  .WORD   000200          ;
1249 007100 177577                  .WORD   177577          ;
1250 007102 000400                  .WORD   000400          ;
1251 007104 177377                  .WORD   177377          ;
1252 007106 001000                  .WORD   001000          ;
1253 007110 176777                  .WORD   176777          ;
1254 007112 002000                  .WORD   002000          ;
1255 007114 175777                  .WORD   175777          ;
1256 007116 004000                  .WORD   004000          ;
```

```
1257 007120  173777                    .WORD   173777              ;
1258 007122  010000                    .WORD   010000              ;
1259 007124  167777                    .WORD   167777              ;
1260 007126  020000                    .WORD   020000              ;
1261 007130  157777                    .WORD   157777              ;
1262 007132  040000                    .WORD   040000              ;
1263 007134  137777                    .WORD   137777              ;
1264 007136  100000                    .WORD   100000              ;
1265 007140  077777                    .WORD   077777              ;
```

```
1267                                        .SBTTL  GLOBAL TEXT SECTION
1268
1269                                     ;++
1270                                     ;        THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
1271                                     ;        MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
1272                                     ;        MORE THAN ONE TEST.
1273                                     ;--
1274
1275
1276                                     ;        NAMES OF DEVICES SUPPORTED BY PROGRAM
1277                                     ;
1278 007142                                      DEVTYP <DSA DISK DRIVE>
     007142                              L$DVTYP::
     007142    104    123    101                 .ASCIZ  %DSA DISK DRIVE%
                                                 .EVEN
1279                                     ;
1280                                     ;        TEST DESCRIPTION
1281                                     ;
1282 007162                                      DESCRIPT <CZUDJAO UDA50-A,KDA50-Q DISK SUBSYSTEM EXERCISER>
     007162                              L$DESC::
     007162    103    132    125                 .ASCIZ  /CZUDJAO UDA50-A,KDA50-Q DISK SUBSYSTEM EXERCISER/
                                                 .EVEN
1283                                     ;
1284                                     ;        UNFORMATTED MESSAGES
1285                                     ;
1286
1287 007244 007254                       TNAMES: .WORD   BASN0
1288 007246 007300                               .WORD   BASN1
1289 007250 007330                               .WORD   BASN2
1290 007252 007357                               .WORD   BASN3
1291
1292 007254    124    105    123         BASN0:  .ASCIZ  \TEST INITIALIZATION\
1293 007300    103    117    116         BASN1:  .ASCIZ  \CONTROLLER VERIFICATION\
1294 007330    123    125    102         BASN2:  .ASCIZ  \SUBSYSTEM VERIFICATION\
1295 007357    123    125    102         BASN3:  .ASCIZ  \SUBSYSTEM EXERCISER\
1296
1297 007403    045    124    000         BASL0:  .ASCIZ  \%T\
1298 007406    045    124    045         BASL1:  .ASCIZ  \%T%A - CONTROLLER AT %06\
1299 007437    045    124    045         BASL2:  .ASCIZ  \%T%A - CONTROLLER AT %06%A, DRIVE %D3\
1300
1301 007505    042    101    126         AMAV:   .ASCIZ  \"AVAILABLE"\
1302 007521    042    104    125         AMDU:   .ASCIZ  \"DUPLICATE UNIT #"\
1303 007544    042    101    103         AMAP:   .ASCIZ  \"ACCESS PATH"\
1304
1305 007562    042    101    102         OPCABO: .ASCIZ  \"ABORT"\                   ; ABORT COMMAND
1306 007572    042    101    103         OPCACC: .ASCIZ  \"ACCESS"\                  ; ACCESS COMMAND
1307 007603    042    101    126         OPCAVL: .ASCIZ  \"AVAILABLE"\               ; AVAILABLE COMMAND
1308 007617    042    103    117         OPCCCD: .ASCIZ  \"COMPARE CONTROLLER DATA"\ ; COMPARE CONTROLLER DATA COMMAND
1309 007651    042    103    117         OPCCMP: .ASCIZ  \"COMPARE HOST DATA"\       ; COMPARE HOST DATA COMMAND
1310 007675    042    104    105         OPCDAP: .ASCIZ  \"DETERMINE ACCESS PATH"\   ; DETERMINE ACCESS PATHS COMMAND
1311 007725    042    105    122         OPCERS: .ASCIZ  \"ERASE"\                   ; ERASE COMMAND
1312 007735    042    106    114         OPCFLU: .ASCIZ  \"FLUSH"\                   ; FLUSH COMMAND
1313 007745    042    107    105         OPCGCS: .ASCIZ  \"GET COMMAND STATUS"\      ; GET COMMAND STATUS COMMAND
1314 007772    042    107    105         OPCGUS: .ASCIZ  \"GET UNIT STATUS"\         ; GET UNIT STATUS COMMAND
1315 010014    042    117    116         OPCONL: .ASCIZ  \"ONLINE"\                  ; ONLINE COMMAND
```

```
1316 010025    042 122 105  OPCRD:  .ASCIZ \"READ"\                          ; READ COMMAND
1317 010034    042 122 105  OPCRPL: .ASCIZ \"REPLACE"\                       ; REPLACE COMMAND
1318 010046    042 123 105  OPCSCC: .ASCIZ \"SET CONTROLLER CHARACTERISTICS"\; SET CONTROLLER CHARACTERISTICS COMMAND
1319 010107    042 123 105  OPCSUC: .ASCIZ \"SET UNIT CHARACTERISTICS"\      ; SET UNIT CHARACTERISTICS COMMAND
1320 010142    042 127 122  OPCWR:  .ASCIZ \"WRITE"\                         ; WRITE COMMAND
1321 010152    042 115 101  OPCMRD: .ASCIZ \"MAINTENANCE READ"\             ; MAINTENANCE READ COMMAND
1322 010175    042 115 101  OPCMWR: .ASCIZ \"MAINTENANCE WRITE"\            ; MAINTENANCE WRITE COMMAND
1323
1324 010221    040 040 000  BLANK:  .ASCIZ \ \
1325 010224    101 122 105  INITWD: .ASCIZ \ARE YOU SURE CUSTOMER DATA CAN BE DESTROYED\
1326
1327 010300    045 116 000  CRLF:   .ASCIZ \%N\
1328 010303    045 101 040  RNTIM:  .ASCIZ \%A - RUN TIME %06%A:%Z2%A:%Z2\
1329 010341    045 116 045  DETHDR: .ASCIZ \%N%ASUBSYSTEM EXERCISER - CONTROLLER AT %06\
1330 010415    045 101 104  DETDUN: .ASCIZ \%ADETERMINISTIC PHASE DONE.%N\
1331 010453    045 116 045  TIMLIM: .ASCIZ \%N%ATEST TIME LIMIT REACHED - TEST STOPPING\
1332 010527    045 116 045  ERRLIM: .ASCIZ \%N%AUNIT %D3%A REACHED ERROR LIMIT - UNIT DROPPED FROM TEST%N\
1333 010625    045 116 045  T3WARN: .ASCIZ \%N%AMANUAL INTERVENTION NOT ALLOWED, USING DEFAULT PARAMETERS%N\
1334 010725    045 116 045  INITWA: .ASCIZ \%N%ACUSTOMER DATA WILL BE DESTROYED ON:%N\
1335 010767    045 123 065  INITWB: .ASCIZ \%S5%AUNIT%S3%ACONTROLLER%S3%ADRIVE%N\
1336 011044    045 123 066  INITWC: .ASCIZ \%S6%D2%S6%06%S6%D3%N\
1337 011071    045 101 116  NOCLOCK:.ASCIZ \%ANO LINE CLOCK AVAILABLE FOR TIMING EVENTS%N\
1338 011147    045 124 045  ATNMSG: .ASCIZ \%T%A ATTENTION MESSAGE RECEIVED%N\
1339 011211    045 101 124  X2:     .ASCIZ \%ATWO UNITS SELECT THE SAME DRIVE%N\
1340 011255    045 101 115  X3:     .ASCIZ \%AMORE THAN 4 DRIVES SELECTED ON THIS CONTROLLER%N\
1341 011340    045 101 116  X4A:    .ASCIZ \%ANOT ENOUGH MEMORY TO TEST THE UNITS SELECTED%N\
1342 011421    045 101 122  X4B:    .ASCIZ \%ARESTART PROGRAM AND TEST FEWER UNITS AT A TIME%N\
1343 011504    045 101 111  X5A:    .ASCIZ \%AINVALID BEGIN/END SET SPECIFIED%N\
1344 011550    045 101 102  X5B:    .ASCIZ \%ABEGIN/END SET SPECIFIES OUT-OF-RANGE LBN.%N\
1345 011626    045 101 115  X5C:    .ASCIZ \%AMAXIMUM LBN: %T%N\
1346 011652    045 101 104  X6:     .ASCIZ \%ADATA STRUCTURE CORRUPT AT LOCATION %06%A, RE-LOAD PROGRAM%N\
1347 011750    045 101 103  X14A:   .ASCIZ \%ACONTROLLER IS NOT SUPPORTED BY THIS DIAGNOSTIC PROGRAM.%N\
1348 012044    045 101 120  X14B:   .ASCIZ \%APROGRAM REQUIRES A UQSSP DISK TYPE CONTROLLER%N\
1349 012126    045 101 103  X14C:   .ASCIZ \%ACONTROLLER REPORTED MODEL CODE %C2%N\
1350 012175    045 101 123  X22A:   .ASCIZ \%ASTEP 1 BIT DID NOT SET IN SA REGISTER DURING INITIALIZATION%N\
1351 012273    045 101 123  X22B:   .ASCIZ \%ASTEP 1 BIT EXPECTED %06%N\
1352 012325    045 101 103  X23A:   .ASCIZ \%ACONTROLLER DID NOT CLEAR PORT COMMUNICATIONS AREA DURING INITIALIZATION%N\
1353 012441    045 101 066  X23C:   .ASCIZ \%A6 WORDS WERE TO BE CLEARED STARTING AT ADDRESS %06%N\
1354 012530    045 101 124  X23D:   .ASCIZ \%ATHE FOLLOWING WORDS WERE NOT CLEARED:%N\
1355 012602    045 123 066  X23E:   .ASCIZ \%S5%AADDRESS%S4%ACONTENTS%N\
1356 012636    045 123 067  X23F:   .ASCIZ \%S7%06%S5%06%N\
1357 012655    045 101 123  X24A:   .ASCIZ \%ASA REGISTER DID NOT GO TO 0 AFTER STEP 3 WRITE OF INITIALIZATION%N\
1358 012762    045 101 120  X24B:   .ASCIZ \%APURGE/POLL DIAGNOSTICS WERE REQUESTED%N\
1359 013034    045 101 111  X25:    .ASCIZ \%AINCORRECT DATA RETURNED IN SA REGISTER DURING INITIALIZATION%N\
1360 013135    045 101 104  X26:    .ASCIZ \%ADATA COMPARISON ERROR DURING WRAP MODE TEST%N\
1361 013215    045 101 123  X27:    .ASCIZ \%ASA REGISTER DID NOT CHANGE AFTER WRITING IT IN WRAP MODE TEST%N\
1362 013317    045 101 103  X31:    .ASCIZ \%ACONTROLLER TIMED OUT BY HOST, NO PROGRESS MADE ON COMMAND%N\
1363 013415    045 101 120  X32:    .ASCIZ \%APACKET RECEIVED FROM CONTROLLER WITH UNKNOWN ENDCODE%N\
1364 013505    045 101 105          .ASCIZ \%AENDCODE: %03%N\
1365 013526    045 101 120  X33:    .ASCIZ \%APACKET RECEIVED FROM CONTROLLER WITH UNKNOWN STATUS%N\
1366 013615    045 101 123          .ASCIZ \%ASTATUS: %03%N\
1367 013635    045 101 103  X34:    .ASCIZ \%ACONTROLLER DETECTED INVALID COMMAND PACKET%N\
1368 013713    045 101 106          .ASCIZ \%AFIELD AT OFFSET %D3%A IS BAD%N\
1369 013754    045 101 120  X35:    .ASCIZ \%APACKET RECEIVED FROM CONTROLLER SPECIFIES UNKNOWN DRIVE%N\
1370 014047    045 101 104          .ASCIZ \%ADRIVE UNIT NUMBER: %D3%N\
```

```
1371 014102      045      101      115  X37A:   .ASCIZ  \#AMEMORY ERROR TRYING TO READ CONTROLLER REGISTERS#N\
1372 014167      045      101      105  X37B:   .ASCIZ  \#AEITHER THE CONTROLLER CSR ADDRESS WAS INCORRECTLY SPECIFIED#N\
1373 014267      045      101      117  X37C:   .ASCIZ  \#AOR THERE IS A BUS OR CONTROLLER PROBLEM#N\
1374 014343      045      101      104  X38A:   .ASCIZ  \#ADATA COMPARE ERROR DURING DATA PATTERN VERIFICATION#N\
1375 014433      045      101      114  X38B:   .ASCIZ  \#ALBN: #T#A   BYTE OFFSET:  #06#N\
1376 014474      045      101      104  X38C:   .ASCIZ  \#ADATA EXPECTED: #06#A  DATA READ: #06#N\
1377 014545      045      101      104  X39A:   .ASCIZ  \#ADATA COMPARE ERROR DURING CONTROLLER MEMORY TEST#N\
1378 014632      045      101      103  X39B:   .ASCIZ  \#ACONTROLLER MEMORY ADDRESS: #06#N\
1379 014675      045      101      104  X39C:   .ASCIZ  \#ADATA WRITTEN: #06#A DATA READ: #06#N\
1380 014744      045      124      045  X41:    .ASCIZ  \#T#A MSCP OPERATION FAILED#N\
1381 015001      045      101      120  X42A:   .ASCIZ  \#APREVIOUS FATAL CONTROLLER ERROR CAUSED THE CONTROLLER AND#N\
1382 015077      045      124      124  X42B:   .ASCIZ  \#ATHE FOLLOWING DRIVES TO BE DROPPED:\
1383 015145      045      123      063  X42C:   .ASCIZ  \#S#D3\
1384 015154      045      101      120  X43:    .ASCIZ  \#APREVIOUS FATAL DRIVE ERROR CAUSED THIS DRIVE TO BE DROPPED#N\
1385 015253      045      101      105  X50A:   .ASCIZ  \#AERROR LOG RECEIVED - CONTROLLER ERROR#N\
1386 015325      045      101      105  X51A:   .ASCIZ  \#AERROR LOG RECEIVED - HOST MEMORY ACCESS ERROR#N\
1387 015407      045      101      110  X51B:   .ASCIZ  \#AHOST MEMORY ADDRESS: #T#N\
1388 015443      045      101      105  X52A:   .ASCIZ  \#AERROR LOG RECEIVED - DISK TRANSFER ERROR#N\
1389 015520      045      101      122  X52B:   .ASCIZ  \#ARETRY COUNT: #D3#A RECOVERY LEVEL: #D3#N\
1390 015573      045      101      105  X53A:   .ASCIZ  \#AERROR LOG RECEIVED - SDI ERROR#N\
1391 015636      045      101      123  X53B:   .ASCIZ  \#ASDI STATUS (R TO L): #06#S1#06#S1#06#S1#06#S1#06#S1#06#N\
1392 015731      045      101      105  X54A:   .ASCIZ  \#AERROR LOG RECEIVED - SMALL DISK ERROR#N\
1393 016003      045      101      103  X54B:   .ASCIZ  \#ACYLINDER: #T#N\
1394 016024      045      101      104  X60:    .ASCIZ  \#ADRIVE UNDER TEST BECAME UNIT-AVAILABLE#N\
1395 016077      045      101      104  X61:    .ASCIZ  \#ADUPLICATE UNIT NUMBER DETECTED BY CONTROLLER#N\
1396 016160      045      101      117  MSCFAI: .ASCIZ  \#AOPERATION FAILED#N\
1397 016205      045      101      117  MSGSUC: .ASCIZ  \#AOPERATION SUCCESSFUL#N\
1398 016236      045      101      117  MSGCON: .ASCIZ  \#AOPERATION CONTINUING#N\
1399 016267      045      101      102  MSGBLK: .ASCIZ  \#ABAD BLOCK REPORTED, LBN: #T#N\
1400 016326      045      101      101  MSGADU: .ASCIZ  \#AADDITIONAL BAD BLOCKS UNREPORTED#N\
1401 016373      045      101      105  MSGLOG: .ASCIZ  \#AERROR LOG GENERATED#N\
1402 016423      045      101      123  XSTA:   .ASCIZ  \#ASTATUS/EVENT CODE: #T#T#N\
1403 016457      045      101      111  XHW:    .ASCIZ  \#AINVALID ANSWER GIVEN TO THE HARDWARE QUESTIONS#N\
1404 016543      045      101      106  XFCE:   .ASCIZ  \#AFATAL ERROR REPORTED BY CONTROLLER#N\
1405 016612      045      101      115  XMSG1:  .ASCIZ  \#AMESSAGE BUFFER CONTAINS:#N\
1406 016647      045      123      062  XMG2:   .ASCIZ  \#S#D2\
1407 016656      045      101      104  XSAE:   .ASCIZ  \#ADATA EXPECTED:  #06#N\
1408 016705      045      101      123  XSA:    .ASCIZ  \#SA REGISTER CONTAINS: #06#N\
1409 016743      045      101      122  XFRU:   .ASCIZ  \#AREPLACE CONTROLLER PROCESSOR MODULE#N\
1410 017013      045      101      122  XFRUA:  .ASCIZ  \#AREPLACE CONTROLLER SDI MODULE#N\
1411
1412                                            .EVEN
1413
```

```
1415                                    .SBTTL  GLOBAL ERROR REPORT SECTION
1416
1417                                    ;++
1418                                    ;       THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS USED
1419                                    ;       BY MORE THAN ONE TEST TO OUTPUT ADDITIONAL ERROR INFORMATION.  PRINTB
1420                                    ;       (BASIC) AND PRINTX (EXTENDED) CALLS ARE USED TO CALL PRINT SERVICES.
1421                                    ;--
1422
1423 017056                            BGNMSG ERR002
     017056                            ERR002::
1424 017056   004737   054270                  CALL    CALR6           ; PRINT HEADER INFO
1425 017062                                    PRINTB  #XBHW           ; PRINT FIRST LINE
     017062   012746   016457                  MOV     #XBHW,-(SP)
     017066   012746   000001                  MOV     #1,-(SP)
     017072   010600                           MOV     SP,R0
     017074   104414                           TRAP    C$PNTB
     017076   062706   000004                  ADD     #4,SP
1426 017102                                    PRINTB  #X2             ; PRINT NEXT LINE
     017102   012746   011211                  MOV     #X2,-(SP)
     017106   012746   000001                  MOV     #1,-(SP)
     017112   010600                           MOV     SP,R0
     017114   104414                           TRAP    C$PNTB
     017116   062706   000004                  ADD     #4,SP
1427 017122                            ENDMSG
     017122                            L10002:
     017122   104423                           TRAP    C$MSG
1428
1429 017124                            BGNMSG ERR003
     017124                            ERR003::
1430 017124   004737   054136                  CALL    CALR5           ; PRINT HEADER INFO
1431 017130                                    PRINTB  #XBHW           ; PRINT FIRST LINE
     017130   012746   016457                  MOV     #XBHW,-(SP)
     017134   012746   000001                  MOV     #1,-(SP)
     017140   010600                           MOV     SP,R0
     017142   104414                           TRAP    C$PNTB
     017144   062706   000004                  ADD     #4,SP
1432 017150                                    PRINTB  #X3             ; PRINT NEXT LINE
     017150   012746   011255                  MOV     #X3,-(SP)
     017154   012746   000001                  MOV     #1,-(SP)
     017160   010600                           MOV     SP,R0
     017162   104414                           TRAP    C$PNTB
     017164   062706   000004                  ADD     #4,SP
1433 017170                            ENDMSG
     017170                            L10003:
     017170   104423                           TRAP    C$MSG
1434
1435 017172                            BGNMSG ERR004
     017172                            ERR004::
1436 017172   004737   054006                  CALL    CALR4           ; PRINT HEADER INFO
1437 017176                                    PRINTB  #X4A
     017176   012746   011340                  MOV     #X4A,-(SP)
     017202   012746   000001                  MOV     #1,-(SP)
     017206   010600                           MOV     SP,R0
     017210   104414                           TRAP    C$PNTB
     017212   062706   000004                  ADD     #4,SP
```

```
1438 017216                        PRINTB    #X4B
     017216    012746  011421      MOV       #X4B,-(SP)
     017222    012746  000001      MOV       #1,-(SP)
     017226    010600              MOV       SP,R0
     017230    104414              TRAP      C$PNTB
     017232    062706  000004      ADD       #4,SP
1439 017236                        ENDMSG
     017236                        L10004:
     017236    104423              TRAP      C$MSG
1440
1441 017240                        BGNMSG ERR005
     017240                        ERR005::
1442 017240    004737  054270      CALL      CALR6            ; PRINT HEADER INFO
1443 017244                        PRINTB    #X5A             ; PRINT FIRST LINE
     017244    012746  011504      MOV       #X5A,-(SP)
     017250    012746  000001      MOV       #1,-(SP)
     017254    010600              MOV       SP,R0
     017256    104414              TRAP      C$PNTB
     017260    062706  000004      ADD       #4,SP
1444 017264                        PRINTX    #X5B             ; PRINT NEXT LINE
     017264    012746  011550      MOV       #X5B,-(SP)
     017270    012746  000001      MOV       #1,-(SP)
     017274    010600              MOV       SP,R0
     017276    104415              TRAP      C$PNTX
     017300    062706  000004      ADD       #4,SP
1445 017304                        PRINTX    #X5C,#TEMP       ; PRINT NEXT LINE
     017304    012746  002212      MOV       #TEMP,-(SP)
     017310    012746  011626      MOV       #X5C,-(SP)
     017314    012746  000002      MOV       #2,-(SP)
     017320    010600              MOV       SP,R0
     017322    104415              TRAP      C$PNTX
     017324    062706  000006      ADD       #6,SP
1446 017330                        ENDMSG
     017330                        L10005:
     017330    104423              TRAP      C$MSG
1447
1448 017332                        BGNMSG ERR006
     017332                        ERR006::
1449 017332    004737  054006      CALL      CALR4            ; PRINT HEADER INFO
1450 017336                        PRINTB    #X6,R1
     017336    010146              MOV       R1,-(SP)
     017340    012746  011652      MOV       #X6,-(SP)
     017344    012746  000002      MOV       #2,-(SP)
     017350    010600              MOV       SP,R0
     017352    104414              TRAP      C$PNTB
     017354    062706  000006      ADD       #6,SP
1451 017360                        ENDMSG
     017360                        L10006:
     017360    104423              TRAP      C$MSG
1452
1453 017362                        BGNMSG ERR014                ; INVALID CONTROLLER TYPE
     017362                        ERR014::
1454 017362    004737  054136      CALL      CALR5            ; PRINT HEADER INFO
1455 017366                        PRINTB    #X14A
     017366    012746  011750      MOV       #X14A,-(SP)
```

```
      017372  012746  000001        MOV     #1,-(SP)
      017376  010600                MOV     SP,R0
      017400  104414                TRAP    C$PNTB
      017402  062706  000004        ADD     #4,SP
 1456 017406                        PRINTX  #X14B
      017406  012746  012044        MOV     #X14B,-(SP)
      017412  012746  000001        MOV     #1,-(SP)
      017416  010600                MOV     SP,R0
      017420  104415                TRAP    C$PNTX
      017422  062706  000004        ADD     #4,SP
 1457 017426                        PRINTX  #X14C,R3
      017426  010346                MOV     R3,-(SP)
      017430  012746  012126        MOV     #X14C,-(SP)
      017434  012746  000002        MOV     #2,-(SP)
      017440  010600                MOV     SP,R0
      017442  104415                TRAP    C$PNTX
      017444  062706  000006        ADD     #6,SP
 1458 017450                ENDMSG
      017450                L10007:
      017450  104423                TRAP    C$MSG
 1459
 1460 017452                BGNMSG ERR021
      017452                ERR021::
 1461 017452  004737  054136        CALL    CALR5           ; PRINT HEADER INFO
 1462 017456  004737  053762        CALL    CALR3           ; PRINT FATAL CONTROLLER ERROR
 1463 017462  004737  054446        CALL    CALR8           ; PRINT SA CONTENTS
 1464 017466  032702  001000        BIT     #1000,R2
 1465 017472  001411                BEQ     ERR21A
 1466 017474                        PRINTB  #XFRUA
      017474  012746  017013        MOV     #XFRUA,-(SP)
      017500  012746  000001        MOV     #1,-(SP)
      017504  010600                MOV     SP,R0
      017506  104414                TRAP    C$PNTB
      017510  062706  000004        ADD     #4,SP
 1467 017514  000402                BR      ERR21E
 1468 017516                ERR21A:
 1469 017516  004737  054424        CALL    CALR7           ; PRINT REPLACE FRU
 1470 017522                ERR21E:
 1471 017522                ENDMSG
      017522                L10010:
      017522  104423                TRAP    C$MSG
 1472
 1473 017524                BGNMSG ERR022
      017524                ERR022::
 1474 017524  004737  054136        CALL    CALR5           ; PRINT HEADER INFO
 1475 017530                        PRINTB  #X22A           ; PRINT ERROR MESSAGE
      017530  012746  012175        MOV     #X22A,-(SP)
      017534  012746  000001        MOV     #1,-(SP)
      017540  010600                MOV     SP,R0
      017542  104414                TRAP    C$PNTB
      017544  062706  000004        ADD     #4,SP
 1476 017550                        PRINTX  #X22B,CNTRSD    ;
      017550  013746  002354        MOV     CNTRSD,-(SP)
      017554  012746  012273        MOV     #X22B,-(SP)
      017560  012746  000002        MOV     #2,-(SP)
```

```
          017564  010600                      MOV     SP,R0
          017566  104415                      TRAP    C$PNTX
          017570  062706  000006              ADD     #6,SP
     1477 017574  004737  054446              CALL    CALR8           ; PRINT SA CONTENTS
     1478 017600  004737  054424              CALL    CALR7           ; PRINT REPLACE FRU
     1479 017604                      ENDMSG
          017604                      L10011:
          017604  104423                      TRAP    C$MSG
     1480
     1481 017606                      BGNMSG ERRO23
          017606                      ERRO23::
     1482 017606  004737  054136              CALL    CALR5           ; PRINT HEADER INFO
     1483 017612                              PRINTB  #X23A
          017612  012746  012325              MOV     #X23A,-(SP)
          017616  012746  000001              MOV     #1,-(SP)
          017622  010600                      MOV     SP,R0
          017624  104414                      TRAP    C$PNTB
          017626  062706  000004              ADD     #4,SP
     1484 017632  010500                      MOV     R5,R0
     1485 017634  062700  000106              ADD     #C.CINT,R0
     1486 017640                              PRINTX  #X23C,R0
          017640  010046                      MOV     R0,-(SP)
          017642  012746  012441              MOV     #X23C,-(SP)
          017646  012746  000002              MOV     #2,-(SP)
          017652  010600                      MOV     SP,R0
          017654  104415                      TRAP    C$PNTX
          017656  062706  000006              ADD     #6,SP
     1487 017662                              PRINTX  #X23D
          017662  012746  012530              MOV     #X23D,-(SP)
          017666  012746  000001              MOV     #1,-(SP)
          017672  010600                      MOV     SP,R0
          017674  104415                      TRAP    C$PNTX
          017676  062706  000004              ADD     #4,SP
     1488 017702                              PRINTX  #X23E
          017702  012746  012602              MOV     #X23E,-(SP)
          017706  012746  000001              MOV     #1,-(SP)
          017712  010600                      MOV     SP,R0
          017714  104415                      TRAP    C$PNTX
          017716  062706  000004              ADD     #4,SP
     1489 017722  005742                      TST     -(R2)
     1490 017724  005712              ERR23A:  TST     (R2)
     1491 017726  001412                      BEQ     ERR23B
     1492 017730                              PRINTX  #X23F,R2,(R2)
          017730  011246                      MOV     (R2),-(SP)
          017732  010246                      MOV     R2,-(SP)
          017734  012746  012636              MOV     #X23F,-(SP)
          017740  012746  000003              MOV     #3,-(SP)
          017744  010600                      MOV     SP,R0
          017746  104415                      TRAP    C$PNTX
          017750  062706  000010              ADD     #10,SP
     1493 017754  005722              ERR23B:  TST     (R2)+
     1494 017756  005303                      DEC     R3
     1495 017760  001361                      BNE     ERR23A
     1496 017762  004737  054424              CALL    CALR7           ; PRINT REPLACE FRU
     1497 017766                      ENDMSG
```

```
        017766                  L10012:
        017766  104423                  TRAP    C$MSG
  1498
  1499 017770                  BGNMSG ERR024
        017770                  ERR024::
  1500 017770  004737  054136           CALL    CALR5           ; PRINT HEADER INFO
  1501 017774                           PRINTB  #X24A           ; PRINT ERROR MESSAGE
        017774  012746  012655          MOV     #X24A,-(SP)
        020000  012746  000001          MOV     #1,-(SP)
        020004  010600                  MOV     SP,R0
        020006  104414                  TRAP    C$PNTB
        020010  062706  000004          ADD     #4,SP
  1502 020014                           PRINTB  #X24B           ;
        020014  012746  012762          MOV     #X24B,-(SP)
        020020  012746  000001          MOV     #1,-(SP)
        020024  010600                  MOV     SP,R0
        020026  104414                  TRAP    C$PNTB
        020030  062706  000004          ADD     #4,SP
  1503 020034  004737  054446           CALL    CALR8           ; PRINT SA CONTENTS
  1504 020040  004737  054424           CALL    CALR7           ; PRINT REPLACE FRU
  1505 020044                  ENDMSG
        020044                  L10013:
        020044  104423                  TRAP    C$MSG
  1506
  1507 020046                  BGNMSG ERR025
        020046                  ERR025::
  1508 020046  004737  054136           CALL    CALR5           ; PRINT HEADER INFO
  1509 020052                           PRINTB  #X25            ; PRINT ERROR MESSAGE
        020052  012746  013034          MOV     #X25,-(SP)
        020056  012746  000001          MOV     #1,-(SP)
        020062  010600                  MOV     SP,R0
        020064  104414                  TRAP    C$PNTB
        020066  062706  000004          ADD     #4,SP
  1510 020072                           PRINTX  #X$AE,R1        ; PRINT EXPECTED DATA
        020072  010146                  MOV     R1,-(SP)
        020074  012746  016656          MOV     #X$AE,-(SP)
        020100  012746  000002          MOV     #2,-(SP)
        020104  010600                  MOV     SP,R0
        020106  104415                  TRAP    C$PNTX
        020110  062706  000006          ADD     #6,SP
  1511 020114  004737  054446           CALL    CALR8           ; PRINT SA CONTENTS
  1512 020120  004737  054424           CALL    CALR7           ; PRINT REPLACE FRU
  1513 020124                  ENDMSG
        020124                  L10014:
        020124  104423                  TRAP    C$MSG
  1514
  1515 020126                  BGNMSG ERR026
        020126                  ERR026::
  1516 020126  004737  054136           CALL    CALR5           ; PRINT HEADER INFO
  1517 020132                           PRINTB  #X26            ; PRINT ERROR MESSAGE
        020132  012746  013135          MOV     #X26,-(SP)
        020136  012746  000001          MOV     #1,-(SP)
        020142  010600                  MOV     SP,R0
        020144  104414                  TRAP    C$PNTB
        020146  062706  000004          ADD     #4,SP
```

```
1518 020152                         PRINTX  #XSAE,WCHNGD        ; PRINT EXPECTED DATA
     020152  013746  002356         MOV     WCHNGD,-(SP)
     020156  012746  016656         MOV     #XSAE,-(SP)
     020162  012746  000002         MOV     #2,-(SP)
     020166  010600                 MOV     SP,R0
     020170  104415                 TRAP    C$PNTX
     020172  062706  000006         ADD     #6,SP
1519 020176  004737  054446         CALL    CALR8               ; PRINT SA CONTENTS
1520 020202  004737  054424         CALL    CALR7               ; PRINT REPLACE FRU
1521 020206                  ENDMSG
     020206             L10015:
     020206  104423                 TRAP    C$MSG

1522
1523 020210          BGNMSG ERR027
     020210             ERR027::
1524 020210  004737  054136         CALL    CALR5               ; PRINT HEADER INFO
1525 020214                         PRINTB  #X27                ; PRINT ERROR MESSAGE
     020214  012746  013215         MOV     #X27,-(SP)
     020220  012746  000001         MOV     #1,-(SP)
     020224  010600                 MOV     SP,R0
     020226  104414                 TRAP    C$PNTB
     020230  062706  000004         ADD     #4,SP
1526 020234  004737  054446         CALL    CALR8               ; PRINT SA CONTENTS
1527 020240  004737  054424         CALL    CALR7               ; PRINT REPLACE FRU
1528 020244                  ENDMSG
     020244             L10016:
     020244  104423                 TRAP    C$MSG

1529
1530 020246          BGNMSG ERR030
     020246             ERR030::
1531 020246  004737  054136         CALL    CALR5               ; PRINT HEADER INFO
1532 020252  004737  053762         CALL    CALR3               ; PRINT FATAL CONTROLLER ERROR
1533 020256  004737  054446         CALL    CALR8               ; PRINT SA CONTENTS
1534 020262                  ENDMSG
     020262             L10017:
     020262  104423                 TRAP    C$MSG

1535
1536 020264          BGNMSG ERR031
     020264             ERR031::
1537 020264  004737  054136         CALL    CALR5               ; PRINT HEADER INFO
1538 020270                         PRINTB  #X31
     020270  012746  013317         MOV     #X31,-(SP)
     020274  012746  000001         MOV     #1,-(SP)
     020300  010600                 MOV     SP,R0
     020302  104414                 TRAP    C$PNTB
     020304  062706  000004         ADD     #4,SP
1539 020310                  ENDMSG
     020310             L10020:
     020310  104423                 TRAP    C$MSG
1540
1541 020312          BGNMSG ERR032                              ; UNKNOWN ENDCODE ERROR
     020312             ERR032::
1542 020312  004737  054136         CALL    CALR5               ; PRINT HEADER INFO
1543 020316                         PRINTB  #X32,R3             ; PRINT MESSAGE
     020316  010346                 MOV     R3,-(SP)
```

```
          020320  012746  013415          MOV     #X32,-(SP)
          020324  012746  000002          MOV     #2,-(SP)
          020330  010600                  MOV     SP,R0
          020332  104414                  TRAP    C$PNTB
          020334  062706  000006          ADD     #6,SP
     1544 020340  004737  022616          CALL    MSGPKT              ; DUMP PACKET CONTENTS
     1545 020344                  ENDMSG
          020344                  L10021:
          020344  104423                  TRAP    C$MSG
     1546
     1547 020346                  BGNMSG ERR033                      ; UNKNOWN COMMAND STATUS ERROR
          020346                  ERR033::
     1548 020346  004737  054136          CALL    CALR5              ; PRINT HEADER INFO
     1549 020352                          PRINTX  #X33,R2            ; PRINT MESSAGE
          020352  010246                  MOV     R2,-(SP)
          020354  012746  013526          MOV     #X33,-(SP)
          020360  012746  000002          MOV     #2,-(SP)
          020364  010600                  MOV     SP,R0
          020366  104415                  TRAP    C$PNTX
          020370  062706  000006          ADD     #6,SP
     1550 020374  004737  022616          CALL    MSGPKT             ; DUMP END MESSAGE CONTENTS
     1551 020400                  ENDMSG
          020400                  L10022:
          020400  104423                  TRAP    C$MSG
     1552
     1553 020402                  BGNMSG ERR034                      ; INVALID COMMAND PACKET ERROR
          020402                  ERR034::
     1554 020402  004737  054136          CALL    CALR5              ; PRINT HEADER INFO
     1555 020406  016504  000104          MOV     C.RHDR(R5),R4      ; GET POINTER TO R.E.
     1556 020412  016403  000020          MOV     P.STS+RE.RP(R4),R3 ; GET STATUS WORD
     1557 020416  000303                  SWAB    R3                 ; GET HIGH
     1558 020420  042703  177400          BIC     #^C<377>,R3        ;    BYTE
     1559 020424                          PRINTX  #X34,R3            ; PRINT MESSAGE
          020424  010346                  MOV     R3,-(SP)
          020426  012746  013635          MOV     #X34,-(SP)
          020432  012746  000002          MOV     #2,-(SP)
          020436  010600                  MOV     SP,R0
          020440  104415                  TRAP    C$PNTX
          020442  062706  000006          ADD     #6,SP
     1560 020446  004737  022616          CALL    MSGPKT             ; DUMP END MESSAGE CONTENTS
     1561 020452                  ENDMSG
          020452                  L10023:
          020452  104423                  TRAP    C$MSG
     1562
     1563 020454                  BGNMSG ERR035                      ; UNKNOWN DRIVE ERROR
          020454                  ERR035::
     1564 020454  004737  054136          CALL    CALR5              ; PRINT HEADER INFO
     1565 020460                          PRINTB  #X35,R1            ; PRINT ERROR MESSAGE
          020460  010146                  MOV     R1,-(SP)
          020462  012746  013754          MOV     #X35,-(SP)
          020466  012746  000002          MOV     #2,-(SP)
          020472  010600                  MOV     SP,R0
          020474  104414                  TRAP    C$PNTB
          020476  062706  000006          ADD     #6,SP
     1566 020502  004737  022616          CALL    MSGPKT             ; DUMP PACKET CONTENTS
```

```
1567 020506              ENDMSG
     020506              L10024:
     020506    104423            TRAP    C$MSG
1568
1569 020510              BGNMSG ERR037                          ; NXM ERROR
     020510              ERR037::
1570 020510  004737 054136       CALL    CALR5                 ; PRINT HEADER INFO
1571 020514                      PRINTB  #X37A                 ; PRINT
     020514  012746 014102       MOV     #X37A,-(SP)
     020520  012746 000001       MOV     #1,-(SP)
     020524  010600              MOV     SP,R0
     020526  104414              TRAP    C$PNTB
     020530  062706 000004       ADD     #4,SP
1572 020534                      PRINTX  #X37B                 ;        ERROR
     020534  012746 014167       MOV     #X37B,-(SP)
     020540  012746 000001       MOV     #1,-(SP)
     020544  010600              MOV     SP,R0
     020546  104415              TRAP    C$PNTX
     020550  062706 000004       ADD     #4,SP
1573 020554                      PRINTX  #X37C                 ;              MESSAGE
     020554  012746 014267       MOV     #X37C,-(SP)
     020560  012746 000001       MOV     #1,-(SP)
     020564  010600              MOV     SP,R0
     020566  104415              TRAP    C$PNTX
     020570  062706 000004       ADD     #4,SP
1574 020574              ENDMSG
     020574              L10025:
     020574  104423            TRAP    C$MSG
1575
1576 020576              BGNMSG ERR038                          ; MSCP PATTERN CHECK ERROR
     020576              ERR038::
1577 020576  012704 004370       MOV     #BUFBA,R4             ; CREATE CHARACTER STRING
1578 020602  004737 047552       CALL    BLDSTR                ;    CONTAINING LBN
1579 020606  004737 054270       CALL    CALR6                 ; PRINT HEADER INFO
1580 020612                      PRINTB  #X38A                 ; PRINT
     020612  012746 014343       MOV     #X38A,-(SP)
     020616  012746 000001       MOV     #1,-(SP)
     020622  010600              MOV     SP,R0
     020624  104414              TRAP    C$PNTB
     020626  062706 000004       ADD     #4,SP
1581 020632                      PRINTX  #X38B,#TEMP,BCLO      ;   ERROR
     020632  013746 004406       MOV     BCLO,-(SP)
     020636  012746 002212       MOV     #TEMP,-(SP)
     020642  012746 014433       MOV     #X38B,-(SP)
     020646  012746 000003       MOV     #3,-(SP)
     020652  010600              MOV     SP,R0
     020654  104415              TRAP    C$PNTX
     020656  062706 000010       ADD     #10,SP
1582 020662                      PRINTX  #X38C,PCGOOD,PCBAD    ;       MESSAGE
     020662  013746 004366       MOV     PCBAD,-(SP)
     020666  013746 004364       MOV     PCGOOD,-(SP)
     020672  012746 014474       MOV     #X38C,-(SP)
     020676  012746 000003       MOV     #3,-(SP)
     020702  010600              MOV     SP,R0
     020704  104415              TRAP    C$PNTX
```

```
           020706   062706   000010              ADD      #10,SP
      1583 020712                       ENDMSG
           020712                       L10026:
           020712   104423                        TRAP     C$MSG
      1584
      1585 020714                       BGNMSG ERRO39                        ; MAINT PATTERN CHECK ERROR
           020714                       ERRO39::
      1586 020714   004737   054136              CALL     CALR5              ; PRINT HEADER INFO
      1587 020720                               PRINTB   #X39A              ; PRINT
           020720   012746   014545              MOV      #X39A,-(SP)
           020724   012746   000001              MOV      #1,-(SP)
           020730   010600                       MOV      SP,R0
           020732   104414                       TRAP     C$PNTB
           020734   062706   000004              ADD      #4,SP
      1588 020740                               PRINTX   #X39B,BCLO         ;   ERROR
           020740   013746   004406              MOV      BCLO,-(SP)
           020744   012746   014632              MOV      #X39B,-(SP)
           020750   012746   000002              MOV      #2,-(SP)
           020754   010600                       MOV      SP,R0
           020756   104415                       TRAP     C$PNTX
           020760   062706   000006              ADD      #6,SP
      1589 020764                               PRINTX   #X39C,PCGOOD,PCBAD ;     MESSAGE
           020764   013746   004366              MOV      PCBAD,-(SP)
           020770   013746   004364              MOV      PCGOOD,-(SP)
           020774   012746   014675              MOV      #X39C,-(SP)
           021000   012746   000003              MOV      #3,-(SP)
           021004   010600                       MOV      SP,R0
           021006   104415                       TRAP     C$PNTX
           021010   062706   000010              ADD      #10,SP
      1590 021014                       ENDMSG
           021014                       L10027:
           021014   104423                        TRAP     C$MSG
      1591
      1592 021016                       BGNMSG ERRO40
           021016                       ERRO40::
      1593 021016   004737   054136              CALL     CALR5              ; PRINT HEADER INFO
      1594 021022                               PRINTB   #X41,MSCPOP        ; MSCP OPERATION FAILED
           021022   013746   002314              MOV      MSCPOP,-(SP)
           021026   012746   014744              MOV      #X41,-(SP)
           021032   012746   000002              MOV      #2,-(SP)
           021036   010600                       MOV      SP,R0
           021040   104414                       TRAP     C$PNTB
           021042   062706   000006              ADD      #6,SP
      1595 021046                               PRINTB   #XSTA,STAMS0,STAMS1 ; PRINT STATUS
           021046   013746   002320              MOV      STAMS1,-(SP)
           021052   013746   002316              MOV      STAMS0,-(SP)
           021056   012746   016423              MOV      #XSTA,-(SP)
           021062   012746   000003              MOV      #3,-(SP)
           021066   010600                       MOV      SP,R0
           021070   104414                       TRAP     C$PNTB
           021072   062706   000010              ADD      #10,SP
      1596 021076   132764   000200   000017     BITB     #EF.BBR,P.FLGS+RE.RP(R4);
      1597 021104   001422                       BEQ      1$                 ;
      1598 021106                               PUSH     R4                 ; CONVERT
      1599 021110   016504   000104              MOV      C.RHDR(R5),R4      ;    LBN TO
```

```
1600 021114  062704  000042                  ADD      #<RE.RP+P.FBBK>,R4      ;       ASCII
1601 021120  004737  047552                  CALL     BLDSTR                 ;        STRING
1602 021124                                  POP      R4
1603 021126                                  PRINTX   #MSGBBR,#TEMP          ; "BAD BLOCK REPORTED"
     021126  012746  002212                  MOV      #TEMP,-(SP)
     021132  012746  016267                  MOV      #MSGBBR,-(SP)
     021136  012746  000002                  MOV      #2,-(SP)
     021142  010600                          MOV      SP,RO
     021144  104415                          TRAP     C#PNTX
     021146  062706  000006                  ADD      #6,SP
1604 021152  132764  000100  000017  1$:     BITB     #EF.BBU,P.FLGS+RE.RP(R4);
1605 021160  001410                          BEQ      2$                    ;
1606 021162                                  PRINTX   #MSGBBU               ; "BAD BLOCK UNREPORTED"
     021162  012746  016326                  MOV      #MSGBBU,-(SP)
     021166  012746  000001                  MOV      #1,-(SP)
     021172  010600                          MOV      SP,RO
     021174  104415                          TRAP     C#PNTX
     021176  062706  000004                  ADD      #4,SP
1607 021202  132764  000040  000017  2$:     BITB     #EF.LOG,P.FLGS+RE.RP(R4);
1608 021210  001410                          BEQ      3$                    ;
1609 021212                                  PRINTX   #MSGLOG               ; "ERROR LOG GENERATED"
     021212  012746  016373                  MOV      #MSGLOG,-(SP)
     021216  012746  000001                  MOV      #1,-(SP)
     021222  010600                          MOV      SP,RO
     021224  104415                          TRAP     C#PNTX
     021226  062706  000004                  ADD      #4,SP
1610 021232  004737  022616          3$:     CALL     MSGPKT                ;
1611 021236                                  ENDMSG
     021236                          L10030:
     021236  104423                          TRAP     C#MSG
1612
1613 021240                                  BGNMSG ERRO41
     021240                          ERRO41::
1614 021240  004737  054270                  CALL     CALR6                 ; PRINT HEADER INFO
1615 021244                                  PRINTB   #X41,MSCPOP           ; MSCP OPERATION FAILED
     021244  013746  002314                  MOV      MSCPOP,-(SP)
     021250  012746  014744                  MOV      #X41,-(SP)
     021254  012746  000002                  MOV      #2,-(SP)
     021260  010600                          MOV      SP,RO
     021262  104414                          TRAP     C#PNTB
     021264  062706  000006                  ADD      #6,SP
1616 021270                                  PRINTB   #XSTA,STAMSO,STAMS1   ; PRINT STATUS
     021270  013746  002320                  MOV      STAMS1,-(SP)
     021274  013746  002316                  MOV      STAMSO,-(SP)
     021300  012746  016423                  MOV      #XSTA,-(SP)
     021304  012746  000003                  MOV      #3,-(SP)
     021310  010600                          MOV      SP,RO
     021312  104414                          TRAP     C#PNTB
     021314  062706  000010                  ADD      #10,SP
1617 021320  132764  000200  000017          BITB     #EF.BBR,P.FLGS+RE.RP(R4);
1618 021326  001422                          BEQ      1$
1619 021330                                  PUSH     R4                    ; CONVERT
1620 021332  016504  000104                  MOV      C.RHDR(R5),R4         ;   LBN TO
1621 021336  062704  000042                  ADD      #<RE.RP+P.FBBK>,R4    ;    ASCII
1622 021342  004737  047552                  CALL     BLDSTR                ;     STRING
```

```
1623 021346                                      POP     R4                      ;
1624 021350                                      PRINTX  #MSGBBR,#TEMP           ; "BAD BLOCK REPORTED"
     021350   012746   002212                    MOV     #TEMP,-(SP)
     021354   012746   016267                    MOV     #MSGBBR,-(SP)
     021360   012746   000002                    MOV     #2,-(SP)
     021364   010600                             MOV     SP,R0
     021366   104415                             TRAP    C$PNTX
     021370   062706   000006                    ADD     #6,SP
1625 021374   132764   000100   000017  1$:      BITB    #EF.BBU,P.FLGS+RE.RP(R4);
1626 021402   001410                             BEQ     2$                      ;
1627 021404                                      PRINTX  #MSGBBU                 ; "BAD BLOCK UNREPORTED"
     021404   012746   016326                    MOV     #MSGBBU,-(SP)
     021410   012746   000001                    MOV     #1,-(SP)
     021414   010600                             MOV     SP,R0
     021416   104415                             TRAP    C$PNTX
     021420   062706   000004                    ADD     #4,SP
1628 021424   132764   000040   000017  2$:      BITB    #EF.LOG,P.FLGS+RE.RP(R4);
1629 021432   001410                             BEQ     3$                      ;
1630 021434                                      PRINTX  #MSGLOG                 ; "ERROR LOG GENERATED"
     021434   012746   016373                    MOV     #MSGLOG,-(SP)
     021440   012746   000001                    MOV     #1,-(SP)
     021444   010600                             MOV     SP,R0
     021446   104415                             TRAP    C$PNTX
     021450   062706   000004                    ADD     #4,SP
1631 021454   004737   022616          3$:       CALL    MSGPKT                  ;
1632 021460                                      ENDMSG
     021460                                     L10031:
     021460   104423                             TRAP    C$MSG

1633
1634 021462                                      BGNMSG  ERRO42                  ; CONTROLLER DROPPED ERROR
     021462                                     ERRO42::
1635 021462   004737   054136                    CALL    CALR5                   ; PRINT HEADER INFO
1636 021466                                      PRINTB  #X42A                   ;
     021466   012746   015001                    MOV     #X42A,-(SP)
     021472   012746   000001                    MOV     #1,-(SP)
     021476   010600                             MOV     SP,R0
     021500   104414                             TRAP    C$PNTB
     021502   062706   000004                    ADD     #4,SP
1637 021506                                      PRINTB  #X42B                   ;
     021506   012746   015077                    MOV     #X42B,-(SP)
     021512   012746   000001                    MOV     #1,-(SP)
     021516   010600                             MOV     SP,R0
     021520   104414                             TRAP    C$PNTB
     021522   062706   000004                    ADD     #4,SP
1638 021526   005065   000042                    CLR     C.NEXT(R5)              ; INIT DRIVE COUNT
1639 021532   004737   045526          1$:       CALL    GTNXTD                  ; GET POINTER TO NEXT DRIVE TABLE
1640 021536   001422                             BEQ     10$                     ; IF NO MORE DRIVES, EXIT
1641 021540   052760   100000   000002           BIS     #DT.AVL,D.UNIT(R0)      ; DISABLE DRIVE
1642 021546                                      PRINTB  #X42C,(R0)              ; PRINT DRIVE NUMBER
     021546   011046                             MOV     (R0),-(SP)
     021550   012746   015145                    MOV     #X42C,-(SP)
     021554   012746   000002                    MOV     #2,-(SP)
     021560   010600                             MOV     SP,R0
     021562   104414                             TRAP    C$PNTB
     021564   062706   000006                    ADD     #6,SP
```

```
1643 021570  005265  000042                INC    C.NEXT(R5)        ; POINT TO NEXT DRIVE
1644 021574  022765  000003  000042        CMP    #3.,C.NEXT(R5)    ; IF DRIVE COUNT <= 3,
1645 021602  002353                         BGE    1$                ;    GET NEXT DRIVE
1646 021604                       10$:      PRINTB #CRLF             ; TERMINATE LINE
     021604  012746  010300                MOV    #CRLF,-(SP)
     021610  012746  000001                MOV    #1,-(SP)
     021614  010600                         MOV    SP,R0
     021616  104414                         TRAP   C$PNTB
     021620  062706  000004                ADD    #4,SP
1647 021624                       ENDMSG
     021624                       L10032:
     021624  104423                         TRAP   C$MSG
1648
1649 021626                       BGNMSG ERRO43            ; DRIVE DROPPED ERROR
     021626                       ERRO43::
1650 021626  004737  054270                CALL   CALR6             ; PRINT HEADER INFO
1651 021632                                 PRINTB #X43             ; PRINT ERROR MESSAGE
     021632  012746  015154                MOV    #X43,-(SP)
     021636  012746  000001                MOV    #1,-(SP)
     021642  010600                         MOV    SP,R0
     021644  104414                         TRAP   C$PNTB
     021646  062706  000004                ADD    #4,SP
1652 021652                       ENDMSG
     021652                       L10033:
     021652  104423                         TRAP   C$MSG
1653
1654 021654                       BGNMSG ERRO50
     021654                       ERRO50::
1655 021654  004737  054136                CALL   CALR5             ; PRINT HEADER INFO
1656 021660                                 PRINTB #X50A            ; PRINT ERROR LOG RECEIVED
     021660  012746  015253                MOV    #X50A,-(SP)
     021664  012746  000001                MOV    #1,-(SP)
     021670  010600                         MOV    SP,R0
     021672  104414                         TRAP   C$PNTB
     021674  062706  000004                ADD    #4,SP
1657 021700  004737  022334                CALL   ERLSTA            ; DO COMMON STUFF
1658 021704                       ENDMSG
     021704                       L10034:
     021704  104423                         TRAP   C$MSG
1659
1660 021706                       BGNMSG ERRO51
     021706                       ERRO51::
1661 021706  004737  054136                CALL   CALR5             ; PRINT HEADER INFO
1662 021712                                 PRINTB #X51A            ; PRINT ERROR LOG RECEIVED
     021712  012746  015325                MOV    #X51A,-(SP)
     021716  012746  000001                MOV    #1,-(SP)
     021722  010600                         MOV    SP,R0
     021724  104414                         TRAP   C$PNTB
     021726  062706  000004                ADD    #4,SP
1663 021732                                 PUSH   R4                ; CONVERT
1664 021734  016504  000104                MOV    C.RHDR(R5),R4     ;   ADDRESS
1665 021740  062704  000036                ADD    #<RE.RP+L.BADR>,R4 ;    TO
1666 021744  004737  047552                CALL   BLDSTR            ;      ASCII
1667 021750                                 POP    R4                ;       STRING
1668 021752                                 PRINTB #X51B,#TEMP      ;
```

```
        021752  012746  002212          MOV     #TEMP,-(SP)
        021756  012746  015407          MOV     #X51B,-(SP)
        021762  012746  000002          MOV     #2,-(SP)
        021766  010600                  MOV     SP,R0
        021770  104414                  TRAP    C#PNTB
        021772  062706  000006          ADD     #6,SP
1669    021776  004737  022334          CALL    ERLSTA              ; DO COMMON STUFF
1670    022002                  ENDMSG
        022002                  L10035:
        022002  104423                  TRAP    C#MSG
1671
1672    022004                  BGNMSG ERRO52
        022004                  ERRO52::
1673    022004  004737  054270          CALL    CALR6              ; PRINT HEADER INFO
1674    022010                          PRINTB  #X52A              ; PRINT ERROR LOG RECEIVED
        022010  012746  015443          MOV     #X52A,-(SP)
        022014  012746  000001          MOV     #1,-(SP)
        022020  010600                  MOV     SP,R0
        022022  104414                  TRAP    C#PNTB
        022024  062706  000004          ADD     #4,SP
1675    022030                          PUSH    <R3,R4>
1676    022034  016504  000104          MOV     C.RHDR(R5),R4      ; GET
1677    022040  062704  000050          ADD     #<RE.RP+L.LVL>,R4  ;   LEVEL &
1678    022044  010403                  MOV     R4,R3             ;     RETRIES
1679    022046  005203                  INC     R3                ;
1680    022050  111404                  MOVB    (R4),R4           ;
1681    022052  042704  177400          BIC     #^C<377>,R4       ;
1682    022056  111303                  MOVB    (R3),R3           ;
1683    022060  042703  177400          BIC     #^C<377>,R3       ;
1684    022064                          PRINTB  #X52B,R3,R4        ;
        022064  010446                  MOV     R4,-(SP)
        022066  010346                  MOV     R3,-(SP)
        022070  012746  015520          MOV     #X52B,-(SP)
        022074  012746  000003          MOV     #3,-(SP)
        022100  010600                  MOV     SP,R0
        022102  104414                  TRAP    C#PNTB
        022104  062706  000010          ADD     #10,SP
1685    022110                          POP     <R4,R3>
1686    022114  004737  022334          CALL    ERLSTA             ; DO COMMON STUFF
1687    022120                  ENDMSG
        022120                  L10036:
        022120  104423                  TRAP    C#MSG
1688
1689    022122                  BGNMSG ERRO53
        022122                  ERRO53::
1690    022122  004737  054270          CALL    CALR6              ; PRINT HEADER INFO
1691    022126                          PRINTB  #X53A              ; PRINT ERROR LOG RECEIVED
        022126  012746  015573          MOV     #X53A,-(SP)
        022132  012746  000001          MOV     #1,-(SP)
        022136  010600                  MOV     SP,R0
        022140  104414                  TRAP    C#PNTB
        022142  062706  000004          ADD     #4,SP
1692    022146                          PUSH    R4
1693    022150  016504  000104          MOV     C.RHDR(R5),R4      ; GET
1694    022154  062704  000062          ADD     #<RE.RP+L.SDI>,R4  ;   SDI
                                                                   ;     STATUS
```

```
1693 022160                     PRINTB  #X53B,10.(R4),8.(R4),6(R4),4(R4),2(R4),(R4);
     022160  011446             MOV     (R4),-(SP)
     022162  016446  000002     MOV     2(R4),-(SP)
     022166  016446  000004     MOV     4(R4),-(SP)
     022172  016446  000006     MOV     6(R4),-(SP)
     022176  016446  000010     MOV     8.(R4),-(SP)
     022202  016446  000012     MOV     10.(P4),-(SP)
     022206  012746  015636      MOV    #X53B,-(SP)
     022212  012746  000007     MOV     #7,-(SP)
     022216  010600             MOV     SP,RO
     022220  104414             TRAP    C#PNTB
     022222  062706  000020     ADD     #20.SP
1696 022226                     POP     R4
1697 022230  004737  022334     CALL    ERLSTA              ; DO COMMON STUFF
1698 022234             ENDMSG
            L10037:
     022234  104423             TRAP    C#MSG

1699
1700 022236             BGNMSG ERRO54
     022236             ERRO54::
1701 022236  004737  054270     CALL    CALR6               ; PRINT HEADER INFO
1702 022242                     PRINTB  #X54A               ; PRINT ERROR LOG RECEIVED
     022242  012746  015731     MOV     #X54A,-(SP)
     022246  012746  000001     MOV     #1,-(SP)
     022252  010600             MOV     SP,RO
     022254  104414             TRAP    C#PNTB
     022256  062706  000004     ADD     #4.SP
1703 022262                     PUSH    R4                  ; CONVERT
1704 022264  016504  000104     MOV     C.RHDR(R5),R4       ;   CYLINDER
1705 022270  062704  000050     ADD     #<RE.RP+L.SCYL>,R4  ;     TO
1706 022274  004737  047552     CALL    BLDSTR              ;       ASCII
1707 022300                     POP     R4                  ;         STRING
1708 022302                     PRINTB  #X54B,#TEMP         ;
     022302  012746  002212     MOV     #TEMP,-(SP)
     022306  012746  016C03     MOV     #X54B,-(SP)
     022312  012746  000002     MOV     #2,-(SP)
     022316  010600             MOV     SP,RO
     022320  104414             TRAP    C#PNTB
     022322  062706  000006     ADD     #6,SP
1709 022326  004737  022334     CALL    ERLSTA              ; DO COMMON STUFF
1710 022332             ENDMSG
            L10040:
     022332  104423             TRAP    C#MSG

1711
1712 022334  032704  000200 ERLSTA: BIT   #LF.SUC,R4        ; SUCCESS?
1713 022340  001014             BNE     2$                  ;
1714 022342  032704  000100 1$:   BIT   #LF.CON,R4          ;
1715 022346  001022             BNE     3$                  ;
1716 022350                     PRINTX  #MSGFAI             ; "OPERATION FAILED"
     022350  012746  016160     MOV     #MSGFAI,-(SP)
     022354  012746  000001     MOV     #1,-(SP)
     022360  010600             MOV     SP,RO
     022362  104415             TRAP    C#PNTX
     022364  062706  000004     ADD     #4,SP
1717 022370  000421             BR      4$                  ;
```

```
1718 022372                    2$:     PRINTX  #MSGSUC             ; "OPERATION SUCCESSFUL"
     022372  012746  016205            MOV     #MSGSUC,-(SP)
     022376  012746  000001            MOV     #1.,-(SP)
     022402  010600                    MOV     SP,R0
     022404  104415                    TRAP    C$PNTX
     022406  062706  000004            ADD     #4,SP
1719 022412  000410                    BR      4$                  ;
1720 022414                    3$:     PRINTX  #MSGCON             ; "OPERATION CONTINUING"
     022414  012746  016236            MOV     #MSGCON,-(SP)
     022420  012746  000001            MOV     #1.,-(SP)
     022424  010600                    MOV     SP,R0
     022426  104415                    TRAP    C$PNTX
     022430  062706  000004            ADD     #4,SP
1721 022434                    4$:     PRINTB  #XSTA,STAMS0,STAMS1  ; PRINT STATUS
     022434  013746  002320            MOV     STAMS1,-(SP)
     022440  013746  002316            MOV     STAMS0,-(SP)
     022444  012746  016423            MOV     #XSTA,-(SP)
     022450  012746  000003            MOV     #3,-(SP)
     022454  010600                    MOV     SP,R0
     022456  104414                    TRAP    C$PNTB
     022460  062706  000010            ADD     #10,SP
1722 022464  004737  022616            CALL    MSGPKT              ; DUMP MESSAGE PACKET
1723 022470  000207                    RETURN                      ;
1724
1725 022472                    BGNMSG ERRO60
     022472                    ERRO60::
1726 022472  004737  054270            CALL    CALR6               ; PRINT HEADER INFO
1727 022476                            PRINTB  #ATNMSG,#AMAV       ;
     022476  012746  007505            MOV     #AMAV,-(SP)
     022502  012746  011147            MOV     #ATNMSG,-(SP)
     022506  012746  000002            MOV     #2.,-(SP)
     022512  010600                    MOV     SP,R0
     022514  104414                    TRAP    C$PNTB
     022516  062706  000006            ADD     #6,SP
1728 022522                            PRINTB  #X60                ; PRINT ERROR LOG RECEIVED
     022522  012746  016024            MOV     #X60,-(SP)
     022526  012746  000001            MOV     #1.,-(SP)
     022532  010600                    MOV     SP,R0
     022534  104414                    TRAP    C$PNTB
     022536  062706  000004            ADD     #4,SP
1729 022542                    ENDMSG
     022542                    L10041:
     022542  104423                    TRAP    C$MSG
1730
1731 022544                    BGNMSG ERRO61
     022544                    ERRO61::
1732 022544  004737  054270            CALL    CALR6               ; PRINT HEADER INFO
1733 022550                            PRINTB  #ATNMSG,#AMDU       ;
     022550  012746  007521            MOV     #AMDU,-(SP)
     022554  012746  011147            MOV     #ATNMSG,-(SP)
     022560  012746  000002            MOV     #2.,-(SP)
     022564  010600                    MOV     SP,R0
     022566  104414                    TRAP    C$PNTB
     022570  062706  000006            ADD     #6,SP
1734 022574                            PRINTB  #X61                ; PRINT ERROR LOG RECEIVED
```

```
          022574  012746  016077            MOV     #X61,-(SP)
          022600  012746  000001            MOV     #1,-(SP)
          022604  010600                    MOV     SP,RO
          022606  104414                    TRAP    C$PNTB
          022610  062706  000004            ADD     #4,SP
     1735 022614                    ENDMSG
          022614                    L10042:
          022614  104423                    TRAP    C$MSG
     1736
     1737
     1738                           ;++
     1739                           ;       MSGPKT - PRINT CONTENTS OF MESSAGE BUFFER FOR ERROR MESSAGES
     1740                           ;--
     1741 022616                    MSGPKT: PUSH    <R2>                    ; SAVE REGISTERS
          022616  010246                    MOV     R2,-(SP)                ; PUSH R2 ON STACK
     1742 022620                            PRINTX  #XMSG1
          022620  012746  016612            MOV     #XMSG1,-(SP)
          022624  012746  000001            MOV     #1,-(SP)
          022630  010600                    MOV     SP,RO
          022632  104415                    TRAP    C$PNTX
          022634  062706  000004            ADD     #4,SP
     1743 022640  016504  000104            MOV     C.RHDR(R5),R4
     1744 022644  016402  000002            MOV     2(R4),R2                ; GET MESSAGE LENGTH IN BYTES
     1745 022650  062704  000006            ADD     #RE.RP,R4               ; POINT TO MESSAGE
     1746 022654  012703  000006    1$:     MOV     #6,R3                   ; GET COUNT OF WORDS / LINE
     1747 022660                    2$:     PRINTX  #XMSG2,(R4)+
          022660  012446                    MOV     (R4)+,-(SP)
          022662  012746  016647            MOV     #XMSG2,-(SP)
          022666  012746  000002            MOV     #2,-(SP)
          022672  010600                    MOV     SP,RO
          022674  104415                    TRAP    C$PNTX
          022676  062706  000006            ADD     #6,SP
     1748 022702  162702  000002            SUB     #2,R2                   ; IF NOT POSITIVE,
     1749 022706  003413                    BLE     5$                      ;    DONE PRINTING.
     1750 022710  005303                    DEC     R3                      ; IF NOT ZERO,
     1751 022712  001362                    BNE     2$                      ;    PRINT NEXT WORD
     1752 022714                            PRINTX  #CRLF                   ; ELSE, TERMINATE LINE
          022714  012746  010300            MOV     #CRLF,-(SP)
          022720  012746  000001            MOV     #1,-(SP)
          022724  010600                    MOV     SP,RO
          022726  104415                    TRAP    C$PNTX
          022730  062706  000004            ADD     #4,SP
     1753 022734  000747                    BR      1$                      ;    PRINT REMAINDER OF PACKET
     1754 022736                    5$:     PRINTX  #CRLF                   ; MAKE SURE LINE IS TERMINATED.
          022736  012746  010300            MOV     #CRLF,-(SP)
          022742  012746  000001            MOV     #1,-(SP)
          022746  010600                    MOV     SP,RO
          022750  104415                    TRAP    C$PNTX
          022752  062706  000004            ADD     #4,SP
     1755 022756                            POP     <R2>                    ; RESTORE SAVED REGISTERS
     1756 022760  000207                    RETURN
     1757
     1759 022762  000000            DBGFLG: .WORD   0                       ; FLAGS FOR DEBUG
     1760                           ;    BIT0 - DUMP CMD PKT
     1761                           ;    BIT1 - DUMP RSP PKT
```

```
1762                                                    ;       BIT2 - CMD SENT MSG
1763                                                    ;       BIT3 - RSP RCVD MSG
1764 022764      045      101      103  DBGSND: .ASCIZ  /%ACOMMAND SENT%N/
1765 023005      045      101      101  DBGATN: .ASCIZ  /%AATTENTION MESSAGE%N/
1766 023033      045      101      111  DBGINV: .ASCIZ  /%AINVALID COMMAND%N/
1767 023057      045      101      062  DBGGCS: .ASCIZ  /%A202 - GET COMMAND STATUS END PACKET%N/
1768 023127      045      101      062  DBGGUS: .ASCIZ  /%A203 - GET UNIT STATUS END PACKET%N/
1769 023174      045      101      062  DBGSCC: .ASCIZ  /%A204 - SET CONTROLLER CHARACTERISTICS%N/
1770 023245      045      101      062  DBGAVL: .ASCIZ  /%A210 - AVAILABLE END PACKET%N/
1771 023304      045      101      062  DBGONL: .ASCIZ  /%A211 - ONLINE END PACKET%N/
1772 023340      045      101      062  DBGSUC: .ASCIZ  /%A212 - SET UNIT CHARACTERISTICS %N/
1773 023404      045      101      062  DBGACC: .ASCIZ  /%A220 - ACCESS END PACKET%N/
1774 023440      045      101      062  DBGERS: .ASCIZ  /%A222 - ERASE END PACKET%N/
1775 023473      045      101      062  DBGMRD: .ASCIZ  /%A230 - MAINTENANCE READ%N/
1776 023526      045      101      062  DBGMWR: .ASCIZ  /%A231 - MAINTENANCE WRITE%N/
1777 023562      045      101      062  DBGCMP: .ASCIZ  /%A240 - COMPARE HOST DATA%N/
1778 023616      045      101      062  DBGRD:  .ASCIZ  /%A241 - READ END PACKET%N/
1779 023650      045      101      062  DBGWR:  .ASCIZ  /%A242 - WRITE END PACKET%N/
1780                                            .EVEN
1781                                    ;++
1782                                    ;       DBGCMD - DUMP COMMAND PACKET FOR DEBUG PURPOSES
1783                                    ;--
1784
1785 023704                            DBGCMD: PUSH    <R3,R4>
1786 023710   032737   000004  022762          BIT     #BIT2,DBGFLG
1787 023716   001502                            BEQ     90$
1788 023720                                     PRINTF  #DBGSND
     023720   012746   022764                   MOV     #DBGSND,-(SP)
     023724   012746   000001                   MOV     #1,-(SP)
     023730   010600                            MOV     SP,R0
     023732   104417                            TRAP    C$PNTF
     023734   062706   000004                   ADD     #4,SP
1789 023740   032737   000001  022762          BIT     #BIT0,DBGFLG
1790 023746   001466                            BEQ     90$
1791 023750                                     PRINTF  #XMSG1
     023750   012746   016612                   MOV     #XMSG1,-(SP)
     023754   012746   000001                   MOV     #1,-(SP)
     023760   010600                            MOV     SP,R0
     023762   104417                            TRAP    C$PNTF
     023764   062706   000004                   ADD     #4,SP
1792 023770                                     PRINTF  #XMSG2,(R4),2(R4),4(R4),14.(R4),16.(R4),18.(R4)
     023770   016446   000022                   MOV     18.(R4),-(SP)
     023774   016446   000020                   MOV     16.(R4),-(SP)
     024000   016446   000016                   MOV     14.(R4),-(SP)
     024004   016446   000004                   MOV     4(R4),-(SP)
     024010   016446   000002                   MOV     2(R4),-(SP)
     024014   011446                            MOV     (R4),-(SP)
     024016   012746   016647                   MOV     #XMSG2,-(SP)
     024022   012746   000007                   MOV     #7,-(SP)
     024026   010600                            MOV     SP,R0
     024030   104417                            TRAP    C$PNTF
     024032   062706   000020                   ADD     #20,SP
1793 024036   062704   000026                   ADD     #ME.CP,R4
1794 024042   012703   000005                   MOV     #5,R3
1795 024046                            1$:      PRINTF  #XMSG2,(R4),2(R4),4(R4),6(R4),8.(R4),10.(R4)
```

```
        024046  016446  000012          MOV     10.(R4),-(SP)
        024052  016446  000010          MOV     8.(R4),-(SP)
        024056  016446  000006          MOV     6(R4),-(SP)
        024062  016446  000004          MOV     4(R4),-(SP)
        024066  016446  000002          MOV     2(R4),-(SP)
        024072  011446                  MOV     (R4),-(SP)
        024074  012746  016647          MOV     #XMSG2,-(SP)
        024100  012746  000007          MOV     #7,-(SP)
        024104  010600                  MOV     SP,R0
        024106  104417                  TRAP    C$PNTF
        024110  062706  000020          ADD     #20,SP
1796    024114  062704  000014          ADD     #12.,R4
1797    024120  005303                  DEC     R3
1798    024122  001351                  BNE     1$
1799    024124              90$:        POP     <R4,R3>
1800    024130  000207                  RETURN
1801
1802                        ;++
1803                        ;           DBGRSP - DUMP RESPONSE PACKET FOR DEBUG PURPOSES
1804                        ;--
1805
1806    024132  032737  000010  022762  DBGRSP: BIT     #BIT3,DBGFLG
1807    024140  001462                  BEQ     90$
1808    024142                          PRINTF  R0
        024142  010046                  MOV     R0,-(SP)
        024144  012746  000001          MOV     #1,-(SP)
        024150  010600                  MOV     SP,R0
        024152  104417                  TRAP    C$PNTF
        024154  062706  000004          ADD     #4,SP
1809    024160  032737  000002  022762  BIT     #BIT1,DBGFLG
1810    024166  001447                  BEQ     90$
1811    024170                          PUSH    <R3,R4>
1812    024174                          PRINTF  #XMSG1
        024174  012746  016612          MOV     #XMSG1,-(SP)
        024200  012746  000001          MOV     #1,-(SP)
        024204  010600                  MOV     SP,R0
        024206  104417                  TRAP    C$PNTF
        024210  062706  000004          ADD     #4,SP
1813    024214  016504  000104          MOV     C.RHDR(R5),R4
1814    024220  012703  000005          MOV     #5,R3
1815    024224              1$:         PRINTF  #XMSG2,(R4),2(R4),4(R4),6(R4),8.(R4),10.(R4)
        024224  016446  000012          MOV     10.(R4),-(SP)
        024230  016446  000010          MOV     8.(R4),-(SP)
        024234  016446  000006          MOV     6(R4),-(SP)
        024240  016446  000004          MOV     4(R4),-(SP)
        024244  016446  000002          MOV     2(R4),-(SP)
        024250  011446                  MOV     (R4),-(SP)
        024252  012746  016647          MOV     #XMSG2,-(SP)
        024256  012746  000007          MOV     #7,-(SP)
        024262  010600                  MOV     SP,R0
        024264  104417                  TRAP    C$PNTF
        024266  062706  000020          ADD     #20,SP
1816    024272  062704  000014          ADD     #12.,R4
1817    024276  005303                  DEC     R3
1818    024300  001351                  BNE     1$
```

```
1819 024302                        POP     <R4,R3>
1820 024306  000207          90$:  RETURN
```

```
1823                                    .SBTTL  GLOBAL SUBROUTINES SECTION
1824
1825                                    ;++
1826                                    ;       BADMEM - PROGRAM DATA STRUCTURES HAVE BEEN CORRUPTED.
1827                                    ;
1828                                    ;       THIS ROUTINE PRINTS A SYSTEM FATAL ERROR AND EXITS THE TEST
1829                                    ;--
1830
1831 024310                     BADMEM: ERRSF 6,,ERR006                      ; SYSTEM FATAL MEMORY CORRUPTION ERROR
     024310  104454                     TRAP    C$ERSF
     024312  000006                     .WORD   6
     024314  000000                     .WORD   0
     024316  017332                     .WORD   ERR006
1832 024320                             DOCLN                               ; DO CLEAN-UP TRAP
     024320  104444                     TRAP    C$DCLN
```

```
1834
1835                                        ;++
1836                                        ;          ALOCM - ALLOCATE A BLOCK OF FREE MEMORY.  REPORT ERROR IF MEMORY
1837                                        ;          EXHAUSTED.
1838
1839                                        ;          INPUTS:
1840                                        ;                  R1 - NUMBER OF WORDS TO ALLOCATE
1841                                        ;                  FFREE - FIRST FREE WORD IN MEMORY
1842                                        ;                  FSIZE - SIZE OF FREE MEMORY AVAILABLE IN WORDS
1843                                        ;          OUTPUTS:
1844                                        ;                  R1 - ADDRESS OF FIRST WORD OF ALLOCATED MEMORY
1845                                        ;                  FFREE - NEW FIRST FREE WORD IN MEMORY
1846                                        ;                  FSIZE - SIZE OF FREE MEMORY LEFT AFTER ALLOCATION
1847
1848                                        ;          SYSTEM FATAL ERROR WILL BE REPORTED IF NOT ENOUGH MEMORY AVAILABLE
1849                                        ;          AND ENTIRE PROGRAM WILL BE STOPPED.
1850                                        ;--
1851
1852 024322                                ALOCM:  PUSH    <FFREE>                 ; SAVE FFREE AT ENTRY
     024322   013746   002166                     MOV     FFREE,-(SP)             ; PUSH FFREE ON STACK
1853 024326   160137   002170                     SUB     R1,FSIZE                ; REDUCE SIZE OF FREE MEMORY
1854 024332   002002                              BGE     1$                      ;
1855 024334   000137   062574                      JMP     NOMEM                  ; REPORT ERROR IF NOT ENOUGH MEMORY
1856 024340   060101                        1$:   ADD     R1,R1                   ; CHANGE WORDS TO BYTES
1857 024342   060137   002166                     ADD     R1,FFREE                ; CALCULATE NEW START OF FREE MEMORY
1858 024346                                       POP     <R1>                    ; GET START OF ALLOCATED MEMORY
     024346   012601                              MOV     (SP)+,R1                ; POP STACK INTO R1
1859 024350   000240                        NOP                                   ; [DBG] INSERT HALT HERE FOR DEBUG
1860 024352   000207                              RETURN
1861
```

```
1863
1864                                    ;++
1865                                    ;         GETBUF - GET A BUFFER FROM THE FREE BUFFER POOL
1866                                    ;
1867                                    ;         INPUTS:
1868                                    ;                 R1 - REQUESTED PACKET SIZE (BLOCKS)
1869                                    ;         OUTPUTS:
1870                                    ;                 R1 - ALLOCATED PACKET SIZE (BLOCKS)
1871                                    ;                 R3 - BYTE OFFSET INTO MAPPED MEMORY
1872                                    ;                 BCLO,BCHI - BYTE COUNT OF BUFFER
1873                                    ;                 BUFBA,BUFEA - STARTING ADDRESS OF BUFFER
1874                                    ;--
1875
1876 024354                 GETBUF: PUSH  <R0,R2,R4,R5>        ; SAVE REGISTER CONTENTS
     024354  010046                  MOV   R0,-(SP)            ; PUSH R0 ON STACK
     024356  010246                  MOV   R2,-(SP)            ; PUSH R2 ON STACK
     024360  010446                  MOV   R4,-(SP)            ; PUSH R4 ON STACK
     024362  010546                  MOV   R5,-(SP)            ; PUSH R5 ON STACK
1877 024364  004737  024570          CALL  BESFIT              ; FIND A BUFFER
1878 024370  005705                  TST   R5                  ; IF ZERO, NO BUFFER
1879 024372  001463                  BEQ   80$                 ;      AVAILABLE
1880 024374  020501                  CMP   R5,R1               ; COMPARE GOT W/ NEEDED
1881 024376  001452                  BEQ   40$                 ; IF EQUAL, USE THE BUFFER
1882 024400  003006                  BGT   11$                 ; IF GREATER, ONLY GET WHAT'S NEEDED
1883 024402  004737  024702          CALL  BUFMRG              ; MERGE ADJACENT FREE BLOCKS
1884 024406  004737  024570          CALL  BESFIT              ; FIND A BUFFER
1885 024412  020501          10$:    CMP   R5,R1               ; COMPARE GOT W/ NEEDED
1886 024414  101443                  BLOS  40$                 ; IF LE, USE ENTIRE BUFFER
1887 024416  013703  004624  11$:    MOV   F.HDR,R3            ; GET FREE PACKET
1888 024422  001004                  BNE   12$                 ; IF NOT ZERO, CONTINUE
1889 024424  012701  024422          MOV   #.-2,R1             ; SAVE CURRENT LOCATION
1890 024430  000137  024310          JMP   BADMEM              ; PRINT SYSTEM FATAL ERROR
1891 024434  011337  004624  12$:    MOV   (R3),F.HDR          ; ADJUST
1892 024440  011413                  MOV   (R4),(R3)           ;   ALL OF THE
1893 024442  010314                  MOV   R3,(R4)             ;     FORWARD LINKS
1894 024444  160105                  SUB   R1,R5               ; SUBTRACT SIZE NEEDED
1895 024446  042764  000377  000002  BIC   #377,2(R4)          ; SAVE REMAINING
1896 024454  050564  000002          BIS   R5,2(R4)            ;     BUFFER SIZE
1897 024460                          PUSH  <R4,R3,R1>          ; SAVE REGISTERS
     024460  010446                  MOV   R4,-(SP)            ; PUSH R4 ON STACK
     024462  010346                  MOV   R3,-(SP)            ; PUSH R3 ON STACK
     024464  010146                  MOV   R1,-(SP)            ; PUSH R1 ON STACK
1898 024466  010501                  MOV   R5,R1               ; CONVERT BLOCKS
1899 024470  004737  045502          CALL  GETBC               ;   TO BYTES
1900 024474                          POP   <R1,R3,R4>          ; RESTORE REGISTERS
     024474  012601                  MOV   (SP)+,R1            ; POP STACK INTO R1
     024476  012603                  MOV   (SP)+,R3            ; POP STACK INTO R3
     024500  012604                  MOV   (SP)+,R4            ; POP STACK INTO R4
1901 024502  066437  000004  004406  ADD   4(R4),BCLO          ; CALCULATE NEW OFFSET
1902 024510  013763  004406  000004  MOV   BCLO,4(R3)          ; SAVE OFFSET TO NEW BUFFER
1903 024516  010163  000002          MOV   R1,2(R3)            ; SAVE NEW BUFFER SIZE
1904 024522  000402                  BR    50$
1905 024524  010403          40$:    MOV   R4,R3               ; POINT TO NEEDED PACKET
1906 024526  010501                  MOV   R5,R1               ; GET ALLOCATED PACKET SIZE
1907 024530  105263  000003  50$:    INCB  3(R3)               ; INCREMENT # OF USERS COUNTER
```

```
1908 024534 016303 000004              MOV     4(R3),R3                ; GET OFFSET
1909 024540 000403                     BR      90$                     ;
1910
1911 024542 012703 177777      80$:    MOV     #-1,R3                  ; NO MEMORY LEFT, GENERATE OFFSET
1912 024546 005001                     CLR     R1                      ; SET SIZE TO ZERO
1913
1914 024550 004737 025006      90$:    CALL    MAPMEM                  ; GET PHYSICAL ADDRESS & BYTE COUNT
1915 024554 000240             NOP                             ; [DBG] INSERT HALT FOR DEBUG
1916 024556                            POP     <R5,R4,R2,R0>           ; RESTORE SAVED REGISTERS
     024556 012605                     MOV     (SP)+,R5                ; POP STACK INTO R5
     024560 012604                     MOV     (SP)+,R4                ; POP STACK INTO R4
     024562 012602                     MOV     (SP)+,R2                ; POP STACK INTO R2
     024564 012600                     MOV     (SP)+,R0                ; POP STACK INTO R0
1917 024566 000207                     RETURN                          ;
1918
1919
1920                           ;++
1921                           ;       BESFIT - FIND A MEMORY PACKET USING A BEST FIT ALGORITHM
1922                           ;
1923                           ;       INPUTS:
1924                           ;               NONE
1925                           ;       OUTPUTS:
1926                           ;               R4 - POINTER TO BEST PACKET
1927                           ;               R5 - SIZE OF BEST PACKET
1928                           ;++
1929 024570 005005             BESFIT: CLR     R5                      ; INITIALIZE TEMPORARY
1930 024572 005004                     CLR     R4                      ;       POINTERS
1931 024574 013703 004622              MOV     M.HDR,R3                ; GET POINTER TO AVAILABLE LIST
1932 024600 001004                     BNE     10$                     ; IF NOT END OF LIST, CONTINUE
1933 024602 012701 024600              MOV     #.-2,R1                 ; SAVE CURRENT LOCATION
1934 024606 000137 024310              JMP     BADMEM                  ; PRINT SYSTEM FATAL ERROR
1935 024612 105763 000003      10$:    TSTB    3(R3)                   ; IF NUMBER OF USERS IS NOT 0
1936 024616 001025                     BNE     30$                     ;       GET NEXT PACKET
1937 024620 116300 000002              MOVB    2(R3),R0                ; GET BUFFER SIZE
1938 024624 001004                     BNE     20$                     ; IF NOT ZERO, CONTINUE
1939 024626 012701 024624              MOV     #.-2,R1                 ; SAVE CURRENT LOCATION
1940 024632 000137 024310              JMP     BADMEM                  ; PRINT SYSTEM FATAL ERROR
1941
1942 024636 120005             20$:    CMPB    R0,R5                   ; COMPARE CURRENT SIZE W/ GOT
1943 024640 001414                     BEQ     30$                     ; IF SAME, GET NEXT BUFFER
1944 024642 103403                     BLO     23$                     ; IF LT, COMPARE W/ NEEDED
1945 024644 120105                     CMPB    R1,R5                   ; COMPARE NEEDED W/ GOT
1946 024646 101411                     BLOS    30$                     ; IF LE, GET NEXT BUFFER
1947 024650 000402                     BR      25$                     ;       ELSE THIS IS BEST FIT
1948 024652 120001             23$:    CMPB    R0,R1                   ; COMPARE CURRENT W/ NEEDED
1949 024654 103406                     BLO     30$                     ; IF LE, GET NEXT BUFFER
1950 024656 010304             25$:    MOV     R3,R4                   ;       ELSE, USE CURRENT BUFFER
1951 024660 010005                     MOV     R0,R5                   ; SAVE SIZE OF CURRENT BUFFER
1952 024662 042705 177400              BIC     #^C<377>,R5             ; CLEAR EXTRANEOUS BITS
1953 024666 020501                     CMP     R5,R1                   ; IF GOT = NEED
1954 024670 001402                     BEQ     31$                     ;       EXIT
1955 024672 011303             30$:    MOV     (R3),R3                 ; POINT TO NEXT PACKET
1956 024674 001346                     BNE     10$                     ; IF NZ, LOOK AT NEXT PACKET
1957 024676                     31$:
1958 024676 000240             NOP                             ; [DBG] INSERT HALT FOR DEBUG
```

1959 024700 000207   RETURN   ;

```
1961
1962
1963                                  ;++
1964                                  ;        BUFMRG - MERGE ADJACENT FREE BLOCKS IN FREE MEMORY LIST
1965                                  ;
1966                                  ;        INPUTS:
1967                                  ;                NONE
1968                                  ;        OUTPUTS:
1969                                  ;                NONE
1970                                  ;--
1971 024702 013703 004622    BUFMRG: MOV     M.HDR,R3                    ; GET POINTER TO AVAILABLE LIST
1972 024706 001433           10$:    BEQ     41$                        ; IF END OF LIST, MEMORY ERROR
1973 024710 011304                   MOV     (R3),R4                    ; GET POINTER TO NEXT PACKET
1974 024712 001427                   BEQ     40$                        ; IF END OF LIST, EXIT
1975 024714 105763 000003            TSTB    3(R3)                      ; CHECK IF INUSE
1976 024720 001022                   BNE     30$                        ; IF SO, GET NEXT PACKET
1977 024722 105764 000003            TSTB    3(R4)                      ; CHECK IF INUSE
1978 024726 001017                   BNE     30$                        ; IF SO, GET NEXT PACKET
1979 024730 116300 000002            MOVB    2(R3),R0                   ; GET BUFFER SIZE OF FIRST PACKET
1980 024734 001420                   BEQ     41$                        ; IF ZERO, SYSTEM FATAL ERROR
1981 024736 116405 000002            MOVB    2(R4),R5                   ; GET SIZE OF ADJACENT BUFFER
1982 024742 001415                   BEQ     41$                        ; IF ZERO, SYSTEM FATAL ERROR
1983 024744 060005                   ADD     R0,R5                      ; CALCULATE TOTAL SIZE
1984 024746 010563 000002            MOV     R5,2(R3)                   ; INSERT NEW SIZE
1985 024752 011413                   MOV     (R4),(R3)                  ; ADJUST POINTERS
1986 024754 013714 004624            MOV     F.HDR,(R4)                 ; PUT UNUSED PACKET POINTER
1987 024760 010437 004624            MOV     R4,F.HDR                   ;    INTO TOP OF FREE LIST
1988 024764 000750                   BR      10$                        ; LOOP
1989
1990 024766 010403           30$:    MOV     R4,R3                      ; POINT TO NEXT PACKET
1991 024770 000746                   BR      10$                        ; LOOP
1992
1993 024772                  40$:
1994 024772 000240           NOP                         ; [DBG] INSERT HALT FOR DEBUG
1995 024774 000207                   RETURN                             ; DONE MERGING PACKETS
1996
1997 024776 012701 024774    41$:    MOV     #.-2,R1                    ; SAVE CURRENT LOCATION
1998 025002 000137 024310            JMP     BADMEM                     ; PRINT SYSTEM FATAL ERROR
```

```
2000
2001                              ;++
2002                              ;        MAPMEM - MAP MEMORY OFFSET TO PHYSICAL MEMORY AND CALCULATE
2003                              ;                 BYTE COUNT OF SEGMENT
2004                              ;
2005                              ;        INPUTS:
2006                              ;                R1 - BLOCK COUNT OF SEGMENT
2007                              ;                R3 - MEMORY OFFSET (BYTES)
2008                              ;        OUTPUTS:
2009                              ;                REGISTERS UNCHANGED
2010                              ;                BCLO,BCHI - BYTE COUNT OF SEGMENT
2011                              ;                BUFBA,BUFEA - PHYSICAL ADDRESS OF SEGMENT
2012                              ;                TSTOFF - MAPPED MEMORY OFFSET TO BUFFER, OR -1
2013                              ;--
2014
2015 025006 005701       MAPMEM: TST     R1                      ; IF NOT ZERO,
2016 025010 001014               BNE     1$                      ;   CONTINUE
2017 025012 005037 004406        CLR     BCLO                    ; ELSE,
2018 025016 005037 004410        CLR     BCHI                    ;   CLEAR
2019 025022 005037 004370        CLR     BUFBA                   ;     EVERYTHING
2020 025026 005037 004372        CLR     BUFEA                   ;
2021 025032 012737 177777 002242 MOV     #-1,TSTOFF             ; FLAG AS NO BUFFER USED
2022 025040 000207               RETURN
2023
2024 025042              1$:     PUSH    <R2,R3,R4>              ; SAVE REGISTERS
     025042 010246               MOV     R2,-(SP)                ; PUSH R2 ON STACK
     025044 010346               MOV     R3,-(SP)                ; PUSH R3 ON STACK
     025046 010446               MOV     R4,-(SP)                ; PUSH R4 ON STACK
2025 025050 010337 002242        MOV     R3,TSTOFF              ; SAVE MEMORY OFFSET
2026 025054 004737 045502        CALL    GETBC                   ; GET BYTE COUNT
2027 025060 032737 100000 002204 BIT     #IMMU,IFLAGS           ; CHECK FOR MEMORY MAPPING
2028 025066 001411               BEQ     10$                     ; IF CLEAR, USE MEMORY MAPPING
2029 025070 013702 002166        MOV     FFREE,R2                ; ELSE, USE
2030 025074 063702 002242        ADD     TSTOFF,R2              ;   FREE MEMORY
2031 025100 010237 004370        MOV     R2,BUFBA               ;
     025104 005037 004372        CLR     BUFEA                   ;
     025110 000414               BR      20$                     ;
2034 025112              10$:    MAP     TSTOFF,BUFBA,BUFEA     ; USE MAPPED MEMORY
     025112 000403               BR      100000$
     025114 002242              .WORD   TSTOFF
     025116 004370              .WORD   BUFBA
     025120 004372              .WORD   BUFEA
     025122 012700 025114 100000$: MOV   #.-6,R0
     025126 104502               TRAP    C:MAP
2035 025130                      BRERROR 20$                     ; IF NOT COMPLETE,
     025130 103004               BCC     20$
2036 025132 012701 025130        MOV     #.-2,R1                ;   DATA STRUCTURES
2037 025136 000137 024310        JMP     BADMEM                  ;     ARE CORRUPT
2038 025142              20$:    POP     <R4,R3,R2>              ; RESTORE REGISTERS
     025142 012604               MOV     (SP)+,R4                ; POP STACK INTO R4
     025144 012603               MOV     (SP)+,R3                ; POP STACK INTO R3
     025146 012602               MOV     (SP)+,R2                ; POP STACK INTO R2
2039 025150 000240               NOP                             ; [DBG] INSERT HALT FOR DEBUG
2040 025152 000207               RETURN                          ;
```

```
2042
2043
2044                                    ;++
2045                                    ;          MMTOBF - COPY A BLOCK OF MAPPED MEMORY TO A BUFFER IN LOW MEMORY.
2046                                    ;
2047                                    ;          INPUTS:
2048                                    ;                  R1 - ADDRESS OF LOW MEMORY BUFFER
2049                                    ;                  R3 - MEMORY OFFSET (BYTES)
2050                                    ;          OUTPUTS:
2051                                    ;                  LOW MEMORY BUFFER CONTAINS 1 BLOCK OF DATA
2052                                    ;                  REGISTERS UNCHANGED
2053                                    ;--
2054 025154                            MMTOBF: PUSH    <R2,R3,R4>              ; SAVE REGISTERS
     025154   010246                           MOV     R2,-(SP)               ; PUSH R2 ON STACK
     025156   010346                           MOV     R3,-(SP)               ; PUSH R3 ON STACK
     025160   010446                           MOV     R4,-(SP)               ; PUSH R4 ON STACK
2055 025162   032737  100000  002204           BIT     #IMMU,IFLAGS           ; CHECK FOR MEMORY MAPPING
2056 025170   001412                           BEQ     10$                    ; IF CLEAR, USE MEMORY MAPPING
2057 025172   013702  002166                   MOV     FFREE,R2               ; ELSE, USE
2058 025176   060302                           ADD     R3,R2                  ;      FREE MEMORY
2059 025200   010104                           MOV     R1,R4                  ;
2060 025202   012700  000400                   MOV     #256.,R0               ;
2061 025206   012224                   1$:     MOV     (R2)+,(R4)+            ;
2062 025210   005300                           DEC     R0                     ;
2063 025212   001375                           BNE     1$                     ;
2064 025214   000423                           BR      20$                    ;
2065 025216   010337  004404           10$:    MOV     R3,MMOFF               ; COPY LOW
2066 025222   010137  004400                   MOV     R1,MMBUF               ;      MEMORY TO
2067 025226   012737  001000  004402           MOV     #512.,MMSIZ            ;          MAPPED
2068 025234                                    BFTOME  MMOFF,MMBUF,MMSIZ      ;              MEMORY
     025234   000403                            BR      10001$
     025236   004404                            .WORD   MMOFF
     025240   004400                            .WORD   MMBUF
     025242   004402                            .WORD   MMSIZ
     025244                            10001$:
     025244   012700  025236                    MOV     #.-6,R0
     025250   104476                            TRAP    C$TOME
2069 025252                                    BCOMPLETE 20$                  ; IF NOT COMPLETE,
     025252   103404                            BCS     20$
2070 025254   012701  025252                   MOV     #.-2,R1                ;      DATA STRUCTURES
2071 025260   000137  024310                   JMP     BADMEM                 ;          ARE CORRUPT
2072 025264                            20$:
2073 025264   000240                   NOP                          ; [DBG] INSERT HALT FOR DEBUG
2074 025266                                    POP     <R4,R3,R2>             ; RESTORE REGISTERS
     025266   012604                            MOV     (SP)+,R4               ; POP STACK INTO R4
     025270   012603                            MOV     (SP)+,R3               ; POP STACK INTO R3
     025272   012602                            MOV     (SP)+,R2               ; POP STACK INTO R2
2075 025274   000207                            RETURN                         ;
```

```
2077
2078
2079                                        ;++
2080                                        ;        BFTOMM - COPY THE CONTENTS OF A BUFFER IN LOW MEMORY TO MAPPED
2081                                        ;        MEMORY.
2082                                        ;
2083                                        ;        INPUTS:
2084                                        ;            R1 - ADDRESS OF LOW MEMORY BUFFER
2085                                        ;            R3 - MEMORY OFFSET (BYTES)
2086                                        ;        OUTPUTS:
2087                                        ;            LOW MEMORY BUFFER COPIED TO MAPPED MEMORY
2088                                        ;            REGISTERS UNCHANGED
2089                                        ;--
2090 025276                        BFTOMM:  PUSH     <R2,R3,R4>                  ; SAVE REGISTERS
     025276  010246                         MOV      R2,-(SP)                    ; PUSH R2 ON STACK
     025300  010346                         MOV      R3,-(SP)                    ; PUSH R3 ON STACK
     025302  010446                         MOV      R4,-(SP)                    ; PUSH R4 ON STACK
2091 025304  032737  100000  002204         BIT      #IMMU,IFLAGS                ; CHECK FOR MEMORY MAPPING
2092 025312  001412                         BEQ      10$                         ; IF CLEAR, USE MEMORY MAPPING
2093 025314  013702  002166                 MOV      FFREE,R2                    ; ELSE, USE
2094 025320  060302                         ADD      R3,R2                       ;     FREE MEMORY
2095 025322  010104                         MOV      R1,R4                       ;
2096 025324  012700  000400                 MOV      #256.,R0                    ;
2097 025330  012422                 1$:     MOV      (R4)+,(R2)+                 ;
2098 025332  005300                         DEC      R0                          ;
2099 025334  001375                         BNE      1$                          ;
2100 025336  000423                         BR       20$                         ;
2101 025340  010337  004404         10$:    MOV      R3,MMOFF                    ; COPY LOW
2102 025344  010137  004400                 MOV      R1,MMBUF                    ;  MEMORY TO
2103 025350  012737  001000  004402         MOV      #512.,MMSIZ                 ;    MAPPED
2104 025356                                 BFFRME   MMOFF,MMBUF,MMSIZ           ;      MEMORY
     025356  000403                         BR       10002$
     025360  004404                         .WORD    MMOFF
     025362  004400                         .WORD    MMBUF
     025364  004402                         .WORD    MMSIZ
     025366                         10002$:
     025366  012700  025360                 MOV      #.-6,R0
     025372  104500                         TRAP     C$FRME
2105 025374                                 BCOMPLETE 20$                        ; IF NOT COMPLETE,
     025374  103404                         BCS      20$
2106 025376  012701  025374                 MOV      #.-2,R1                     ;   DATA STRUCTURES
2107 025402  000137  024310                 JMP      BADMEM                      ;     ARE CORRUPT
2108 025406                         20$:
2109 025406  000240                 NOP                                         ; [DBG] INSERT HALT FOR DEBUG
2110 025410                                 POP      <R4,R3,R2>                  ; RESTORE REGISTERS
     025410  012604                         MOV      (SP)+,R4                    ; POP STACK INTO R4
     025412  012603                         MOV      (SP)+,R3                    ; POP STACK INTO R3
     025414  012602                         MOV      (SP)+,R2                    ; POP STACK INTO R2
2111 025416  000207                         RETURN                              ;
```

```
2113
2114
2115                                      ;++          RETBUF - RETURN A BUFFER TO THE FREE BUFFER POOL
2116                                      ;
2117                                      ;            INPUTS:
2118                                      ;                    TSTOFF - BYTE OFFSET INTO MAPPED MEMORY OF BUFFER
2119                                      ;            OUTPUTS:
2120                                      ;                    TSTOFF - -1 TO INDICATE NO BUFFER IN USE
2121                                      ;
2122                                      ;--
2123  025420                             RETBUF: PUSH    <R0,R4>                 ; SAVE ALL REGISTERS
      025420  010046                             MOV     R0,-(SP)                ; PUSH R0 ON STACK
      025422  010446                             MOV     R4,-(SP)                ; PUSH R4 ON STACK
2124  025424  013704  004622                     MOV     M.HDR,R4                ; GET POINTER TO FIRST PACKET
2125  025430  001407                     1$:     BEQ     3$                      ; IF END OF LIST, MEMORY ERROR
2126  025432  026437  000004  002242             CMP     4(R4),TSTOFF            ; CHECK CURRENT OFFSET
2127  025440  101003                             BHI     3$                      ; IF GT, FATAL MEMORY ERROR
2128  025442  001406                             BEQ     10$                     ; IF SAME, EXIT LOOP
2129  025444  011404                             MOV     (R4),R4                 ; GET NEXT PACKET
2130  025446  000770                             BR      1$                      ; LOOP
2131  025450  012701  025446             3$:     MOV     #.-2,R1                 ; SAVE CURRENT LOCATION
2132  025454  000137  024310                     JMP     BADMEM                  ; PRINT SYSTEM FATAL ERROR
2133  025460  105764  000003             10$:    TSTB    3(R4)                   ; IF ZERO,
2134  025464  001771                             BEQ     3$                      ;    PROGRAM ERROR
2135  025466  105364  000003                     DECB    3(R4)                   ; DECREMENT USE COUNT
2136  025472  001017                             BNE     30$                     ; IF USE COUNT NOT ZERO, EXIT
2137  025474  012700  000102                     MOV     #66.,R0                 ; ELSE,
2138  025500  026460  000004  004412     20$:    CMP     4(R4),MUTOFF(R0)        ;    REMOVE
2139  025506  001006                             BNE     22$                     ;       BUFFER
2140  025510  012760  177777  004412     21$:    MOV     #-1,MUTOFF(R0)          ;          FROM
2141  025516  005060  004516                     CLR     MUTSIZ(R0)              ;            MULTI-USE
2142  025522  000403                             BR      30$                     ;              LIST
2143  025524  005300                     22$:    DEC     R0                      ;
2144  025526  005300                             DEC     R0                      ;
2145  025530  002363                             BGE     20$                     ;
2146
2147  025532  012737  177777  002242     30$:    MOV     #-1,TSTOFF              ; FLAG AS NO BUFFER IN USE
2148  025540                                     POP     <R4,R0>                 ; RESTORE SAVED REGISTERS
      025540  012604                             MOV     (SP)+,R4                ; POP STACK INTO R4
      025542  012600                             MOV     (SP)+,R0                ; POP STACK INTO R0
2149  025544  000240                             NOP                            ; [DBG] INSERT HALT FOR DEBUG
2150  025546  000207                             RETURN                         ;
```

```
2152
2153
2154                                         ;++
2155                                         ;         GETME - GET POINTER TO THE REQUESTED MESSAGE ENVELOPE
2156                                         ;
2157                                         ;         INPUTS:
2158                                         ;             R1 - DESIRED COMMAND REFERENCE NUMBER
2159                                         ;             R5 - ADDRESS OF CONTROLLER TABLE
2160                                         ;         OUTPUTS:
2161                                         ;             R4 - POINTER TO MESSAGE ENVELOPE
2162                                         ;--
2163  025550  016504  000162         GETME:  MOV    C.CMBL(R5),R4           ; GET TOP OF COMMAND LIST
2164  025554  001406                 1$:     BEQ    3$                     ; IF ZERO, ME NOT FOUND
2165  025556  026401  000026                 CMP    P.CRF+ME.CP(R4),R1     ; IF CMD REF # IS SAME,
2166  025562  001407                         BEQ    4$                     ;   EXIT
2167  025564  016404  000002                 MOV    ME.CBL(R4),R4          ; ELSE,
2168  025570  000771                         BR     1$                     ;   GET NEXT PACKET
2169  025572  012701  025570         3$:     MOV    #.-2,R1                ; SAVE CURRENT LOCATION
2170  025576  000137  024310                 JMP    BADMEM                 ; PRINT SYSTEM FATAL ERROR
2171  025602                         4$:
2172  025602  000240                 NOP                    ; [DBG] INSERT HALT FOR DEBUG
2173  025604  000207                         RETURN                        ;
2174
2175                                         ;++
2176                                         ;         RTNALL - RETURN ALL MESSAGE ENVELOPES TO FREE LIST.
2177                                         ;
2178                                         ;         INPUTS:
2179                                         ;             R5 - ADDRESS OF CONTROLLER TABLE
2180                                         ;         OUTPUTS:
2181                                         ;             T.HDR - POINTS TO RETURNED ENVELOPE
2182                                         ;--
2183
2184  025606  016504  000160         RTNALL: MOV    C.CMFL(R5),R4          ; GET POINTER TO FIRST M.E.
2185  025612  001434                         BEQ    10$                    ; IF LIST EMPTY, EXIT
2186  025614  022764  177777  000020         CMP    #-1,ME.OFF(R4)         ; IF MEMORY OFFSET
2187  025622  001405                         BEQ    1$                     ;   IS NOT -1,
2188  025624  016437  000020  002242         MOV    ME.OFF(R4),TSTOFF      ;     FREE MEMORY
2189  025632  004737  025420                 CALL   RETBUF                 ;       BUFFER
2190  025636  010502                 1$:     MOV    R5,R2                  ;     GET HEADER
2191  025640  062702  000160                 ADD    #C.CMFL,R2             ;
2192  025644  016412  000000                 MOV    ME.CFL(R4),(R2)        ; ADJUST FORWARD LINK OF LAST PACKET
2193  025650  001003                         BNE    2$                     ;
2194  025652  010502                         MOV    R5,R2                  ; IF ZERO, END OF LIST
2195  025654  062702  000162                 ADD    #C.CMBL,R2             ;   R2 POINTS TO LIST HEADER
2196  025660  016412  000002         2$:     MOV    ME.CBL(R4),(R2)        ; ADJUST BACK LINK OF NEXT PACKET
2197
2198  025664  013764  006242  000000         MOV    T.HDR,ME.CFL(R4)       ; ADD ENVELOPE TO
2199  025672  005064  000002                 CLR    ME.CBL(R4)             ;   TOP OF FREE LIST
2200  025676  010437  006242                 MOV    R4,T.HDR               ;
2201  025702  000741                         BR     RTNALL                 ;
2202
2203  025704  016504  000164         10$:    MOV    C.TBSF(R5),R4          ; GET POINTER TO FIRST M.E.
2204  025710  001434                         BEQ    20$                    ; IF LIST EMPTY, EXIT
2205  025712  022764  177777  000020         CMP    #-1,ME.OFF(R4)         ; IF MEMORY OFFSET
2206  025720  001405                         BEQ    11$                    ;   IS NOT -1,
```

```
2207 025722  016437  000020  002242          MOV     ME.OFF(R4),TSTOFF       ;      FREE MEMORY
2208 025730  004737  025420                   CALL    RETBUF                  ;        BUFFER
2209 025734  010502                    11$:   MOV     R5,R2                   ;      GET HEADER
2210 025736  062702  000164                   ADD     #C.TBSF,R2              ;
2211 025742  016412  000000                   MOV     ME.CFL(R4),(R2)         ; ADJUST FORWARD LINK OF LAST PACKET
2212 025746  001003                           BNE     12$
2213 025750  010502                           MOV     R5,R2                   ; IF ZERO, END OF LIST
2214 025752  062702  000166                   ADD     #C.TBSB,R2              ;   R2 POINTS TO LIST HEADER
2215 025756  016412  000002             12$:  MOV     ME.CBL(R4),(R2)         ; ADJUST BACK LINK OF NEXT PACKET
2216
2217 025762  013764  006242  000000           MOV     T.HDR,ME.CFL(R4)        ; ADD ENVELOPE TO
2218 025770  005064  000002                   CLR     ME.CBL(R4)              ;    TOP OF FREE LIST
2219 025774  010437  006242                   MOV     R4,T.HDR                ;
2220 026000  000741                            BR      10$                     ;
2221
2222 026002                             20$:
2223 026002  000240                     NOP                                   ; [DBG] INSERT HALT FOR DEBUG
2224 026004  000207                            RETURN                          ;
2225
2226
2227                                     ;++
2228                                     ;       RTNME - RETURN MESSAGE ENVELOPE TO FREE LIST.
2229                                     ;
2230                                     ;       INPUTS:
2231                                     ;          R5 - ADDRESS OF CONTROLLER TABLE
2232                                     ;          C.RHDR(R5) - POINTER TO RESPONSE PACKET
2233                                     ;       OUTPUTS:
2234                                     ;          T.HDR - POINTS TO RETURNED ENVELOPE
2235                                     ;--
2236 026006  016501  000104             RTNME: MOV     C.RHDR(R5),R1           ; GET
2237 026012  016101  000006                    MOV     P.CRF+RE.RP(R1),R1      ;   POINTER TO
2238 026016  004737  025550                    CALL    GETME                   ;     MESSAGE ENVELOPE
2239 026022  022764  177777  000020            CMP     #-1,ME.OFF(R4)          ; IF MEMORY OFFSET
2240 026030  001405                            BEQ     2$                      ;   IS NOT -1,
2241 026032  016437  000020  002242            MOV     ME.OFF(R4),TSTOFF       ;     FREE MEMORY
2242 026040  004737  025420                    CALL    RETBUF                  ;       BUFFER
2243 026044  016402  000002              2$:   MOV     ME.CBL(R4),R2           ; GET PREVIOUS PACKET
2244 026050  001003                            BNE     3$                      ; IF TOP OF LIST,
2245 026052  010502                            MOV     R5,R2                   ;   GET HEADER
2246 026054  062702  000160                    ADD     #C.CMFL,R2              ;
2247 026060  011412                      3$:   MOV     (R4),(R2)               ; ADJUST FORWARD LINK OF LAST PACKET
2248 026062  001004                            BNE     5$                      ;
2249 026064  016465  000002  000162            MOV     ME.CBL(R4),C.CMBL(R5)   ; IF ZERO, END OF LIST
2250 026072  000404                            BR      6$                      ;
2251 026074  011402                      5$:   MOV     (R4),R2                 ; POINT TO NEXT PACKET
2252 026076  016462  000002  000002            MOV     ME.CBL(R4),ME.CBL(R2)   ; ADJUST BACK LINK OF NEXT PACKET
2253 026104  013764  006242  000000      6$:   MOV     T.HDR,ME.CFL(R4)        ; ADD ENVELOPE TO
2254 026112  005064  000002                    CLR     ME.CBL(R4)              ;    TOP OF FREE LIST
2255 026116  010437  006242                    MOV     R4,T.HDR                ;
2256 026122  005237  006236                    INC     TBSSIZ                  ; INCREMENT FREE M.E. COUNT
2257 026126  000240                     NOP                                   ; [DBG] INSERT HALT FOR DEBUG
2258 026130  000207                            RETURN                          ;
```

```
2260
2261
2262                                    ;++
2263                                    ;        CLRRSP - PUT AN EMPTY RESPONSE BUFFER IN THE RESPONSE RING.
2264                                    ;
2265                                    ;        INPUTS:
2266                                    ;            R5 - ADDRESS OF CONTROLLER TABLE
2267                                    ;        OUTPUTS:
2268                                    ;            C.RPTR(R5) - CONTAINS POINTER TO RESPONSE BUFFER
2269                                    ;--
2270 026132  016502  000104            CLRRSP: MOV   C.RHDR(R5),R2                ; POINT TO FREE R.E.
2271 026136  016265  000000  000104            MOV   RE.RFL(R2),C.RHDR(R5)        ; POINT TO NEXT R.E.
2272 026144  012762  000200  000002            MOV   #128.,RE.RL(R2)             ; SET RESPONSE PACKET LENGTH
2273 026152  005062  000004                    CLR   RE.RV(R2)                    ; SET CIRCUIT TYPE, ETC
2274 026156  062702  000006                    ADD   #RE.RP,R2                    ; POINT TO TOP OF RESPONSE PACKET
2275 026162  016500  000076                    MOV   C.RPTR(R5),R0                ; GET RING ENTRY ADDRESS
2276 026166  010210                            MOV   R2,(R0)                      ; PUT ADDRESS OF FREE PACKET IN RING
2277 026170  012760  140000  000002            MOV   #<RG.OWN+RG.FLG>,2(R0)       ; RETURN BUFFER TO CONTROLLER
2278 026176  062765  000004  000076            ADD   #4,C.RPTR(R5)                ; UPDATE RESPONSE POINTER
2279 026204  026565  000076  000102            CMP   C.RPTR(R5),C.REND(R5)        ; IF NOT AT END OF RESPONSE RING,
2280 026212  001003                            BNE   15$                          ;    EXIT.
2281 026214  016565  000100  000076            MOV   C.RBAS(R5),C.RPTR(R5)        ; ELSE, POINT TO FIRST ENTRY.
2282 026222  016500  000076            15$:    MOV   C.RPTR(R5),R0                ; GET RING ENTRY ADDRESS
2283 026226  032760  100000  000002            BIT   #RG.OWN,2(R0)                ; IF HOST OWNS ENTRY,
2284 026234  001403                            BEQ   20$                          ;    DON'T CLEAR MESSAGE RECEIVED FLAG
2285 026236  042765  020000  000014            BIC   #CT.MSG,C.FLG(R5)            ; CLEAR MESSAGE RECEIVED FLAG
2286 026244                            20$:
2287 026244  000240                    NOP                                        ; [DBG] INSERT HALT FOR DEBUG
2288 026246  000207                            RETURN                             ;
```

```
2290
2291
2292                                    ;++
2293                                    ;        RANDOM - CALCULATE A PSEUDO RANDOM NUMBER IN THE RANGE 0 - 2↑32-1
2294                                    ;
2295                                    ;        INPUTS:
2296                                    ;                R1 - MAXIMUM VALUE FOR RANDOM NUMBER (LOW WORD)
2297                                    ;                R2 - MAXIMUM VALUE FOR RANDOM NUMBER (HIGH WORD)
2298                                    ;                LOSEED,HISEED - SEED FOR RANDOM NUMBER GENERATOR
2299                                    ;
2300                                    ;        OUTPUTS:
2301                                    ;                R1 - RANDOM NUMBER (LOW WORD)
2302                                    ;                R2 - RANDOM NUMBER (HIGH WORD)
2303                                    ;                LOSEED,HISEED - UPDATED WITH NEW SEED
2304                                    ;--
2305  026250              RANDOM: PUSH    <R0,R3,R4>              ; SAVE REGISTERS
      026250  010046              MOV     R0,-(SP)                ;       PUSH R0 ON STACK
      026252  010346              MOV     R3,-(SP)                ;       PUSH R3 ON STACK
      026254  010446              MOV     R4,-(SP)                ;       PUSH R4 ON STACK
2306  026256              PUSH    <R2,R1>                 ; SAVE MAX VALUE
      026256  010246              MOV     R2,-(SP)                ;       PUSH R2 ON STACK
      026260  010146              MOV     R1,-(SP)                ;       PUSH R1 ON STACK
2307  026262  005003              CLR     R3                      ; INITIALIZE
2308  026264  005004              CLR     R4                      ;       MASK WORDS
2309
2310                                    ;        CALCULATE MASK FOR RANDOM NUMBER
2311                                    ;
2312  026266  006303      1$:     ASL     R3                      ; SHIFT MASK
2313  026270  006104              ROL     R4                      ;       ONE BIT LEFT
2314  026272  005203              INC     R3                      ; TURN ON BIT 0
2315  026274  020402              CMP     R4,R2                   ; SEE IF
2316  026276  103773              BLO     1$                      ;       MASK IS
2317  026300  101002              BHI     2$                      ;               GREATER THAN
2318  026302  020301              CMP     R3,R1                   ;                       MAXIMUM VALUE
2319  026304  103770              BLO     1$                      ; IF NOT, LOOP
2320  026306  005103      2$:     COM     R3                      ;       ELSE, COMPLIMENT MASK
2321  026310  005104              COM     R4                      ;
2322
2323                                    ;        GET RANDOM NUMBER
2324                                    ;
2325  026312  013701  002360  10$:    MOV     LOSEED,R1           ; GET LOW ORDER SEED
2326  026316  013702  002362          MOV     HISEED,R2           ; GET HIGH WORD OF SEED
2327  026322  012700  000007          MOV     #7,R0               ; INIT LOOP COUNT
2328  026326  060101      11$:    ADD     R1,R1                   ; ROTATE LO ORDER NUMBER BY 1
2329  026330  006102              ROL     R2                      ; ROTATE HI ORDER NUMBER BY 1 (PROPOGATE CARRY)
2330  026332  005300              DEC     R0                      ; DECREMENT COUNT
2331  026334  001374              BNE     11$                     ; IF COUNT INCOMPLETE, BRANCH
2332  026336  063701  002360          ADD     LOSEED,R1           ; ADD ORIGINAL SEED (X129)
2333  026342  005502              ADC     R2                      ; PROPOGATE CARRY
2334  026344  063702  002362          ADD     HISEED,R2           ; ADD HISEED
2335  026350  062701  001057          ADD     #1057,R1            ; ADD LO CONSTANT
2336  026354  005502              ADC     R2                      ; PROPOGATE CARRY
2337  026356  062702  047401          ADD     #47401,R2           ; ADD HI CONSTANT
2338  026362  010137  002360          MOV     R1,LOSEED           ; SAVE LO ORDER SEED
2339  026366  010237  002362          MOV     R2,HISEED           ; SAVE HI ORDER SEED
```

```
2340                                  ;
2341                                  ;          MAKE SURE NUMBER IS IN RANGE
2342                                  ;
2343 026372  040301                   BIC     R3,R1               ; GET ONLY
2344 026374  040402                   BIC     R4,R2               ;    NEEDED BITS
2345 026376  020266  000002           CMP     R2,2(SP)            ; CHECK
2346 026402  101343                   BHI     10$                 ;    IF NUMBER
2347 026404  103402                   BLO     20$                 ;       IS IN RANGE
2348 026406  020116                   CMP     R1,(SP)             ;
2349 026410  101340                   BHI     10$                 ;          IF NOT, GET ANOTHER NUMBER
2350 026412  022626           20$:    CMP     (SP)+,(SP)+         ; THROW AWAY MAX VALUE
2351 026414  000240           NOP                         ; [DBG] INSERT HALT FOR DEBUG
2352 026416                            POP     <R4,R3,R0>          ; RESTORE SAVED REGISTERS
     026416  012604                   MOV     (SP)+,R4                    ; POP STACK INTO R4
     026420  012603                   MOV     (SP)+,R3                    ; POP STACK INTO R3
     026422  012600                   MOV     (SP)+,R0                    ; POP STACK INTO R0
2353 026424  000207                   RETURN
```

```
2355
2356                                      ;++
2357                                      ;    DRIVER - THIS ROUTINE IS THE DRIVER FOR THE TESTS. IT CAUSES THE
2358                                      ;    COMMAND PACKETS TO BE GENERATED AND PROCESSES THE RESPONSES.
2359                                      ;
2360                                      ;    INPUTS:
2361                                      ;        R5 - CONTROLLER TABLE ADDRESS
2362                                      ;
2363                                      ;--
2364 026426  013705  002200     DRIVER:  MOV     CTABS,R5                     ; GET CONTROLLER TABLE ADDRESS
2365 026432  032765  110000  000014  1$: BIT     #<CT.AVL!CT.IOC>,C.FLG(R5)   ; IF NOT AVAILABLE FOR TEST,
2366 026440  001072                       BNE     31$                          ;   LOOK AT NEXT CONTROLLER
2367 026442  116537  000002  002074       MOVB    C.UNIT(R5),L$LUN             ; STORE UNIT NUMBER UNDER TEST
2368                                      ;
2369                                      ;    SEND COMMAND TO CONTROLLER
2370                                      ;
2371 026450  032765  040000  000014       BIT     #CT.DUN,C.FLG(R5)            ; IF DONE TESTING THIS CONTROLLER,
2372 026456  001012                       BNE     10$                          ;   DON'T GENERATE ANY MORE PACKETS
2373 026460  032737  010000  002204       BIT     #IDUN,IFLAGS                 ; IF TEST TIMED OUT,
2374 026466  001006                       BNE     10$                          ;   DON'T GENERATE ANYMORE PACKETS
2375 026470  004777  153552               CALL    @TSPGEN                      ; BUILD TEST PACKETS
2376 026474  032765  110000  000014       BIT     #<CT.AVL!CT.IOC>,C.FLG(R5)   ; IF NOT AVAILABLE FOR TEST,
2377 026502  001051                       BNE     31$                          ;   LOOK AT NEXT CONTROLLER
2378 026504  004737  051504     10$:      CALL    SNDMSG                       ; SEND NEXT COMMAND TO CONTROLLER
2379                                      ;
2380                                      ;    GET RESPONSE FROM CONTROLLER
2381                                      ;
2382 026510  032765  020000  000014       BIT     #CT.MSG,C.FLG(R5)            ; IF MESSAGE FLAG SET,
2383 026516  001020                       BNE     20$                          ;   PROCESS RESPONSE PACKET.
2384 026520  004737  027140               CALL    CTLCHK                       ; ELSE, IF CONTROLLER NOT RUNNING,
2385 026524  103440                        BCS     31$                          ;   EXIT LOOP
2386 026526  005765  000164               TST     C.TBSF(R5)                   ; IF TBS LIST IS NOT EMPTY,
2387 026532  001014                       BNE     30$                          ;   CONTINUE
2388 026534  005765  000160               TST     C.CMFL(R5)                   ; IF COMMAND LIST IS NOT EMPTY,
2389 026540  001011                       BNE     30$                          ;   CONTINUE
2390 026542  032765  000400  000014       BIT     #CT.MRW,C.FLG(R5)            ; IF MAINTENANCE OPERATION IN
2391 026550  001005                       BNE     30$                          ;   PROGRESS, DON'T DO GUS
2392 026552  004737  050114               CALL    GUSTAT                       ; ISSUE GET UNIT STATUS TO KEEP
2393 026556  000402                       BR      30$                          ;   CONTROLLER ALIVE
2394
2395 026560  004737  027500     20$:      CALL    RSPIN                        ; ELSE, LOOK AT PACKET
2396                                      ;
2397                                      ;    IF TEST NOT DONE, SWITCH TO NEXT CONTROLLER
2398                                      ;    ELSE, EXIT TEST ROUTINE
2399                                      ;
2400 026564  032765  040000  000014  30$: BIT     #CT.DUN,C.FLG(R5)            ; IF TEST NOT DONE ON CONTROLLER,
2401 026572  001421                       BEQ     35$                          ;   CONTINUE
2402 026574  005765  000164               TST     C.TBSF(R5)                   ; IF TBS LIST IS NOT EMPTY,
2403 026600  001016                       BNE     35$                          ;   CONTINUE
2404 026602  005765  000160               TST     C.CMFL(R5)                   ; IF COMMAND LIST IS NOT EMPTY,
2405 026606  001013                       BNE     35$                          ;   CONTINUE
2406 026610  004737  053452               CALL    RESET                        ; RESET CONTROLLER & RE-INIT TABLE
2407 026614  052765  010000  000014       BIS     #CT.IOC,C.FLG(R5)            ; DISABLE CONTROLLER
2408 026622  005337  002210               DEC     UTEST                        ; DECREMENT CONTROLLER UNDER TEST COUNT
2409 026626  005737  002210     31$:      TST     UTEST                        ; IF UNITS UNDER TEST ZERO,
```

```
2410 026632  003001                           BGT     35$              ; EXIT ROUTINE
2411 026634  000207                           RETURN                   ;
2412
2413 026636  062705  000170          35$:     ADD     #C.SIZE,R5       ; MOVE TO NEXT TABLE
2414 026642  005715                           TST     (R5)             ; IF NOT END OF CONTROLLER LIST,
2415 026644  001272                           BNE     1$               ;   LOOK AT NEXT CONTROLLER
2416                                  ;
2417                                  ;        CHECK FOR TEST TIMEOUT
2418                                  ;
2419 026646  022737  000003  002206  40$:     CMP     #3.,TNUM         ; IF THIS IS NOT TEST 3,
2420 026654  001125                           BNE     60$              ;   DON'T TIMEOUT TEST OR PRINT REPORT
2421 026656  032737  010000  002204           BIT     #IDUN,IFLAGS     ; IF WE'RE WAITING FOR CONTROLLERS
2422 026664  001101                           BNE     50$              ;   TO FINISH, DON'T CHECK TIMEOUT
2423 026666  005737  002146                   TST     SFPTBL+SO.TL          ; IF TEST TIMEOUT IS ZERO,
2424 026672  001476                           BEQ     50$              ;   DON'T TIMEOUT TEST
2425 026674  005737  002322                   TST     KW.CSR           ; SEE IF A CLOCK ON SYSTEM
2426 026700  001473                           BEQ     50$              ; DON'T TIME OUT IF NO CLOCK
2427 026702  023737  002334  002256           CMP     KW.EL+2,TSTIM+2  ; CHECK HIGH WORD OF ELAPSED TIME
2428 026710  101005                           BHI     41$              ; IF GREATER, TEST DONE
2429 026712  001066                           BNE     50$              ; IF NOT SAME, KEEP TESTING
2430 026714  023737  002332  002254           CMP     KW.EL,TSTIM      ; CHECK LOW WORD OF ELAPSED TIME
2431 026722  103462                           BLO     50$              ; IF LESS, KEEP TESTING
2432 026724  052737  010000  002204  41$:     BIS     #IDUN,IFLAGS     ; SET TEST DONE BIT
2433 026732                                   PRINTF  #TIMLIM          ; PRINT TIMEOUT MESSAGE
     026732  012746  010453                   MOV     #TIMLIM,-(SP)
     026736  012746  000001                   MOV     #1.,-(SP)
     026742  010600                           MOV     SP,R0
     026744  104417                           TRAP    C$PNTF
     026746  062706  000004                   ADD     #4,SP
2434 026752  004737  053550                   CALL    RNTIME           ; GET RUN TIME
2435 026756                                   PRINTF  #RNTIM,RNTHH,RNTMM,RNTSS; PRINT RUN TIME
     026756  013746  002272                   MOV     RNTSS,-(SP)
     026762  013746  002270                   MOV     RNTMM,-(SP)
     026766  013746  002266                   MOV     RNTHH,-(SP)
     026772  012746  010303                   MOV     #RNTIM,-(SP)
     026776  012746  000004                   MOV     #4,-(SP)
     027002  010600                           MOV     SP,R0
     027004  104417                           TRAP    C$PNTF
     027006  062706  000012                   ADD     #12,SP
2436 027012                                   PRINTF  #CRLF            ; TERMINATE LINE
     027012  012746  010300                   MOV     #CRLF,-(SP)
     027016  012746  000001                   MOV     #1,-(SP)
     027022  010600                           MOV     SP,R0
     027024  104417                           TRAP    C$PNTF
     027026  062706  000004                   ADD     #4,SP
2437 027032  013705  002200                   MOV     CTABS,R5         ; GET CONTROLLER TABLE ADDRESS
2438 027036  052765  040000  000014  42$:     BIS     #CT.DUN,C.FLG(R5); MARK CONTROLLER DONE
2439 027044  005065  000030                   CLR     C.STEP(R5)       ; RESET TEST STEP COUNTER
2440 027050  005065  000042                   CLR     C.NEXT(R5)       ; RESET NEXT UNIT INDEX
2441 027054  005065  000032                   CLR     C.UCNT(R5)       ; RESET UNIT UNDER TEST COUNT
2442 027060  062705  000170                   ADD     #C.SIZE,R5       ; GET NEXT CONTROLLER TABLE ADDRESS
2443 027064  005715                           TST     (R5)             ; CHECK IF ANOTHER CONTROLLER TABLE
2444 027066  001363                           BNE     42$              ; DO NEXT CONTROLLER
2445                                  ;
2446                                  ;        CHECK FOR TIME TO PRINT STATISTICAL REPORT
```

```
2447                                          ;
2448 027070  005737  002150        50$:   TST    SFPTBL+SO.SRI        ; IF STATISTICAL REPORT INTERVAL = 0,
2449 027074  001415                        BEQ    60$                 ;    DON'T PRINT REPORTS
2450 027076  005737  002322                TST    KW.CSR              ; ANY CLOCK ON SYSTEM?
2451 027102  001412                        BEQ    60$                 ;      BYPASS IF NOT
2452 027104  023737  002334  002252        CMP    KW.EL+2,STIME+2     ; CHECK HIGH WORD OF ELAPSED TIME
2453 027112  101005                        BHI    51$                 ; IF GREATER PRINT REPORT
2454 027114  001005                        BNE    60$                 ; IF NOT SAME, ITS NOT TIME YET
2455 027116  023737  002332  002250        CMP    KW.EL,STIME         ; CHECK LOW WORD OF ELAPSED TIME
2456 027124  103401                        BLO    60$                 ; IF LESS, ITS NOT TIME YET
2457 027126                        51$:   DORPT                       ; PRINT A STATISTICAL REPORT
     027126  104424                        TRAP   C$DRPT
2458
2459 027130                        60$:   BREAK                       ; >>>>>>>BREAK BACK TO MONITOR<<<<<<<
     027130  104422                        TRAP   C$BRK
2460 027132  000240                        NOP              ; [DBG] INSERT HALT HERE FOR DEBUG
2461 027134  000137  026426                JMP    DRIVER              ; LOOK AT FIRST CONTROLLER AGAIN
```

```
2463                             ;++
2464                             ;    CTLCHK - VERIFY CONTROLLER IS STILL FUNCTIONING BY MAKING
2465                             ;    SURE SA REGISTER IS ZERO AND OLDEST OUTSTANDING COMMAND IS
2466                             ;    MAKING PROGRESS.
2467                             ;
2468                             ;    INPUTS:
2469                             ;           R5 - POINTER TO CONTROLLER TABLE
2470                             ;--
2471
2472 027140 011504              CTLCHK: MOV   (R5),R4              ; GET ADDRESS OF IP REGISTER
2473 027142 016402 000002               MOV   2(R4),R2             ; LOOK AT SA REGISTER
2474 027146 001407                       BEQ   20$                  ; IF ZERO, CONTROLLER STILL RUNNING
2475 027150                              ERRDF 30,,ERR030           ; ELSE, REPORT FATAL CONTROLLER ERROR
     027150 104455                       TRAP  C$ERDF
     027152 000036                       .WORD 30
     027154 000000                       .WORD 0
     027156 020246                       .WORD ERR030
2476 027160 004737 027350               CALL  DRPCNT               ; DROP CONTROLLER FROM TESTING
2477 027164 000207                      RETURN                     ;
2478
2479 027166 016504 000162       20$:    MOV   C.CMBL(R5),R4         ; GET POINTER TO OLDEST COMMAND
2480 027172 001463                      BEQ   30$                  ; IF LIST EMPTY, DON'T TIME OUT
2481 027174 005737 002322               TST   KW.CSR               ; IF NO CLOCK ON SYSTEM,
2482 027200 001460                      BEQ   30$                  ;    DON'T TIME OUT COMMAND
2483 027202 023764 002334 000010        CMP   KW.EL+2,ME.CTH(R4)   ; CHECK HIGH WORD OF ELAPSED TIME
2484 027210 101005                      BHI   23$                  ; IF GREATER, RESPONSE TIMED OUT
2485 027212 103453                      BLO   30$                  ; IF LOWER, ITS OK
2486 027214 023764 002332 000006        CMP   KW.EL,ME.CTO(R4)     ; CHECK LOW WORD OF ELAPSED TIME
2487 027222 103447                      BLO   30$                  ; IF LESS, PLENTY OF TIME LEFT
2488 027224 032765 000400 000014 23$:   BIT   #CT.MRW,C.FLG(R5)    ; IF MAINTENANCE OPERATION IN
2489 027232 001004                      BNE   24$                  ;    PROGRESS, TIMEOUT CONTROLLER
2490 027234 032764 000070 000036        BIT   #70,P.OPCD+ME.CP(R4) ; IF NOT IMMEDIATE COMMAND
2491 027242 001007                      BNE   25$                  ;    SEND GCS PACKET
2492 027244                      24$:    ERRDF 31,,ERR031           ; REPORT TIMEOUT ERROR
     027244 104455                       TRAP  C$ERDF
     027246 000037                       .WORD 31
     027250 000000                       .WORD 0
     027252 020264                       .WORD ERR031
2493 027254 004737 027350               CALL  DRPCNT               ; DROP CONTROLLER FROM TESTING
2494 027260 000207                      RETURN                     ;
2495
2496 027262 032764 040000 000016 25$:   BIT   #CF.CTO,ME.CFG(R4)   ; IF GCS COMMAND ALREADY ISSUED,
2497 027270 001365                      BNE   24$                  ;    CONTROLLER MUST BE HUNG
2498 027272 016501 000006               MOV   C.CTO(R5),R1         ; RE-INIT
2499 027276 010400                      MOV   R4,R0                ;    COMMAND
2500 027300 062700 000006               ADD   #ME.CTO,R0           ;       TIMER
2501 027304 004737 052070               CALL  SETTO                ;
2502 027310 052764 040000 000016        BIS   #CF.CTO,ME.CFG(R4)   ; SET GCS ISSUED FLAG
2503 027316 016401 000004               MOV   ME.CRF(R4),R1        ; GET CMD REF # OF OUTSTANDING COMMAND
2504 027322 005002                      CLR   R2                   ; CLEAR HIGH WORD
2505 027324 004737 050056               CALL  GCSTAT               ; SEND GCS COMMAND
2506 027330 103004                      BCC   30$                  ; IF NOT SUCCESSFUL,
2507 027332 012701 027330               MOV   #.-2,R1              ; SAVE CURRENT LOCATION
2508 027336 000137 024310               JMP   BADMEM               ; PRINT SYSTEM FATAL ERROR
2509
```

```
2510 027342 000241          30$:    CLC
2511 027344 000240          NOP                     ; [DBG] INSERT HALT FOR DEBUG
2512 027346 000207                  RETURN                          ;
2513
```

```
2515                                        ;++
2516                                        ;        DRPCNT - DROP A CONTROLLER AFTER FATAL ERROR REPORTED BY MSCP
2517                                        ;--
2518
2519 027350 032765 100000 000014  DRPCNT:  BIT     #CT.AVL,C.FLG(R5)       ; IF CONTROLLER ALREADY DISABLED,
2520 027356 001017                          BNE     2$                      ;   EXIT ROUTINE
2521 027360 004737 053452                   CALL    RESET                   ; RESET THE CONTROLLER
2522 027364 052765 100000 000014            BIS     #CT.AVL,C.FLG(R5)       ; DISABLE CONTROLLER
2523 027372 004737 025606                   CALL    RTNALL                  ; RETURN BUFFERS TO FREE LIST
2524 027376                                 ERRDF   42,,ERR042              ; PRINT CONTROLLER DROPPED ERROR
     027376 104455                           TRAP    C$ERDF
     027400 000052                           .WORD   42
     027402 000000                           .WORD   0
     027404 021462                           .WORD   ERR042
2525 027406 005337 002210                   DEC     UTEST                   ; REDUCE CONTROLLERS UNDER TEST COUNT
2526 027412 001401                          BEQ     2$                      ; IF MORE CONTROLLERS TO TEST,
2527 027414                                 DORPT                           ;   PRINT STATISTICAL REPORT
     027414 104424                           TRAP    C$DRPT
2528 027416 000261                  2$:     SEC                             ; INDICATE ERROR STATUS
2529 027420 000240                          NOP             ; [DBG] INSERT HALT FOR DEBUG
2530 027422 000207                          RETURN                          ;
2531
2532                                        ;++
2533                                        ;        DRPUNT - DROP A DRIVE FROM TESTING
2534                                        ;
2535                                        ;
2536                                        ;        INPUTS:
2537                                        ;            R1 - POINTER TO DRIVE TABLE
2538                                        ;        OUTPUTS:
2539                                        ;            DT.AVL FLAG SET IN D.UNIT FIELD OF DRIVE TABLE
2540                                        ;            D.CCNT & D.WCNT FIELDS CLEARED
2541                                        ;            TEMPORARY DRIVE FLAGS CLEARED
2542                                        ;--
2543 027424 032761 100000 000002  DRPUNT:  BIT     #DT.AVL,D.UNIT(R1)      ; IF DRIVE ALREADY DROPPED,
2544 027432 001017                          BNE     1$                      ;   SKIP ERROR REPORT
2545 027434 042761 147000 000002            BIC     #DT.CLR,D.UNIT(R1)      ; CLEAR TEMPORARY DRIVE FLAGS
2546 027442 005061 000060                   CLR     D.CCNT(R1)              ; ZERO "COMPARE HOST DATA" COUNT
2547 027446 005061 000056                   CLR     D.WCNT(R1)              ; ZERO OUTSTANDING WRITE COUNT
2548 027452 052761 100000 000002            BIS     #DT.AVL,D.UNIT(R1)      ; DISABLE DRIVE
2549 027460                                 ERRDF   43,,ERR043              ; PRINT UNIT DROPPED ERROR
     027460 104455                           TRAP    C$ERDF
     027462 000053                           .WORD   43
     027464 000000                           .WORD   0
     027466 021626                           .WORD   ERR043
2550 027470                                 DORPT                           ; PRINT STATISTICAL REPORT
     027470 104424                           TRAP    C$DRPT
2551 027472 000261                  1$:     SEC                             ; INDICATE ERROR STATUS
2552 027474 000240                          NOP             ; [DBG] INSERT HALT FOR DEBUG
2553 027476 000207                          RETURN                          ;
```

```
2555                             ;++
2556                             ;       RSPIN - CONTROLLER HAS RESPONDED. REMOVE FULL RESPONSE
2557                             ;       PACKET FROM RESPONSE RING. INSERT A NEW RESPONSE PACKET AND
2558                             ;       RETURN THE RING TO THE CONTROLLER. PROCESS THE RESPONSE
2559                             ;       PACKET.
2560                             ;
2561                             ;       INPUTS:
2562                             ;             R5 - POINTER TO CONTROLLER TABLE
2563                             ;--
2564
2565 027500  004737  026132    RSPIN:  CALL    CLRRSP              ; RE-INIT RESPONSE RING
2566 027504  016504  000104            MOV     C.RHDR(R5),R4       ; POINT TO FULL R.E.
2567 027510  016401  000012            MOV     P.UNIT+RE.RP(R4),R1 ; GET UNIT NUMBER
2568 027514  016402  000020            MOV     P.STS+RE.RP(R4),R2  ; GET STATUS/EVENT CODE
2569 027520  116403  000016            MOVB    P.OPCD+RE.RP(R4),R3 ; GET
2570 027524  042703  177400            BIC     #177400,R3          ;   ENDCODE
2571 027530  032703  000200            BIT     #200,R3             ; IF END MESSAGE,
2572 027534  001007                    BNE     RSPEND              ;   RESPOND TO END MSG
2573 027536  032703  000100            BIT     #100,R3             ; IF ATTENTION MESSAGE,
2574 027542  001002                    BNE     20$                 ;   RESPOND TO ATTENTION MESSAGE
2575 027544  000137  040036            JMP     RSPERL              ;   ELSE, RESPOND TO ERROR LOG MESSAGE
2576 027550  000137  037704    20$:    JMP     RSPATN              ;
```

```
2578
2579
2580                                      ;++
2581                                      ;          RSPEND - RESPOND TO END MESSAGE
2582                                      ;
2583                                      ;          INPUTS:
2584                                      ;                   R1 - DRIVE UNIT NUMBER
2585                                      ;                   R2 - STATUS/EVENT CODE
2586                                      ;                   R3 - ENDCODE
2587                                      ;                   R4 - POINTER TO RESPONSE PACKET
2588                                      ;                   R5 - CONTROLLER TABLE ADDRESS
2589                                      ;          OUTPUTS:
2590                                      ;                   RESPONSE PACKET CLEARED, REPONSE RING OWNED BY CONTROLLER.
2591                                      ;                   CONTENTS OF REGISTERS R1,R2,R3,R4 UNDEFINED
2592                                      ;                   R5 UNCHANGED
2593                                      ;--
2594 027554 032765 000400 000014 RSPEND: BIT  #CT.MRW,C.FLG(R5)   ; IF MAINTENANCE OPERATION IN
2595 027562 001006                        BNE  1$                   ;   PROGRESS, DON'T ADJUST CREDITS
2596 027564 016400 000004                 MOV  RE.RV(R4),R0         ; SAVE
2597 027570 042700 177760                 BIC  #^C<17>,R0           ;   CREDIT
2598 027574 060065 000152                 ADD  R0,C.CRED(R5)        ;     COUNT
2599 027600 005000                 1$:    CLR  R0                   ; CALCULATE
2600 027602 026003 027630         2$:    CMP  ENDNUM(R0),R3        ;   DISPATCH
2601 027606 001002                        BNE  3$                   ;     TABLE
2602 027610 000170 027676                 JMP  @ENDADR(R0)          ;       OFFSET
2603 027614 062700 000002         3$:    ADD  #2,R0                ;
2604 027620 020027 000044                 CMP  R0,#ENDSIZ           ;
2605 027624 003766                        BLE  2$                   ;
2606 027626 000446                        BR   INVCMD               ; ENDCODE NOT IN LIST
2607
2608                                      ;
2609                                      ;          END MESSAGE RESPONSE DISPATCH TABLE
2610                                      ;
2611
2612 027630 000242         ENDNUM: .WORD  242                      ; 242 - WRITE END PACKET
2613 027632 000241                 .WORD  241                      ; 241 - READ END PACKET
2614 027634 000222                 .WORD  222                      ; 222 - ERASE END PACKET
2615 027636 000220                 .WORD  220                      ; 220 - ACCESS END PACKET
2616 027640 000231                 .WORD  231                      ; 231 - MAINTENANCE WRITE
2617 027642 000230                 .WORD  230                      ; 230 - MAINTENANCE READ
2618 027644 000240                 .WORD  240                      ; 240 - COMPARE HOST DATA
2619 027646 000202                 .WORD  202                      ; 202 - GET COMMAND STATUS END PACKET
2620 027650 000203                 .WORD  203                      ; 203 - GET UNIT STATUS END PACKET
2621 027652 000204                 .WORD  204                      ; 204 - SET CONTROLLER CHARACTERISTICS
2622 027654 000210                 .WORD  210                      ; 210 - AVAILABLE END PACKET
2623 027656 000211                 .WORD  211                      ; 211 - ONLINE END PACKET
2624 027660 000212                 .WORD  212                      ; 212 - SET UNIT CHARACTERISTICS
2625 027662 000201                 .WORD  201                      ; 201 - ABORT END PACKET
2626 027664 000205                 .WORD  205                      ; 207 - SERIOUS EXCEPTION
2627 027666 000213                 .WORD  213                      ; 213 - DETERMINE ACCESS PATH
2628 027670 000221                 .WORD  221                      ; 221 - COMPARE CONTROLLER DATA
2629 027672 000223                 .WORD  223                      ; 223 - FLUSH END PACKET
2630 027674 000224                 .WORD  224                      ; 224 - REPLACE END PACKET
2631
2632        000044         ENDSIZ =      <.-ENDNUM-2>             ; LEGAL NUMBERS ARE LOWER THAN THIS
```

```
2633
2634 027676 032166              ENDADR: .WORD     ENDWR         ; 242 - WRITE END PACKET
2635 027700 031720                      .WORD     ENDRD         ; 241 - READ END PACKET
2636 027702 031120                      .WORD     ENDERS        ; 222 - ERASE END PACKET
2637 027704 031004                      .WORD     ENDACC        ; 220 - ACCESS END PACKET
2638 027706 031510                      .WORD     ENDMWR        ; 231 - MAINTENANCE WRITE
2639 027710 031240                      .WORD     ENDMRD        ; 230 - MAINTENANCE READ
2640 027712 031600                      .WORD     ENDCMP        ; 240 - COMPARE HOST DATA
2641 027714 027774                      .WORD     ENDGCS        ; 202 - GET COMMAND STATUS END PACKET
2642 027716 030152                      .WORD     ENDGUS        ; 203 - GET UNIT STATUS END PACKET
2643 027720 030200                      .WORD     ENDSCC        ; 204 - SET CONTROLLER CHARACTERISTICS
2644 027722 030274                      .WORD     ENDAVL        ; 210 - AVAILABLE END PACKET
2645 027724 030364                      .WORD     ENDONL        ; 211 - ONLINE END PACKET
2646 027726 030714                      .WORD     ENDSUC        ; 212 - SET UNIT CHARACTERISTICS
2647 027730 027744                      .WORD     INVCMD        ; 201 - ABORT END PACKET
2648 027732 027744                      .WORD     INVCMD        ; 207 - SERIOUS EXCEPTION
2649 027734 027744                      .WORD     INVCMD        ; 213 - DETERMINE ACCESS PATH
2650 027736 027744                      .WORD     INVCMD        ; 221 - COMPARE CONTROLLER DATA
2651 027740 027744                      .WORD     INVCMD        ; 223 - FLUSH END PACKET
2652 027742 027744                      .WORD     INVCMD        ; 224 - REPLACE END PACKET
2653
2654                             ;++
2655                             ;       INVCMD - REPORT INVALID END MESSAGE RECEIVED AND DROP CONTROLLER
2656                             ;
2657                             ;--
2658
2659 027744                     INVCMD:
2661 027744 012700 023033               MOV      #DBGINV,R0     ; [DEBUG] PRINT RECEIVED MSG TYPE
2662 027750 004737 024132               CALL     DBGRSP         ; [DEBUG] & BUFFER CONTENTS
2664 027754                             ERRDF    32,,ERR032     ; PRINT BAD ENDCODE ERROR
     027754 104455                       TRAP     C$ERDF
     027756 000040                       .WORD    32
     027760 000000                       .WORD    0
     027762 020312                       .WORD    ERR032
2665 027764 004737 027350               CALL     DRPCNT         ; DROP CONTROLLER FROM TEST
2666 027770 000240             NOP       RETURN                  ; [DBG] INSERT HALT FOR DEBUG
2667 027772 000207                                              ;
2668
2669                             ;++
2670                             ;       ENDGCS - "GET COMMAND STATUS" END MESSAGE RECEIVED
2671                             ;
2672                             ;       IF NO CHANGE IN STATUS, COMMAND HAS TIMED OUT.
2673                             ;       ELSE, SAVE NEW STATUS AND RE-INIT COMMAND TIMER.
2674                             ;--
2675
2676 027774                     ENDGCS:
2678 027774 012700 023057               MOV      #DBGGCS,R0     ; [DEBUG] PRINT RECEIVED MSG TYPE
2679 030000 004737 024132               CALL     DBGRSP         ; [DEBUG] & BUFFER CONTENTS
2681 030004 012737 007745 002314        MOV      #OPCGCS,MSCPOP ; POINT TO OPCODE MESSAGE STRING
2682 030012 005764 000026               TST      P.CMST+RE.RP(R4) ; IF COMMAND
2683 030016 001003                      BNE      1$             ;   STATUS IS ZERO,
2684 030020 005764 000030               TST      P.CMST+2+RE.RP(R4) ;   COMMAND IS DONE SO
2685 030024 001446                      BEQ      22$            ;     EXIT
2686 030026 016401 000022     1$:        MOV      P.OTRF+RE.RP(R4),R1 ; GET MESSAGE ENVELOPE
2687 030032 010400                       MOV      R4,R0          ;
```

```
2688 030034  004737  025550              CALL    GETME              ;         FOR OUTSTANDING COMMAND
2689 030040  026064  000030  000014      CMP     P.CMST+2+RE.RP(R0),ME.ST2(R4); CHECK
2690 030046  103415                      BLO     20$                ; IF PROGRESS
2691 030050  101004                      BHI     10$                ;         HAS BEEN
2692 030052  026064  000026  000012      CMP     P.CMST+RE.RP(R0),ME.ST1(R4); MADE ON
2693 030060  103410                      BLO     20$                ;           COMMAND
2694 030062                      10$:    ERRDF   31,,ERRO31         ; REPORT TIMEOUT ERROR
     030062  104455                      TRAP    C$ERDF
     030064  000037                      .WORD   31
     030066  000000                      .WORD   0
     030070  020264                      .WORD   ERRO31
2695 030072  004737  027350              CALL    DRPCNT             ; DROP CONTROLLER FROM TEST
2696 030076  000240                      NOP                ; [DBG] INSERT HALT FOR DEBUG
2697 030100  000207                      RETURN             ;
2698
2699 030102  016064  000026  000012 20$: MOV     P.CMST+RE.RP(R0),ME.ST1(R4); SAVE LATEST
2700 030110  016064  000030  000014      MOV     P.CMST+2+RE.RP(R0),ME.ST2(R4);   COMMAND STATUS
2701 030116  042764  040000  000016      BIC     #CF.CTO,ME.CFG(R4) ; CLEAR GCS COMMAND ISSUED FLAG
2702 030124  016501  000006              MOV     C.CTO(R5),R1       ; RE-INIT
2703 030130  010400                      MOV     R4,R0              ;       COMMAND
2704 030132  062700  000006              ADD     #ME.CTO,R0         ;        TIMER
2705 030136  004737  052070              CALL    SETTO
2706 030142  004737  026006         22$: CALL    RTNME              ; RETURN M.E.
2707 030146  000240                      NOP                ; [DBG] INSERT HALT FOR DEBUG
2708 030150  000207                      RETURN             ;
2709
2710                              ;++
2711                              ;       ENDGUS - "GET UNIT STATUS" END MESSAGE RECEIVED
2712                              ;       THE RESPONSE PACKET IS IGNORED AND THE STATUS RETURNED IS NOT
2713                              ;       CHECKED.
2714                              ;--
2715
2716 030152                      ENDGUS:
2718 030152  012700  023127              MOV     #DBGGUS,R0         ; [DEBUG] PRINT RECEIVED MSG TYPE
2719 030156  004737  024132              CALL    DBGRSP             ; [DEBUG] & BUFFER CONTENTS
2720 030162  012737  007772  002314      MOV     #OPCGUS,MSCPOP     ; POINT TO OPCODE MESSAGE STRING
2722 030170  004737  026006              CALL    RTNME              ; RETURN M.E. & CLEAR RESPONSE RING
2723 030174                      10$:
2724 030174  000240                      NOP                ; [DBG] INSERT HALT FOR DEBUG
2725 030176  000207                      RETURN             ;
2726
2727                              ;++
2728                              ;       ENDSCC - "SET CONTROLLER CHARACTERISTICS" END MESSAGE RECEIVED
2729                              ;
2730                              ;--
2731
2732 030200                      ENDSCC:
2734 030200  012700  023174              MOV     #DBGSCC,R0         ; [DEBUG] PRINT RECEIVED MSG TYPE
2735 030204  004737  024132              CALL    DBGRSP             ; [DEBUG] & BUFFER CONTENTS
2737 030210  012737  010046  002314      MOV     #OPCSCC,MSCPOP     ; POINT TO OPCODE MESSAGE STRING
2738 030216  005001                      CLR     R1                 ; TABLE POINTER SANITY CHECK ON CHKSTA
2739 030220  004737  032332              CALL    CHKSTA             ; CHECK END MESSAGE STATUS
2740 030224  103421                      BCS     20$                ;   IF ERROR, EXIT
2741 030226  016465  000026  000006      MOV     P.CTMO+RE.RP(R4),C.CTO(R5); GET CONTROLLER TIME OUT INTERVAL
2742 030234  016465  000032  000060      MOV     P.CNTI+RE.RP(R4),C.UID(R5)      ; GET
```

```
2743 030242 016465 000034 000062          MOV    P.CNTI+2+RE.RP(R4),C.UID+2(R5) ;    CONTROLLER
2744 030250 016465 000036 000064          MOV    P.CNTI+4+RE.RP(R4),C.UID+4(R5) ;        ID
2745 030256 016465 000040 000066          MOV    P.CNTI+6+RE.RP(R4),C.UID+6(R5) ;
2746 030264 004737 026006                 CALL   RTNME                    ; RETURN M.E. & CLEAR RESPONSE RING
2747 030270                          20$:
2748 030270 000240                   NOP                                  ; [DBG] INSERT HALT FOR DEBUG
2749 030272 000207                        RETURN                          ;
2750
2751                                  ;++
2752                                  ;         ENDAVL - "AVAILABLE" END MESSAGE RECEIVED
2753                                  ;
2754                                  ;--
2755
2756 030274                          ENDAVL:
2758 030274 012700 023245                 MOV    #DBGAVL,R0               ; [DEBUG] PRINT RECEIVED MSG TYPE
2759 030300 004737 024132                 CALL   DBGRSP                  ; [DEBUG] & BUFFER CONTENTS
2761 030304 012737 007603 002314          MOV    #OPCAVL,MSCPOP          ; POINT TO OPCODE MESSAGE STRING
2762 030312 004737 047164                 CALL   GTDRVT                  ; GET DRIVE TABLE
2763 030316 103007                        BCC    1$                      ;    IF TABLE NOT FOUND, PRINT ERROR
2764 030320                               ERRDF  35,,ERRO35              ; UNKNOWN DRIVE ERROR
     030320 104455                         TRAP   C$ERDF
     030322 000043                         .WORD  35
     030324 000000                         .WORD  0
     030326 020454                         .WORD  ERRO35
2765 030330 004737 027350                 CALL   DRPCNT                  ; DROP CONTROLLER FROM TEST
2766 030334 000411                         BR     20$                     ; EXIT
2767 030336 032761 100000 000002 1$:       BIT    #DT.AVL,D.UNIT(R1)      ; IF DRIVE DROPPED,
2768 030344 001003                         BNE    10$                     ;    IGNORE RESPONSE PACKET
2769 030346 004737 032332                 CALL   CHKSTA                  ; CHECK RETURN STATUS
2770 030352 103402                         BCS    20$                     ;    IF ERROR, EXIT
2771 030354 004737 026006            10$:   CALL   RTNME                   ; RETURN M.E. & CLEAR RESPONSE RING
2772 030360                          20$:
2773 030360 000240                   NOP                                  ; [DBG] INSERT HALT FOR DEBUG
2774 030362 000207                        RETURN                          ;
2775
2776                                  ;++
2777                                  ;         ENDONL - "ONLINE" END MESSAGE RECEIVED
2778                                  ;
2779                                  ;   1. SET DRIVE ONLINE FLAG
2780                                  ;   2. SET UNIQUE IDENTIFIER
2781                                  ;   3. SET VOLUME SERIAL NUMBER
2782                                  ;   4. IF B/E SETS SPECIFIED, VERIFY THEY ARE IN RANGE
2783                                  ;      ELSE, STORE UNIT SIZE IN D.END1(R0))
2784                                  ;   5. SET NEXT LBN TO ACCESS (D.LBN(R0)) TO 0
2785                                  ;--
2786
2787 030364                          ENDONL:
2789 030364 012700 023304                 MOV    #DBGONL,R0              ; [DEBUG] PRINT RECEIVED MSG TYPE
2790 030370 004737 024132                 CALL   DBGRSP                  ; [DEBUG] & BUFFER CONTENTS
2792 030374 012737 010014 002314          MOV    #OPCONL,MSCPOP          ; POINT TO OPCODE MESSAGE STRING
2793 030402 004737 047164                 CALL   GTDRVT                  ; GET DRIVE TABLE
2794 030406 103007                        BCC    1$                      ;    IF TABLE NOT FOUND, PRINT ERROR
2795 030410                               ERRDF  35,,ERRO35              ; UNKNOWN DRIVE ERROR
     030410 104455                         TRAP   C$ERDF
     030412 000043                         .WORD  35
```

```
          030414  000000                              .WORD   0
          030416  020454                              .WORD   ERRO35
2796 030420  004737  027350                   CALL   DRPCNT              ; DROP CONTROLLER FROM TEST
2797 030424  000514                           BR     20$                 ; EXIT
2798 030426  032761  100000  000002  1$:      BIT    #DT.AVL,D.UNIT(R1)  ; IF DRIVE DROPPED,
2799 030434  001106                           BNE    10$                 ;    IGNORE RESPONSE PACKET
2800 030436  004737  032332                   CALL   CHKSTA              ; CHECK END MESSAGE STATUS
2801 030442  103505                           BCS    20$                 ;    IF ERROR, EXIT
2802 030444  052761  001000  000002           BIS    #DT.ONL,D.UNIT(R1)  ; SET DRIVE ONLINE FLAG
2803 030452  016461  000032  000122           MOV    P.UNTI+RE.RP(R4),D.UID(R1); SET
2804 030460  016461  000034  000124           MOV    P.UNTI+2+RE.RP(R4),D.UID+2(R1);    UNIQUE
2805 030466  016461  000036  000126           MOV    P.UNTI+4+RE.RP(R4),D.UID+4(R1);       IDENTIFIER
2806 030474  016461  000040  000130           MOV    P.UNTI+6+RE.RP(R4),D.UID+6(R1);
2807 030502  016461  000056  000132           MOV    P.VSER+RE.RP(R4),D.VSN(R1); SET VOLUME
2808 030510  016461  000060  000134           MOV    P.VSER+2+RE.RP(R4),D.VSN+2(R1);    SERIAL NUMBER
2809
2810 030516  032761  020000  000002           BIT    #DT.BES,D.UNIT(R1)  ; IF TESTING ENTIRE DISK,
2811 030524  001413                           BEQ    2$                  ;    SKIP
2812 030526  016461  000052  000012           MOV    P.UNSZ+RE.RP(R4),D.END1(R1); ELSE,
2813 030534  016461  000054  000014           MOV    P.UNSZ+2+RE.RP(R4),D.END1+2(R1);   SAVE
2814 030542  162761  000001  000012           SUB    #1,D.END1(R1)       ;    LAST
2815 030550  005661  000014                   SBC    D.END1+2(R1)        ;       LBN
2816 030554  016161  000006  000052  2$:      MOV    D.BGN1(R1),D.LBN(R1); SET NEXT LBN TO ACCESS TO
2817 030562  016161  000010  000054           MOV    D.BGN1+2(R1),D.LBN+2(R1);   FIRST BLOCK IN FIRST BEGIN/END SET
2818 030570  042761  042000  000002           BIC    #<DT.EOM+DT.DUN>,D.UNIT(R1); CLEAR DONE TESTING & EOM FLAG
2819 030576  032761  020000  000002           BIT    #DT.BES,D.UNIT(R1)  ; IF NOT USING ENTIRE DISK,
2820 030604  001022                           BNE    10$                 ;    VERIFY THEY ARE IN RANGE
2821 030606  010102                           MOV    R1,R2               ; GET POINTER
2822 030610  062702  000006                   ADD    #D.BGN1,R2          ;    TO FIRST BEGIN/END SET
2823 030614  016103  000004                   MOV    D.BEC(R1),R3        ; GET COUNT OF BEGIN/END SETS
2824 030620  026264  000006  000054  5$:      CMP    6(R2),P.UNSZ+RE.RP+2(R4); CHECK
2825 030626  103405                           BLO    6$                  ;    LAST
2826 030630  101014                           BHI    22$                 ;       LBN
2827 030632  026264  000004  000052           CMP    4(R2),P.UNSZ+RE.RP(R4) ;    IN
2828 030640  103010                           BHIS   22$                 ;          BEGIN/END SET
2829 030642  062702  000010          6$:      ADD    #8.,R2              ; GET NEXT
2830 030646  005303                           DEC    R3                  ;    BEGIN/END
2831 030650  003363                           BGT    5$                  ;       SET
2832 030652  004737  026006          10$:     CALL   RTNME               ; RETURN M.E. & CLEAR RESPONSE RING
2833 030656                          20$:
2834 030656  000240                           NOP                        ; [DBG] INSERT HALT FOR DEBUG
2835 030660  000207                           RETURN                     ;
2836
2837 030662  062704  000052          22$:     ADD    #P.UNSZ+RE.RP,R4    ; BUILD STRING CONTAINING
2838 030666  162714  000001                   SUB    #1,(R4)             ;
2839 030672  005664  000002                   SBC    2(R4)              ;
2840 030676  004737  047552                   CALL   BLDSTR             ;       MAX LBN
2841 030702                                   ERRSF  5,,ERRO05          ;
     030702  104454                           TRAP   C$ERSF
     030704  000005                           .WORD  5
     030706  000000                           .WORD  0
     030710  017240                           .WORD  ERRO05
2842 030712                                   DOCLN
     030712  104444                           TRAP   C$DCLN             ;
2843
```

```
2844                                        ;++
2845                                        ;        ENDSUC - "SET UNIT CHARACTERISTICS" END MESSAGE RECEIVED
2846                                        ;
2847                                        ;--
2848
2849 030714                                 ENDSUC:
2851 030714  012700  023340                 MOV     #DBGSUC,R0              ; [DEBUG] PRINT RECEIVED MSG TYPE
2852 030720  004737  024132                 CALL    DBGRSP                 ; [DEBUG] & BUFFER CONTENTS
2854 030724  012737  010107  002314         MOV     #OPCSUC,MSCPOP         ; POINT TO OPCODE MESSAGE STRING
2855 030732  004737  047164                 CALL    GTDRVT                 ; GET DRIVE TABLE
2856 030736  103007                          BCC     1$                     ;   IF TABLE NOT FOUND, PRINT ERROR
2857 030740                                 ERRDF   35,,ERRO35             ; UNKNOWN DRIVE ERROR
     030740  104455                          TRAP    C$ERDF
     030742  000043                          .WORD   35
     030744  000000                          .WORD   0
     030746  020454                          .WORD   ERRO35
2858 030750  004737  027350                 CALL    DRPCNT                 ; DROP CONTROLLER FROM TEST
2859 030754  000411                          BR      20$                    ; EXIT
2860 030756  032761  100000  000002  1$:    BIT     #DT.AVL,D.UNIT(R1)     ; IF DRIVE DROPPED,
2861 030764  001003                          BNE     10$                    ;   IGNORE RESPONSE PACKET
2862 030766  004737  032332                 CALL    CHKSTA                 ; CHECK END MESSAGE STATUS
2863 030772  103402                          BCS     20$                    ;   IF ERROR, EXIT
2864 030774  004737  026006  10$:           CALL    RTNME                  ; RETURN M.E. & CLEAR RESPONSE RING
2865 031000                          20$:
2866 031000  000240                 NOP                             ; [DBG] INSERT HALT FOR DEBUG
2867 031002  000207                         RETURN                          ;
2868
2869                                        ;++
2870                                        ;        ENDACC - "ACCESS" END MESSAGE RECEIVED
2871                                        ;
2872                                        ;
2873                                        ;--
2874 031004                                 ENDACC:
2876 031004  012700  023404                 MOV     #DBGACC,R0             ; [DEBUG] PRINT RECEIVED MSG TYPE
2877 031010  004737  024132                 CALL    DBGRSP                ; [DEBUG] & BUFFER CONTENTS
2879 031014  012737  007572  002314         MOV     #OPCACC,MSCPOP        ; POINT TO OPCODE MESSAGE STRING
2880 031022  004737  047164                 CALL    GTDRVT                ; GET DRIVE TABLE
2881 031026  103007                          BCC     1$                    ;   IF TABLE NOT FOUND, PRINT ERROR
2882 031030                                 ERRDF   35,,ERRO35            ; UNKNOWN DRIVE ERROR
     031030  104455                          TRAP    C$ERDF
     031032  000043                          .WORD   35
     031034  000000                          .WORD   0
     031036  020454                          .WORD   ERRO35
2883 031040  004737  027350                 CALL    DRPCNT                ; DROP CONTROLLER FROM TEST
2884 031044  000423                          BR      20$                   ; EXIT
2885 031046  032761  100000  000002  1$:    BIT     #DT.AVL,D.UNIT(R1)    ; IF DRIVE DROPPED,
2886 031054  001015                          BNE     10$                   ;   IGNORE RESPONSE PACKET
2887 031056  004737  032332                 CALL    CHKSTA                ; CHECK END MESSAGE STATUS
2888 031062  103414                          BCS     20$                   ;   IF ERROR, EXIT
2889 031064  066461  000022  000076         ADD     P.BCNT+RE.RP(R4),D.XFLA(R1); GET RETURNED BYTECOUNT
2890 031072  005561  000100                 ADC     D.XFMA(R1)            ;
2891 031076  066461  000024  000100         ADD     P.BCNT+RE.RP+2(R4),D.XFMA(R1);
2892 031104  005561  000102                 ADC     D.XFHA(R1)            ;
2893 031110  004737  026006  10$:           CALL    RTNME                 ; RETURN M.E. & CLEAR RESPONSE RING
2894 031114                          20$:
```

```
2895 031114 000240               NOP                              ; [DBG] INSERT HALT FOR DEBUG
2896 031116 000207                        RETURN                  ;
2897
2898                             ;++
2899                             ;        ENDERS - "ERASE" END MESSAGE RECEIVED
2900                             ;
2901                             ;--
2902
2903 031120                      ENDERS:
2905 031120 012700 023440        MOV      #DBGERS,R0              ; [DEBUG] PRINT RECEIVED MSG TYPE
2906 031124 004737 024132        CALL     DBGRSP                 ; [DEBUG] & BUFFER CONTENTS
2908 031130 012737 007725 002314 MOV      #OPCERS,MSCPOP         ; POINT TO OPCODE MESSAGE STRING
2909 031136 004737 047164        CALL     GTDRVT                 ; GET DRIVE TABLE
2910 031142 103007               BCC      1$                     ;   IF TABLE NOT FOUND, PRINT ERROR
2911 031144                      ERRDF    35,,ERRO35             ; UNKNOWN DRIVE ERROR
     031144 104455               TRAP     C$ERDF
     031146 000043               .WORD    35
     031150 000000               .WORD    0
     031152 020454               .WORD    ERRO35
2912 031154 004737 027350        CALL     DRPCNT                 ; DROP CONTROLLER FROM TEST
2913 031160 000425               BR       20$                    ; EXIT
2914 031162 032761 100000 000002 1$:   BIT  #DT.AVL,D.UNIT(R1)   ; IF DRIVE DROPPED,
2915 031170 001017               BNE      10$                    ;   IGNORE RESPONSE PACKET
2916 031172 005361 000056        DEC      D.WCNT(R1)             ; DECREMENT OUTSTANDING WRITES
2917 031176 004737 032332        CALL     CHKSTA                 ; CHECK END MESSAGE STATUS
2918 031202 103414               BCS      20$                    ;   IF ERROR, EXIT
2919 031204 066461 000022 000062 ADD      P.BCNT+RE.RP(R4),D.XFLW(R1); GET RETURNED BYTECOUNT
2920 031212 005561 000064        ADC      D.XFMW(R1)             ;
2921 031216 066461 000024 000064 ADD      P.BCNT+RE.RP+2(R4),D.XFMW(R1);
2922 031224 005561 000066        ADC      D.XFHW(R1)             ;
2923 031230 004737 026006        10$:  CALL  RTNME               ; RETURN M.E. & CLEAR RESPONSE RING
2924 031234                      20$:
2925 031234 000240               NOP                              ; [DBG] INSERT HALT FOR DEBUG
2926 031236 000207                        RETURN                  ;
2927
2928                             ;++
2929                             ;        ENDMRD - "MAINTENANCE READ" END MESSAGE RECEIVED
2930                             ;
2931                             ;        COMPARE DATA READ WITH CURRENT DATA PATTERN
2932                             ;
2933                             ;--
2934 031240                      ENDMRD:
2936 031240 012700 023473        MOV      #DBGMRD,R0             ; [DEBUG] PRINT RECEIVED MSG TYPE
2937 031244 004737 024132        CALL     DBGRSP                 ; [DEBUG] & BUFFER CONTENTS
2939 031250 012737 010152 002314 MOV      #OPCMRD,MSCPOP         ; POINT TO OPCODE MESSAGE STRING
2940 031256 012765 000001 000152 MOV      #1,C.CRED(R5)          ; SET CREDITS TO 1
2941 031264 005001               CLR      R1                     ; TABLE POINTER SANITY CHECK ON CHKSTA
2942 031266 004737 032332        CALL     CHKSTA                 ; CHECK END MESSAGE STATUS
2943 031272 103477               BCS      40$                    ;   IF ERROR, EXIT
2944 031274 066465 000022 000052 ADD      P.BCNT+RE.RP(R4),C.XFLR(R5); GET RETURNED BYTECOUNT
2945 031302 005565 000054        ADC      C.XFMR(R5)             ;
2946 031306 066465 000024 000054 ADD      P.BCNT+RE.RP+2(R4),C.XFMR(R5);
2947 031314 005565 000056        ADC      C.XFHR(R5)             ;
2948
2949 031320                      PUSH     <R0,R1,R2,R3,R4>       ; SAVE REGISTERS
```

```
      031320  010046              MOV     R0,-(SP)              ; PUSH R0 ON STACK
      031322  010146              MOV     R1,-(SP)              ; PUSH R1 ON STACK
      031324  010246              MOV     R2,-(SP)              ; PUSH R2 ON STACK
      031326  010346              MOV     R3,-(SP)              ; PUSH R3 ON STACK
      031330  010446              MOV     R4,-(SP)              ; PUSH R4 ON STACK
2950  031332  016504  000104      MOV     C.RHDR(R5),R4         ; GET
2951  031336  016401  000006      MOV     P.CRF+RE.RP(R4),R1    ;   POINTER TO
2952  031342  004737  025550      CALL    GETME                ;     MESSAGE ENVELOPE
2953  031346  016437  000020 002242  MOV  ME.OFF(R4),TSTOFF    ; GET POINTER TO DATA BUFFER
2954  031354  013703  002242      MOV     TSTOFF,R3            ; GET OFFSET TO DATA BUFFER
2955  031360  012701  002364      MOV     #OBUFF,R1            ; GET POINTER TO LOW MEMORY BUFFER
2956  031364  004737  025154      CALL    MMTOBF               ; COPY INPUT BUFFER TO LOW MEMORY
2957  031370  016502  000040      MOV     C.TPAT(R5),R2        ; GET DATA PATTERN INDEX
2958  031374  006302              ASL     R2                   ;
2959  031376  016400  000042      MOV     P.BCNT+ME.CP(R4),R0  ; GET
2960  031402  000241              CLC                          ;   WORD
2961  031404  006000              ROR     R0                   ;     COUNT
2962  031406  016403  000066      MOV     P.RGOFF+ME.CP(R4),R3 ; GET MEMORY OFFSET (WORD)
2963
2964  031412  022162  007036  15$:  CMP   (R1)+,MNTPAT(R2)     ; CHECK BUFFER CONTENTS
2965  031416  001007              BNE     30$                  ; IF NOT SAME, GO TO ERROR ROUTINE
2966  031420  005203              INC     R3                   ; CHECK
2967  031422  005300              DEC     R0                   ;   NEXT
2968  031424  003372              BGT     15$                  ;     WORD
2969  031426  004737  026006      CALL    RTNME                ; RETURN M.E. & CLEAR RESPONSE RING
2970  031432  000241              CLC                          ; CLEAR CARRY TO INDICATE SUCCESS
2971  031434  000416              BR      40$                  ;
2972
2973  031436  010337  004406  30$:  MOV   R3,BCLO              ;   OF BAD DATA
2974  031442  016237  007036 004364  MOV  MNTPAT(R2),PCGOOD    ; POINT TO
2975  031450  016137  177776 004366  MOV  -2(R1),PCBAD         ;   FAILING DATA
2976  031456                      ERRDF   39.,ERRO39           ; PRINT DATA COMPARE ERROR
      031456  104455              TRAP    C$ERDF
      031460  000047              .WORD   39
      031462  000000              .WORD   0
      031464  020714              .WORD   ERRO39
2977  031466  004737  027350      CALL    DRPCNT               ; DROP CONTROLLER
2978
2979  031472              40$:    POP     <R4,R3,R2,R1,R0>     ; RESTORE SAVED REGISTERS
      031472  012604              MOV     (SP)+,R4             ; POP STACK INTO R4
      031474  012603              MOV     (SP)+,R3             ; POP STACK INTO R3
      031476  012602              MOV     (SP)+,R2             ; POP STACK INTO R2
      031500  012601              MOV     (SP)+,R1             ; POP STACK INTO R1
      031502  012600              MOV     (SP)+,R0             ; POP STACK INTO R0
2980  031504  000240              NOP                          ; [DBG] INSERT HALT FOR DEBUG
2981  031506  000207              RETURN                       ;
2982
2983
2984                      ;++
2985                      ;       ENDMWR - "MAINTENANCE WRITE" END MESSAGE RECEIVED
2986                      ;
2987                      ;--
2988  031510              ENDMWR:
2990  031510  012700  023526      MOV     #DBGMWR,R0           ; [DEBUG] PRINT RECEIVED MSG TYPE
2991  031514  004737  024132      CALL    DBGRSP               ; [DEBUG] & BUFFER CONTENTS
```

```
2993 031520  012737 010175 002314        MOV    #OPCMWR,MSCPOP              ; POINT TO OPCODE MESSAGE STRING
2994 031526  012765 000001 000152        MOV    #1,C.CRED(R5)              ; SET CREDITS TO 1
2995 031534  005001                       CLR    R1                         ; TABLE POINTER SANITY CHECK ON CHKSTA
2996 031536  004737 032332               CALL   CHKSTA                     ; CHECK END MESSAGE STATUS
2997 031542  103414                       BCS    10$                        ;     IF ERROR, EXIT
2998 031544  066465 000022 000044        ADD    P.BCNT+RE,RP(R4),C.XFLW(R5); GET RETURNED BYTECOUNT
2999 031552  005565 000046               ADC    C.XFMW(R5)                 ;
3000 031556  066465 000024 000046        ADD    P.BCNT+RE,RP+2(R4),C.XFMW(R5);
3001 031564  005565 000050               ADC    C.XFHW(R5)                 ;
3002 031570  004737 026006               CALL   RTNME                      ; RETURN M.E. & CLEAR RESPONSE RING
3003 031574                      10$:
3004 031574  000240                       NOP                              ; [DBG] INSERT HALT FOR DEBUG
3005 031576  000207                       RETURN                           ;
3006
3007                             ;++
3008                             ;      ENDCMP - "COMPARE HOST DATA" END MESSAGE RECEIVED
3009                             ;
3010                             ;--
3011
3012 031600                      ENDCMP:
3014 031600  012700 023562               MOV    #DBGCMP,R0                 ; [DEBUG] PRINT RECEIVED MSG TYPE
3015 031604  004737 024132               CALL   DBGRSP                     ; [DEBUG] & BUFFER CONTENTS
3017 031610  012737 007651 002314        MOV    #OPCCMP,MSCPOP             ; POINT TO OPCODE MESSAGE STRING
3018 031616  004737 047164               CALL   GTDRVT                     ; GET DRIVE TABLE
3019 031622  103007                       BCC    1$                         ;    IF TABLE NOT FOUND, PRINT ERROR
3020 031624                               ERRDF  35,,ERRO35                 ; UNKNOWN DRIVE ERROR
     031624  104455                       TRAP   C$ERDF
     031626  000043                       .WORD  35
     031630  000000                       .WORD  0
     031632  020454                       .WORD  ERRO35
3021 031634  004737 027350               CALL   DRPCNT                     ; DROP CONTROLLER FROM TEST
3022 031640  000425                       BR     20$                        ; EXIT
3023 031642  032761 100000 000002  1$:   BIT    #DT.AVL,D.UNIT(R1)         ; IF DRIVE DROPPED,
3024 031650  001017                       BNE    10$                        ;    IGNORE RESPONSE PACKET
3025 031652  005361 000060               DEC    D.CCNT(R1)                 ; DECREMENT OUTSTANDING CMP COUNT
3026 031656  004737 032332               CALL   CHKSTA                     ; CHECK END MESSAGE STATUS
3027 031662  103414                       BCS    20$                        ;    IF ERROR, EXIT
3028 031664  066461 000022 000104        ADD    P.BCNT+RE,RP(R4),D.XFLC(R1); GET RETURNED BYTECOUNT
3029 031672  005561 000106               ADC    D.XFMC(R1)                 ;
3030 031676  066461 000024 000106        ADD    P.BCNT+RE,RP+2(R4),D.XFMC(R1);
3031 031704  005561 000110               ADC    D.XFHC(R1)                 ;
3032 031710  004737 026006        10$:   CALL   RTNME                      ; RETURN M.E. & CLEAR RESPONSE RING
3035 031714                      20$:
3034 031714  000240                       NOP                              ; [DBG] INSERT HALT FOR DEBUG
3035 031716  000207                       RETURN                           ;
3036
3037                             ;++
3038                             ;      ENDRD - "READ" END MESSAGE RECEIVED
3039                             ;
3040                             ;--
3041
3042 031720                      ENDRD:
3044 031720  012700 023616               MOV    #DBGRD,R0                  ; [DEBUG] PRINT RECEIVED MSG TYPE
3045 031724  004737 024132               CALL   DBGRSP                     ; [DEBUG] & BUFFER CONTENTS
3047 031730  012737 010025 002314        MOV    #OPCRD,MSCPOP              ; POINT TO OPCODE MESSAGE STRING
```

```
3048 031736  004737 047164           CALL    GTDRVT                  ; GET DRIVE TABLE
3049 031742  103007                   BCC     1$                      ;   IF TABLE NOT FOUND, PRINT ERROR
3050 031744                           ERRDF   35,,ERRO35              ; UNKNOWN DRIVE ERROR
     031744  104455                   TRAP    C$ERDF
     031746  000043                   .WORD   35
     031750  000000                   .WORD   0
     031752  020454                   .WORD   ERRO35
3051 031754  004737 027350           CALL    DRPCNT                  ; DROP CONTROLLER FROM TEST
3052 031760  000500                   BR      20$                     ; EXIT
3053 031762  032761 100000 000002 1$: BIT     #DT.AVL,D.UNIT(R1)      ; IF DRIVE DROPPED,
3054 031770  001072                   BNE     11$                     ;   IGNORE RESPONSE PACKET
3055 031772  004737 032332           CALL    CHKSTA                  ; CHECK END MESSAGE STATUS
3056 031776  103471                   BCS     20$                     ; IF ERROR, EXIT
3057 032000  066461 000022 000070     ADD     P.BCNT+RE.RP(R4),D.XFLR(R1); GET RETURNED BYTECOUNT
3058 032006  005561 000072            ADC     D.XFMR(R1)              ;
3059 032012  066461 000024 000072     ADD     P.BCNT+RE.RP+2(R4),D.XFMR(R1);
3060 032020  005561 000074            ADC     D.XFHR(R1)              ;
3061
3062 032024  032765 002000 000014     BIT     #CT.DET,C.FLG(R5)       ; IF DETERMINISTIC PHASE,
3063 032032  001051                   BNE     11$                     ;   DON'T DO PATTERN CHECK
3064 032034  032737 000004 002154     BIT     #SM.PCK,SO.BIT+SFPTBL   ; IF PATTERN CHECK NOT SELECTED,
3065 032042  001445                   BEQ     11$                     ;   DON'T DO PATTERN CHECKS
3066 032044                           PUSH    <R1>                    ; SAVE REGISTER
     032044  010146                   MOV     R1,-(SP)                ; PUSH R1 ON STACK
3067 032046  012701 000017            MOV     #15.,R1                 ; DO
3068 032052  005002                   CLR     R2                      ;   DATA
3069 032054  004737 026250           CALL    RANDOM                  ;     PATTERN
3070 032060  005701                   TST     R1                      ;       VERIFICATION?
3071 032062  001034                   BNE     10$                     ;
3072 032064                           POP     <R1>                    ; RESTORE REGISTER
     032064  012601                   MOV     (SP)+,R1                ; POP STACK INTO R1
3073 032066  066461 000022 000104     ADD     P.BCNT+RE.RP(R4),D.XFLC(R1); GET RETURNED BYTECOUNT
3074 032074  005561 000106            ADC     D.XFMC(R1)              ;
3075 032100  066461 000024 000106     ADD     P.BCNT+RE.RP+2(R4),D.XFMC(R1);
3076 032106  005561 000110            ADC     D.XFHC(R1)              ;
3077 032112  004737 046100           CALL    PATCHK                  ; DO PATTERN CHECK
3078 032116  103017                   BCC     11$                     ; CARRY CLEAR ON SUCCESS
3079 032120  005261 000116            INC     D.CERR(R1)              ; INCREMENT COMPARE ERROR COUNT
3080 032124  005261 000112            INC     D.HERR(R1)              ; INCREMENT HARD ERROR COUNT
3081 032130  005737 002144            TST     SO.EL+SFPTBL            ; IF HARD ERROR LIMIT ZERO,
3082 032134  001412                   BEQ     20$                     ;   DON'T DROP DRIVE
3083 032136  026137 000112 002144     CMP     D.HERR(R1),SO.EL+SFPTBL ; EXCEEDED
3084 032144  103406                   BLO     20$                     ;   HARD ERROR LIMIT?
3085 032146  004737 027424           CALL    DRPUNT                  ; DROP DRIVE
3086 032152  000403                   BR      20$                     ;
3087 032154                           POP     <R1>                    ; RESTORE SAVED REGISTER
     032154  012601                   MOV     (SP)+,R1                ; POP STACK INTO R1
3088 032156  004737 026006       11$: CALL    RTNME                   ; RETURN M.E. & CLEAR RESPONSE RING
3089 032162                       20$:
3090 032162  000240               NOP             ; [DBG] INSERT HALT FOR DEBUG
3091 032164  000207                   RETURN                          ;
3092
3093                              ;++
3094                              ;       ENDWR - "WRITE" END MESSAGE RECEIVED
3095                              ;
```

```
3096                                    ;      IF DT.CMP SET & D.WCNT = 0, ISSUE COMPARE HOST DATA COMMAND USING
3097                                    ;      MESSAGE ENVELOPE FROM LAST WRITE COMMAND.
3098                                    ;--
3099
3100 032166                     ENDWR:
3102 032166  012700  023650       MOV    #DBGWR,R0              ; [DEBUG] PRINT RECEIVED MSG TYPE
3103 032172  004737  024132       CALL   DBGRSP                ; [DEBUG] & BUFFER CONTENTS
3105 032176  012737  010142  002314 MOV  #OPCWR,MSCPOP          ; POINT TO OPCODE MESSAGE STRING
3106 032204  004737  047164       CALL   GTDRVT                ; GET DRIVE TABLE
3107 032210  103007              BCC    1$                    ;    IF TABLE NOT FOUND, PRINT ERROR
3108 032212                       ERRDF  35,,ERRO35            ; UNKNOWN DRIVE ERROR
     032212  104455               TRAP   C$ERDF
     032214  000043               .WORD  35
     032216  000000               .WORD  0
     032220  020454               .WORD  ERRO35
3109 032222  004737  027350       CALL   DRPCNT                ; DROP CONTROLLER FROM TEST
3110 032226  000437              BR     20$                   ; EXIT
3111 032230  032761  100000  000002 1$: BIT #DT.AVL,D.UNIT(R1) ; IF DRIVE DROPPED,
3112 032236  001031              BNE    10$                   ;    IGNORE RESPONSE PACKET
3113 032240  005361  000056       DEC    D.WCNT(R1)            ; ANY OUTSTANDING WRITES?
3114 032244  004737  032332       CALL   CHKSTA                ; CHECK END MESSAGE STATUS
3115 032250  103426              BCS    20$                   ; IF ERROR, EXIT
3116 032252  066461  000022  000062 ADD  P.BCNT+RE.RP(R4),D.XFLW(R1); GET RETURNED BYTECOUNT
3117 032260  005561  000064       ADC    D.XFMW(R1)            ;
3118 032264  066461  000024  000064 ADD  P.BCNT+RE.RP+2(R4),D.XFMW(R1);
3119 032272  005561  000066       ADC    D.XFMW(R1)            ;
3120 032276  005761  000056       TST    D.WCNT(R1)            ; IF OUTSTANDING WRITES <> 0,
3121 032302  001007              BNE    10$                   ;    EXIT
3122 032304  032761  004000  000002 BIT  #DT.CMP,D.UNIT(R1)    ; DO COMPARE HOST DATA?
3123 032312  001403              BEQ    10$
3124 032314  004737  050360       CALL   CMPHD                 ; DO IT
3125 032320  000402              BR     20$                   ;
3126 032322  004737  026006   10$: CALL  RTNME                 ; RETURN M.E. & CLEAR RESPONSE RING
3127 032326               20$:
3128 032326  000240              NOP                          ; [DBG] INSERT HALT FOR DEBUG
3129 032330  000207              RETURN                       ;
3130
```

```
3132
3133
3134                                    ;++
3135                                    ;         CHKSTA - OBTAIN STATUS/EVENT CODE OF END MESSAGE, TRANSLATE IT TO
3136                                    ;         ASCII, AND PRINT AN ERROR MESSAGE IF AN ERROR STATUS IS RETURNED.
3137                                    ;
3138                                    ;         INPUTS:
3139                                    ;                 R1 - DRIVE TABLE POINTER
3140                                    ;                 R2 - STATUS/EVENT CODE
3141                                    ;                 R3 - ENDCODE
3142                                    ;                 R4 - POINTER TO RESPONSE PACKET
3143                                    ;                 R5 - CONTROLLER TABLE ADDRESS
3144                                    ;         OUTPUTS:
3145                                    ;                 STAMS0 - STATUS MESSAGE
3146                                    ;                 STAMS1 - SUBCODE MESSAGE
3147                                    ;                 CARRY CLEAR ON "SUCCESS"
3148                                    ;                 CARRY SET ON "ERROR"
3149                                    ;--
3150   032332                          CHKSTA: PUSH    <R2>                 ; SAVE REGISTERS
       032332   010246                         MOV     R2,-(SP)             ; PUSH R2 ON STACK
3151   032334   042702   177740                BIC     #^CST.MSK,R2         ; GET STATUS CODE
3152   032340   005000                         CLR     R0                   ; CALCULATE
3153   032342   026002   032372        1$:     CMP     STACOD(R0),R2        ;   DISPATCH
3154   032346   001002                         BNE     2$                   ;     TABLE
3155   032350   000170   032422                JMP     @STAADR(R0)          ;       OFFSET
3156   032354   005200                2$:     INC     R0                   ;
3157   032356   005200                         INC     R0                   ;
3158   032360   020027   000026                CMP     R0,#STASIZ           ;
3159   032364   003766                         BLE     1$                   ;
3160   032366   000137   033114                JMP     STAUNK               ; STATUS CODE NOT IN LIST
3161
3162   032372   000000                STACOD: .WORD   0                    ; 00 - SUCCESS
3163   032374   000001                         .WORD   1                    ; 01 - INVALID COMMAND
3164   032376   000002                         .WORD   2                    ; 02 - COMMAND ABORTED
3165   032400   000003                         .WORD   3                    ; 03 - UNIT OFFLINE
3166   032402   000004                         .WORD   4                    ; 04 - UNIT AVAILABLE
3167   032404   000005                         .WORD   5                    ; 05 - MEDIA FORMAT ERROR
3168   032406   000006                         .WORD   6                    ; 06 - WRITE PROTECTED
3169   032410   000007                         .WORD   7                    ; 07 - COMPARE ERROR
3170   032412   000010                         .WORD   8.                   ; 08 - DATA ERROR
3171   032414   000011                         .WORD   9.                   ; 09 - HOST BUFFER ACCESS ERROR
3172   032416   000012                         .WORD   10.                  ; 10 - CONTROLLER ERROR
3173   032420   000013                         .WORD   11.                  ; 11 - DRIVE ERROR
3174
3175            000026                STASIZ  =       <.-STACOD-2>         ; LEGAL NUMBERS ARE LOWER THAN THIS
3176
3177   032422   032452                STAADR: .WORD   STASUC               ; 00 - SUCCESS
3178   032424   033024                         .WORD   STAINV               ; 01 - INVALID COMMAND
3179   032426   033220                         .WORD   STAABO               ; 02 - COMMAND ABORTED
3180   032430   033250                         .WORD   STAOFL               ; 03 - UNIT OFFLINE
3181   032432   033700                         .WORD   STAAVA               ; 04 - UNIT AVAILABLE
3182   032434   034000                         .WORD   STAMFE               ; 05 - MEDIA FORMAT ERROR
3183   032436   034320                         .WORD   STAWPR               ; 06 - WRITE PROTECTED
3184   032440   034562                         .WORD   STACMP               ; 07 - COMPARE ERROR
3185   032442   034702                         .WORD   STADAT               ; 08 - DATA ERROR
```

```
3186 032444 035630                              .WORD    STAHST              ; 09 - HOST BUFFER ACCESS ERROR
3187 032446 036220                              .WORD    STACNT              ; 10 - CONTROLLER ERROR
3188 032450 037004                              .WORD    STADRV              ; 11 - DRIVE ERROR
3189
3190                                    ;++
3191                                    ;           STASUC - "SUCCESS" STATUS RETURNED
3192                                    ;
3193                                    ;--
3194
3195 032452                            STASUC: POP      <R2>                 ; RESTORE REGISTERS
     032452 012602                             MOV      (SP)+,R2            ; POP STACK INTO R2
3196 032454 032737 020000 002204                BIT      #IERL,IFLAGS        ; IF NOT ERROR LOG,
3197 032462 001460                              BEQ      99$                 ;    DON'T NEED SUCCESS MESSAGE
3198 032464 022702 000040                        CMP      #SC.SPI,R2          ; SPIN-DOWN IGNORED?
3199 032470 001004                              BNE      2$                  ;
3200 032472 012737 032645 002320                MOV      #MSSUC1,STAMS1      ;
3201 032500 000446                              BR       20$                 ;
3202 032502 022702 000100         2$:           CMP      #SC.STC,R2          ; STILL CONNECTED?
3203 032506 001004                              BNE      3$                  ;
3204 032510 012737 032667 002320                MOV      #MSSUC2,STAMS1      ;
3205 032516 000437                              BR       20$                 ;
3206 032520 022702 000200         3$:           CMP      #SC.DUP,R2          ; DUPLICATE UNIT NUMBER?
3207 032524 001004                              BNE      4$                  ;
3208 032526 012737 032707 002320                MOV      #MSSUC3,STAMS1      ;
3209 032534 000430                              BR       20$                 ;
3210 032536 022702 000400         4$:           CMP      #SC.AOL,R2          ; ALREADY ONLINE?
3211 032542 001004                              BNE      5$                  ;
3212 032544 012737 032735 002320                MOV      #MSSUC4,STAMS1      ;
3213 032552 000421                              BR       20$                 ;
3214 032554 022702 001000         5$:           CMP      #SC.SOL,R2          ; STILL ONLINE?
3215 032560 001004                              BNE      6$                  ;
3216 032562 012737 032754 002320                MOV      #MSSUC5,STAMS1      ;
3217 032570 000412                              BR       20$                 ;
3218 032572 022702 010000         6$:           CMP      #SC.ROV,R2          ; READ ONLY (VOLUME FORMAT)?
3219 032576 001004                              BNE      7$                  ;
3220 032600 012737 032771 002320                MOV      #MSSUC6,STAMS1      ;
3221 032606 000403                              BR       20$                 ;
3222 032610 012737 010221 002320  7$:           MOV      #BLANK,STAMS1       ; SUBCODE NOT IN LIST
3223 032616 012737 032632 002316  20$:          MOV      #MSSUC0,STAMS0      ; INDICATE SUCCESS
3224 032624 000241                99$:          CLC                          ;
3225 032626 000240                              NOP                  ; [DBG] INSERT HALT FOR DEBUG
3226 032630 000207                              RETURN                       ;
3227
3228 032632  123  125  103       MSSUC0: .ASCIZ  /SUCCESS - /
3229 032645  123  120  111       MSSUC1: .ASCIZ  /SPIN-DOWN IGNORED/
3230 032667  123  124  111       MSSUC2: .ASCIZ  /STILL CONNECTED/
3231 032707  104  125  120       MSSUC3: .ASCIZ  /DUPLICATE UNIT NUMBER/
3232 032735  101  114  122       MSSUC4: .ASCIZ  /ALREADY ONLINE/
3233 032754  123  124  111       MSSUC5: .ASCIZ  /STILL ONLINE/
3234 032771  122  105  101       MSSUC6: .ASCIZ  /READ ONLY (VOLUME FORMAT)/
3235                                      .EVEN
3236
3237                                    ;++
3238                                    ;           STAINV - "INVALID COMMAND" STATUS RETURNED
3239                                    ;--
```

```
3240
3241 033024                          STAINV: POP     <R2>                        ; RESTORE REGISTERS
     033024   012602                          MOV     (SP)+,R2                    ; POP STACK INTO R2
3242 033026   012737   010221 002320          MOV     #BLANK,STAMS1               ; NO SUBCODES AVAILABLE
3243 033034   012737   033074 002316          MOV     #MSINVO,STAMSO
3244 033042   032737   020000 002204          BIT     #IERL,IFLAGS                ; IF ERROR LOG,
3245 033050   001006                          BNE     99$                         ;   DON'T PRINT ERROR
3246 033052                                   ERRSF   34,,ERRO34                  ; REPORT ERROR STATUS
     033052   104454                           TRAP    C$ERSF
     033054   000042                           .WORD   34
     033056   000000                           .WORD   0
     033060   020402                           .WORD   ERRO34
3247 033062   004737   027350                 CALL    DRPCNT                      ; DROP CONTROLLER FROM TEST
3248 033066   000261                  99$:     SEC                                ; SET CARRY FOR ERROR
3249 033070   000240                  NOP                      ; [DBG] INSERT HALT FOR DEBUG
3250 033072   000207                          RETURN                             ;
3251
3252 033074   111    116    126      MSINVO: .ASCIZ  /INVALID COMMAND/
3253                                          .EVEN
3254
3255                                  ;++
3256                                  ;      STAUNK - UNKNOWN STATUS RETURNED
3257                                  ;
3258                                  ;--
3259 033114                          STAUNK: POP     <R2>                        ; RESTORE REGISTERS
     033114   012602                          MOV     (SP)+,R2                    ; POP STACK INTO R2
3260 033116   012737   010221 002320          MOV     #BLANK,STAMS1               ; NO SUBCODES AVAILABLE
3261 033124   012737   033164 002316          MOV     #MSUNKO,STAMSO
3262 033132   032737   020000 002204          BIT     #IERL,IFLAGS                ; IF ERROR LOG,
3263 033140   001006                          BNE     99$                         ;   DON'T PRINT ERROR
3264 033142                                   ERRDF   33,,ERRO33                  ; REPORT ERROR STATUS
     033142   104455                           TRAP    C$ERDF
     033144   000041                           .WORD   33
     033146   000000                           .WORD   0
     033150   020346                           .WORD   ERRO33
3265 033152   004737   027350                 CALL    DRPCNT                      ; DROP CONTROLLER FROM TEST
3266 033156   000261                  99$:     SEC                                ; SET CARRY FOR ERROR
3267 033160   000240                  NOP                      ; [DBG] INSERT HALT FOR DEBUG
3268 033162   000207                          RETURN                             ;
3269
3270 033164   125    116    113      MSUNKO: .ASCIZ  /UNKNOWN COMMAND STATUS CODE/
3271                                          .EVEN
3272
3273
3274                                  ;++
3275                                  ;      STAABO - "COMMAND ABORTED" STATUS RETURNED
3276                                  ;      DON'T PRINT AN ERROR MESSAGE.
3277                                  ;      SET CARRY SO THE END MESSAGE WILL BE PUT IN THE BIT BUCKET.
3278                                  ;--
3279 033220                          STAABO: POP     <R2>                        ; RESTORE REGISTERS
     033220   012602                          MOV     (SP)+,R2                    ; POP STACK INTO R2
3280 033222   012737   010221 002320          MOV     #BLANK,STAMS1               ; SO ERROR LOGS
3281 033230   012737   010221 002316          MOV     #BLANK,STAMSO               ;   PRINT PROPERLY
3282 033236   004737   026006                 CALL    RTNME                       ; RETURN M.E. & CLEAR RESPONSE RING
3283 033242   000261                          SEC                                 ; INDICATE ERROR STATUS
```

```
3284 033244  000240              NOP                         ; [DBG] INSERT HALT FOR DEBUG
3285 033246  000207                      RETURN              ;
3286
3287                     ;++
3288                     ;       STAOFL - "UNIT OFFLINE" STATUS RETURNED
3289                     ;--
3290
3291 033250             STAOFL: POP     <R2>                ; RESTORE REGISTERS
     033250  012602              MOV     (SP)+,R2            ; POP STACK INTO R2
3292 033252  042702  000037      BIC     #ST.MSK,R2          ; GET SUBCODE
3293 033256  022702  000040      CMP     #SC.NVL,R2          ; NO VOLUME MOUNTED OR DRIVE DISABLED?
3294 033262  001004              BNE     2$                  ;
3295 033264  012737  033460  002320  MOV #MSOFL1,STAMS1      ;
3296 033272  000436              BR      20$                 ;
3297 033274  022702  000100  2$:  CMP    #SC.IOP,R2          ; UNIT INOPERATIVE?
3298 033300  001004              BNE     3$                  ;
3299 033302  012737  033524  002320  MOV #MSOFL2,STAMS1      ;
3300 033310  000427              BR      20$                 ;
3301 033312  022702  000400  3$:  CMP    #SC.DIS,R2          ; UNIT DISABLED BY FIELD SERVICE?
3302 033316  001004              BNE     4$                  ;
3303 033320  012737  033545  002320  MOV #MSOFL3,STAMS1      ;
3304 033326  000420              BR      20$                 ;
3305 033330  022702  000200  4$:  CMP    #SC.DUP,R2          ; DUPLICATE UNIT NUMBER?
3306 033334  001004              BNE     5$                  ;
3307 033336  012737  033604  002320  MOV #MSOFL4,STAMS1      ;
3308 033344  000411              BR      20$                 ;
3309 033346  005702          5$:  TST    R2                  ; UNIT UNKNOWN OR ONLINE OTHER CTRLR
3310 033350  001004              BNE     6$                  ;
3311 033352  012737  033632  002320  MOV #MSOFL5,STAMS1      ;
3312 033360  000403              BR      20$                 ;
3313 033362  012737  010221  002320  6$: MOV #BLANK,STAMS1   ; SUBCODE NOT IN LIST
3314 033370  012737  033440  002316  20$: MOV #MSOFL0,STAMS0 ;
3315 033376  032737  020000  002204  BIT  #IERL,IFLAGS       ; IF ERROR LOG,
3316 033404  001012              BNE     99$                 ;   DON'T PRINT ERROR
3317 033406  005261  000112      INC     D.HERR(R1)          ; INCREMENT HARD ERROR COUNT
3318 033412                      ERRDF   41,,ERRO41          ; REPORT ERROR STATUS
     033412  104455              TRAP    C$ERDF
     033414  000051              .WORD   41
     033416  000000              .WORD   0
     033420  021240              .WORD   ERRO41
3319 033422  004737  027424      CALL    DRPUNT              ; DROP DRIVE
3320 033426  004737  026006      CALL    RTNME               ; RETURN M.E. & CLEAR RESPONSE RING
3321 033432  000261          99$: SEC                        ; INDICATE ERROR STATUS
3322 033434  000240              NOP                         ; [DBG] INSERT HALT FOR DEBUG
3323 033436  000207              RETURN                      ;
3324
3325 033440  125  116  111  MSOFL0: .ASCIZ /UNIT OFFLINE - /
3326 033460  116  117  040  MSOFL1: .ASCIZ /NO VOLUME MOUNTED OR DRIVE DISABLED/
3327 033524  125  116  111  MSOFL2: .ASCIZ /UNIT INOPERATIVE/
3328 033545  125  116  111  MSOFL3: .ASCIZ /UNIT DISABLED BY FIELD SERVICE/
3329 033604  104  125  120  MSOFL4: .ASCIZ /DUPLICATE UNIT NUMBER/
3330 033632  125  116  111  MSOFL5: .ASCIZ /UNIT UNKNOWN OR ONLINE TO OTHER CTRLR/
3331                              .EVEN
3332
3333                     ;++
```

```
3334                                    ;       STAAVA - "UNIT AVAILABLE" STATUS RETURNED
3335                                    ;--
3336
3337 033700                    STAAVA: POP     <R2>                    ; RESTORE REGISTERS
     033700 012602                     MOV     (SP)+,R2                ; POP STACK INTO R2
3338 033702 012737 010221 002320       MOV     #BLANK,STAMS1           ; NO SUBCODES AVAILABLE
3339 033710 012737 033760 002316       MOV     #MSAVAO,STAMSO
3340 033716 032737 020000 0C2204       BIT     #IERL,IFLAGS            ; IF ERROR LOG,
3341 033724 001012                     BNE     99$                     ;   DON'T PRINT ERROR
3342 033726 005261 000112              INC     D.HERR(R1)              ; INCREMENT HARD ERROR COUNT
3343 033732                            ERRDF   41,,ERR041              ; REPORT ERROR STATUS
     033732 104455                     TRAP    C$ERDF
     033734 000051                     .WORD   41
     033736 000000                     .WORD   0
     033740 021240                     .WORD   ERR041
3344 033742 004737 027424              CALL    DRPUNT                  ; DROP DRIVE
3345 033746 004737 026006              CALL    RTNME                   ; RETURN M.E. & CLEAR RESPONSE RING
3346 033752 000261              99$:   SEC                             ; INDICATE ERROR STATUS
3347 033754 000240              NOP                            ; [DBG] INSERT HALT FOR DEBUG
3348 033756 000207                     RETURN                          ;
3349
3350 033760    125    116    111 MSAVAO: .ASCIZ  /UNIT AVAILABLE/
3351                                    .EVEN
3352
3353
3354                                    ;++
3355                                    ;       STAMFE - "MEDIA FORMAT ERROR" STATUS RETURNED
3356                                    ;--
3357 034000                    STAMFE: POP     <R2>                    ; RESTORE REGISTERS
     034000 012602                     MOV     (SP)+,R2                ; POP STACK INTO R2
3358 034002 042702 000037              BIC     #ST.MSK,R2              ; GET SUBCODE
3359 034006 022702 000240              CMP     #SC.576,R2              ; DISK FORMATTED WITH 576 BYTE SECTORS?
3360 034012 001004                     BNE     2$                      ;
3361 034014 012737 034202 002320       MOV     #MSMFE1,STAMS1          ;
3362 034022 000430                     BR      20$                     ;
3363 034024 022702 000300       2$:    CMP     #SC.FCT,R2              ; FCT CORRUPT?
3364 034030 001004                     BNE     3$                      ;
3365 034032 012737 034247 002320       MOV     #MSMFE2,STAMS1          ;
3366 034040 000421                     BR      20$                     ;
3367 034042 022702 000400       3$:    CMP     #SC.RCT,R2              ; RCT CORRUPT?
3368 034046 001004                     BNE     4$                      ;
3369 034050 012737 034263 002320       MOV     #MSMFE3,STAMS1          ;
3370 034056 000412                     BR      20$                     ;
3371 034060 022702 000440       4$:    CMP     #SC.RBN,R2              ; NO RBN AVAILABLE
3372 034064 001004                     BNE     5$                      ;
3373 034066 012737 034277 002320       MOV     #MSMFE4,STAMS1          ;
3374 034074 000403                     BR      20$                     ;
3375 034076 012737 010221 002320 5$:   MOV     #BLANK,STAMS1           ; SUBCODE NOT IN LIST
3376 034104 012737 034154 002316 20$:  MOV     #MSMFEO,STAMSO
3377 034112 032737 020000 002204       BIT     #IERL,IFLAGS            ; IF ERROR LOG,
3378 034120 001012                     BNE     99$                     ;   DON'T PRINT ERROR
3379 034122 005261 000112              INC     D.HERR(R1)              ; INCREMENT HARD ERROR COUNT
3380 034126                            ERRDF   41,,ERR041             ; REPORT ERROR STATUS
     034126 104455                     TRAP    C$ERDF
     034130 000051                     .WORD   41
```

```
              034132   000000                     .WORD   0
              034134   021240                     .WORD   ERR041
       3381  034136   004737   027424      CALL   DRPUNT                ; DROP DRIVE
       3382  034142   004737   026006      CALL   RTNME                 ; RETURN M.E. & CLEAR RESPONSE RING
       3383  034146   000261          99$:  SEC                         ; INDICATE ERROR STATUS
       3384  034150   000240                NOP                 ; [DBG] INSERT HALT FOR DEBUG
       3385  034152   000207                RETURN                      ;
       3386
       3387  034154    115      105      104  MSMFE0: .ASCIZ  /MEDIA FORMAT ERROR - /
       3388  034202    104      111      123  MSMFE1: .ASCIZ  /DISK FORMATTED WITH 576 BYTE SECTORS/
       3389  034247    106      103      124  MSMFE2: .ASCIZ  /FCT CORRUPT/
       3390  034263    122      103      124  MSMFE3: .ASCIZ  /RCT CORRUPT/
       3391  034277    116      117      040  MSMFE4: .ASCIZ  /NO RBN AVAILABLE/
       3392                                           .EVEN
       3393
       3394                                    ;++
       3395                                    ;       STAWPR - "WRITE PROTECTED" STATUS RETURNED
       3396                                    ;--
       3397
       3398  034320                       STAWPR: POP    <R2>                     ; RESTORE REGISTERS
              034320   012602                     MOV    (SP)+,R2                 ; POP STACK INTO R2
       3399  034322   042702   000037            BIC    #ST.MSK,R2               ; GET SUBCODE
       3400  034326   022702   020000            CMP    #SC.HWP,R2               ; HARDWARE WRITE PROTECT?
       3401  034332   001004                      BNE    2$                       ;
       3402  034334   012737   034457   002320    MOV    #MSWPR1,STAMS1           ;
       3403  034342   000412                       BR     20$                     ;
       3404  034344   022702   010000       2$:    CMP    #SC.SWP,R2               ; SOFTWARE WRITE PROTECT?
       3405  034350   001004                      BNE    3$                       ;
       3406  034352   012737   034520   002320    MOV    #MSWPR2,STAMS1           ;
       3407  034360   000403                       BR     20$                     ;
       3408  034362   012737   010221   002320 3$: MOV    #BLANK,STAMS1           ; SUBCODE NOT IN LIST
       3409  034370   012737   034434   002316 20$:MOV    #MSWPR0,STAMS0          ;
       3410  034376   032737   020000   002204    BIT    #IERL,IFLAGS            ; IF ERROR LOG,
       3411  034404   001010                      BNE    99$                     ;   DON'T PRINT ERROR
       3412  034406   005261   000114            INC    D.SERR(R1)               ; INCREMENT SOFT ERROR COUNT
       3413  034412                              ERRSOFT 41,,ERR041              ; REPORT ERROR STATUS
              034412   104457                     TRAP   C$ERSOFT
              034414   000051                     .WORD  41
              034416   000000                     .WORD  0
              034420   021240                     .WORD  ERR041
       3414  034422   004737   026006            CALL   RTNME                 ; RETURN M.E. & CLEAR RESPONSE RING
       3415  034426   000261          99$:  SEC                         ; INDICATE ERROR STATUS
       3416  034430   000240                NOP                 ; [DBG] INSERT HALT FOR DEBUG
       3417  034432   000207                RETURN                      ;
       3418
       3419  034434    127      122      111  MSWPR0: .ASCIZ  /WRITE PROTECTED - /
       3420  034457    125      116      111  MSWPR1: .ASCIZ  /UNIT IS HARDWARE WRITE PROTECTED/
       3421  034520    125      116      111  MSWPR2: .ASCIZ  /UNIT IS SOFTWARE WRITE PROTECTED/
       3422                                           .EVEN
       3423
       3424                                    ;++
       3425                                    ;       STACMP - "COMPARE ERROR" STATUS RETURNED
       3426                                    ;--
       3427
       3428  034562                       STACMP: POP    <R2>                     ; RESTORE REGISTERS
```

```
        034562  012602                    MOV      (SP)+,R2              ; POP STACK INTO R2
3429    034564  012737  010221  002320    MOV      #BLANK,STAMS1         ; NO SUBCODES AVAILABLE
3430    034572  012737  034664  002316    MOV      #MSCMPO,STAMSO
3431    034600  032737  020000  002204    BIT      #IERL,IFLAGS          ; IF ERROR LOG,
3432    034606  001023                    BNE      99$                   ;    DON'T PRINT ERROR
3433    034610  005261  000116            INC      D.CERR(R1)            ; INCREMENT COMPARE ERROR COUNT
3434    034614  005261  000112            INC      D.HERR(R1)            ; INCREMENT HARD ERROR COUNT
3435    034620                            ERRHRD   41,,ERRO41            ; REPORT ERROR STATUS
        034620  104456                    TRAP     C$ERHRD
        034622  000051                    .WORD    41
        034624  000000                    .WORD    0
        034626  021240                    .WORD    ERRO41
3436    034630  005737  002144            TST      SO.EL+SFPTBL          ; IF HARD ERROR LIMIT ZERO,
3437    034634  001406                    BEQ      98$                   ;    DON'T DROP DRIVE
3438    034636  026137  000112  002144    CMP      D.HERR(R1),SO.EL+SFPTBL ; EXCEEDED
3439    034644  103402                    BLO      98$                   ;       HARD ERROR LIMIT?
3440    034646  004737  027424            CALL     DRPUNT                ; DROP DRIVE
3441    034652  004737  026006    98$:    CALL     RTNME                 ; RETURN M.E. & CLEAR RESPONSE RING
3442    034656  000261    99$:    SEC                                    ; INDICATE ERROR STATUS
3443    034660  000240            NOP                           ; [DBG] INSERT HALT FOR DEBUG
3444    034662  000207                    RETURN                        ;
3445
3446    034664    103    117    115  MSCMPO:  .ASCIZ  /COMPARE ERROR/
3447                                          .EVEN
3448
3449                            ;++
3450                            ;       STADAT - "DATA ERROR" STATUS RETURNED
3451                            ;--
3452
3453    034702                    STADAT:  POP      <R2>                 ; RESTORE REGISTERS
        034702  012602                    MOV      (SP)+,R2             ; POP STACK INTO R2
3454    034704  042702  000037            BIC      #ST.MSK,R2           ; GET SUBCODE
3455    034710  005000                    CLR      RO                   ; INITIALIZE INDEX
3456    034712  026002  035126    4$:     CMP      NUMDAT(RO),R2        ; IS THIS THE SUBCODE?
3457    034716  001411                    BEQ      5$                   ;
3458    034720  062700  000002            ADD      #2,RO                ; GET NEXT SUBCODE
3459    034724  022700  000030            CMP      #SIZDAT,RO           ; ANY SUBCODES LEFT?
3460    034730  002370                    BGE      4$                   ;
3461    034732  012737  010221  002320    MOV      #BLANK,STAMS1        ; SUBCODE NOT IN LIST
3462    034740  000403                    BR       6$
3463    034742  016037  035160  002320  5$:  MOV   ADRDAT(RO),STAMS1    ; GET ADDRESS OF MESSAGE STRING
3464    034750  022702  000400    6$:     CMP      #SC.1EC,R2           ; DON'T
3465    034754  003403                    BLE      14$                  ;    COUNT ECC
3466    034756  022702  000200            CMP      #SC.CEC,R2           ;       CORRECTABLE DATA
3467    034762  001026                    BNE      20$                  ;          AS A DATA ERROR
3468    034764  005261  000120    14$:    INC      D.ECCC(R1)           ; COUNT ECC CORRECTED DATA
3469    034770  012737  035230  002316    MOV      #MSDATZ,STAMSO       ;
3470    034776  032737  020000  002204    BIT      #IERL,IFLAGS         ; IF ERROR LOG,
3471    035004  001045                    BNE      99$                  ;    DON'T PRINT ERROR
3472    035006  005261  000114            INC      D.SERR(R1)           ; INCREMENT SOFT ERROR COUNT
3473    035012  032737  000002  002154    BIT      #SM.PSE,SO.BIT+SFPTBL ; PRINT SOFT
3474    035020  001404                    BEQ      15$                  ;    ERRORS?
3475    035022                            ERRSOFT  41,,ERRO41           ; REPORT ERROR STATUS
        035022  104457                    TRAP     C$ERSOFT
        035024  000051                    .WORD    41
```

```
       035026  000000                        .WORD   0
       035030  021240                        .WORD   ERR041
3476   035032  000241           15$:  CLC                          ; NO ERROR OCCURRED
3477   035034  000240                 NOP                          ; [DBG] INSERT HALT FOR DEBUG
3478   035036  000207                 RETURN                       ;
3479
3480   035040  012737 035212 002316  20$:  MOV   #MSDAT0,STPMS0     ;
3481   035046  032737 020000 002204        BIT   #IERL,IFLAGS       ; IF ERROR LOG,
3482   035054  001021                       BNE   99$                ;   DON'T PRINT ERROR
3483   035056  005261 000112               INC   D.HERR(R1)         ; INCREMENT HARD ERROR COUNT
3484   035062                              ERRHRD 41,,ERR041        ; REPORT ERROR STATUS
       035062  104456                       TRAP  C$ERHRD
       035064  000051                       .WORD  41
       035066  000000                       .WORD  0
       035070  021240                       .WORD  ERR041
3485   035072  005737 002144               TST   SO.EL+SFPTBL       ; IF HARD ERROR LIMIT ZERO,
3486   035076  001406                       BEQ   98$                ;   DON'T DROP DRIVE
3487   035100  026137 000112 002144        CMP   D.HERR(R1),SO.EL+SFPTBL ; EXCEEDED
3488   035106  103402                       BLO   98$                ;   HARD ERROR LIMIT?
3489   035110  004737 027424               CALL  DRPUNT             ; DROP DRIVE
3490   035114  004737 026006        98$:  CALL  RTNME             ; RETURN M.E. & CLEAR RESPONSE RING
3491   035120  000261           99$:  SEC                          ; INDICATE ERROR STATUS
3492   035122  000240                 NOP                          ; [DBG] INSERT HALT FOR DEBUG
3493   035124  000207                 RETURN                       ;
3494
3495   035126  000000           NUMDAT: .WORD  SC.FOR              ; FORCED ERROR
3496   035130  000100                   .WORD  SC.HDR              ; INVALID HEADER
3497   035132  000140                   .WORD  SC.DSY              ; DATA SYNC NOT FOUND
3498   035134  000200                   .WORD  SC.CEC              ; CORRECTABLE ECC ERROR
3499   035136  000340                   .WORD  SC.UEC              ; UNCORRECTABLE ECC ERROR
3500   035140  000400                   .WORD  SC.1EC              ; 1 SYMBOL ECC ERROR
3501   035142  000440                   .WORD  SC.2EC              ; 2 SYMBOL ECC ERROR
3502   035144  000500                   .WORD  SC.3EC              ; 3 SYMBOL ECC ERROR
3503   035146  000540                   .WORD  SC.4EC              ; 4 SYMBOL ECC ERROR
3504   035150  000600                   .WORD  SC.5EC              ; 5 SYMBOL ECC ERROR
3505   035152  000640                   .WORD  SC.6EC              ; 6 SYMBOL ECC ERROR
3506   035154  000700                   .WORD  SC.7EC              ; 7 SYMBOL ECC ERROR
3507   035156  000740                   .WORD  SC.8EC              ; 8 SYMBOL ECC ERROR
3508
3509           000030           SIZDAT =  <.-NUMDAT-2>            ; LEGAL NUMBERS ARE LOWER THAN THIS
3510
3511   035160  035612           ADRDAT: .WORD  MSDAT0              ; FORCED ERROR
3512   035162  035260                   .WORD  MSDAT1              ; INVALID HEADER
3513   035164  035277                   .WORD  MSDAT2              ; DATA SYNC NOT FOUND
3514   035166  010221                   .WORD  BLANK               ; CORRECTABLE ECC ERROR
3515   035170  035323                   .WORD  MSDAT4              ; UNCORRECTABLE ECC ERROR
3516   035172  035353                   .WORD  MSDAT5              ; 1 SYMBOL ECC ERROR
3517   035174  035376                   .WORD  MSDAT6              ; 2 SYMBOL ECC ERROR
3518   035176  035422                   .WORD  MSDAT7              ; 3 SYMBOL ECC ERROR
3519   035200  035446                   .WORD  MSDAT8              ; 4 SYMBOL ECC ERROR
3520   035202  035472                   .WORD  MSDAT9              ; 5 SYMBOL ECC ERROR
3521   035204  035516                   .WORD  MSDATA              ; 6 SYMBOL ECC ERROR
3522   035206  035542                   .WORD  MSDATB              ; 7 SYMBOL ECC ERROR
3523   035210  035566                   .WORD  MSDATC              ; 8 SYMBOL ECC ERROR
3524
```

```
3525 035212    104    101    124   MSDAT0: .ASCIZ  /DATA ERROR - /
3526 035230    105    103    103   MSDATZ: .ASCIZ  /ECC CORRECTABLE DATA - /
3527 035260    111    116    126   MSDAT1: .ASCIZ  /INVALID HEADER/
3528 035277    104    101    124   MSDAT2: .ASCIZ  /DATA SYNC NOT FOUND/
3529 035323    125    116    103   MSDAT4: .ASCIZ  /UNCORRECTABLE ECC ERROR/
3530 035353    061    040    123   MSDAT5: .ASCIZ  /1 SYMBOL CORRECTED/
3531 035376    062    040    123   MSDAT6: .ASCIZ  /2 SYMBOLS CORRECTED/
3532 035422    063    040    123   MSDAT7: .ASCIZ  /3 SYMBOLS CORRECTED/
3533 035446    064    040    123   MSDAT8: .ASCIZ  /4 SYMBOLS CORRECTED/
3534 035472    065    040    123   MSDAT9: .ASCIZ  /5 SYMBOLS CORRECTED/
3535 035516    066    040    123   MSDATA: .ASCIZ  /6 SYMBOLS CORRECTED/
3536 035542    067    040    123   MSDATB: .ASCIZ  /7 SYMBOLS CORRECTED/
3537 035566    070    040    123   MSDATC: .ASCIZ  /8 SYMBOLS CORRECTED/
3538 035612    106    117    122   MSDATD: .ASCIZ  /FORCED ERROR/
3539                                       .EVEN
3540
3541
3542                               ;++
3543                               ;       STAHST - "HOST BUFFER ACCESS ERROR" STATUS RETURNED
3544                               ;--
3545 035630                        STAHST: POP     <R2>                  ; RESTORE REGISTERS
     035630    012602                      MOV     (SP)+,R2              ; POP STACK INTO R2
3546 035632    042702  000037              BIC     #ST.MSK,R2            ; GET SUBCODE
3547 035636    022702  000040              CMP     #SC.OTA,R2            ; ODD TRANSFER ADDRESS?
3548 035642    001004                      BNE     1$                    ;
3549 035644    012737  036046  002320      MOV     #MSHST1,STAMS1        ;
3550 035652    000437                      BR      20$                   ;
3551 035654    022702  000100      1$:     CMP     #SC.OBC,R2            ; ODD BYTE COUNT?
3552 035660    001004                      BNE     2$                    ;
3553 035662    012737  036073  002320      MOV     #MSHST2,STAMS1        ;
3554 035670    000430                      BR      20$                   ;
3555 035672    022702  000140      2$:     CMP     #SC.NXM,R2            ; NON-EXISTANT MEMORY?
3556 035676    001004                      BNE     3$                    ;
3557 035700    012737  036112  002320      MOV     #MSHST3,STAMS1        ;
3558 035706    000421                      BR      20$                   ;
3559 035710    022702  000200      3$:     CMP     #SC.HMP,R2            ; HOST MEMORY PARITY ERROR?
3560 035714    001004                      BNE     4$                    ;
3561 035716    012737  036136  002320      MOV     #MSHST4,STAMS1        ;
3562 035724    000412                      BR      20$                   ;
3563 035726    022702  000240      4$:     CMP     #SC.IPT,R2            ; INVALID PAGE TABLE ENTRY?
3564 035732    001004                      BNE     5$                    ;
3565 035734    012737  036167  002320      MOV     #MSHST5,STAMS1        ;
3566 035742    000403                      BR      20$                   ;
3567 035744    012737  010221  002320 5$:  MOV     #BLANK,STAMS1         ; SUBCODE NOT IN LIST
3568 035752    012737  036012  002316 20$: MOV     #MSHST0,STAMS0        ;
3569 035760    032737  020000  002204      BIT     #IERL,IFLAGS          ; IF ERROR LOG,
3570 035766    001006                      BNE     99$                   ;    DON'T PRINT ERROR
3571 035770                                ERRDF   40,,ERR040            ; REPORT ERROR STATUS
     035770    104455                      TRAP    C$ERDF
     035772    000050                      .WORD   40
     035774    000000                      .WORD   0
     035776    021016                      .WORD   ERR040
3572 036000    004737  027350      98$:    CALL    DRPCNT                ; DROP CONTROLLER FROM TEST
3573 036004    000261              99$:    SEC                           ; INDICATE ERROR STATUS
3574 036006    000240                      NOP            ; [DBG] INSERT HALT FOR DEBUG
```

```
3575 036010  000207                        RETURN                           ;
3576
3577 036012     110    117    123  MSHST0: .ASCIZ  /HOST BUFFER ACCESS ERROR - /
3578 036046     117    104    104  MSHST1: .ASCIZ  /ODD TRANSFER ADDRESS/
3579 036073     117    104    104  MSHST2: .ASCIZ  /ODD BYTE COUNT/
3580 036112     116    117    116  MSHST3: .ASCIZ  /NON-EXISTANT MEMORY/
3581 036136     110    117    123  MSHST4: .ASCIZ  /HOST MEMORY PARITY ERROR/
3582 036167     111    116    126  MSHST5: .ASCIZ  /INVALID PAGE TABLE ENTRY/
3583                                        .EVEN
3584
3585                                ;++
3586                                ;       STACNT - "CONTROLLER ERROR" STATUS RETURNED
3587                                ;--
3588
3589 036220                        STACNT: POP     <R2>                     ; RESTORE REGISTERS
     036220  012602                        MOV     (SP)+,R2                 ; POP STACK INTO R2
3590 036222  042702  000037                BIC     #ST.MSK,R2               ; GET SUBCODE
3591 036226  005000                        CLR     R0                       ; INITIALIZE INDEX
3592 036230  026002  036326        1$:     CMP     NUMCNT(R0),R2            ; IS THIS THE SUBCODE?
3593 036234  001411                        BEQ     10$
3594 036236  062700  000002                ADD     #2,R0                    ; GET NEXT SUBCODE
3595 036242  022700  000020                CMP     #SIZCNT,R0               ; ANY SUBCODES LEFT?
3596 036246  002370                        BGE     1$                       ;
3597 036250  012737  010221  002320        MOV     #BLANK,STAMS1            ; SUBCODE NOT IN LIST
3598 036256  000403                        BR      11$
3599 036260  016037  036350  002320 10$:   MOV     ADRCNT(R0),STAMS1       ; GET ADDRESS OF MESSAGE STRING
3600 036266  012737  036372  002316 11$:   MOV     #MSCNT0,STAMS0          ;
3601 036274  032737  020000  002204        BIT     #IERL,IFLAGS            ; IF ERROR LOG,
3602 036302  001006                        BNE     99$                      ;   DON'T PRINT ERROR
3603 036304                                ERRDF   40,,ERRO40               ; REPORT ERROR STATUS
     036304  104455                        TRAP    C$ERDF
     036306  000050                        .WORD   40
     036310  000000                        .WORD   0
     036312  021016                        .WORD   ERRO40
3604 036314  004737  027350        98$:    CALL    DRPCNT                   ; DROP CONTROLLER FROM TEST
3605 036320  000261                99$:    SEC                              ; INDICATE ERROR STATUS
3606 036322  000240                        NOP                     ; [DBG] INSERT HALT FOR DEBUG
3607 036324  000207                        RETURN                           ;
3608
3609 036326  000040                NUMCNT: .WORD   SC.SDS                   ; SERDES OVERRUN OR UNDERRUN
3610 036330  000100                        .WORD   SC.EDC                   ; EDC ERROR
3611 036332  000140                        .WORD   SC.IIC                   ; INCONSISTANT INTERNAL CONTROL
3612 036334  000200                        .WORD   SC.IEE                   ; INTERNAL EDC ERROR
3613 036336  000240                        .WORD   SC.LPI                   ; LESI ADAPTOR CARD PARITY ERROR INPUT
3614 036340  000300                        .WORD   SC.LPO                   ; LESI ADAPTOR CARD PARITY ERROR,OUTPUT
3615 036342  000340                        .WORD   SC.CIP                   ; "CABLE IN PLACE" NOT ASSERTED
3616 036344  000400                        .WORD   SC.COU                   ; CONTROLLER OVERRUN OR UNDERRUN
3617 036346  000440                        .WORD   SC.CME                   ; CONTROLLER MEMORY ERROR
3618
3619         000020                SIZCNT  =       <.-NUMCNT-2>            ; LEGAL NUMBERS ARE LOWER THAN THIS
3620
3621 036350  036416                ADRCNT: .WORD   MSCNT1                   ; SERDES OVERRUN OR UNDERRUN
3622 036352  036451                        .WORD   MSCNT2                   ; EDC ERROR
3623 036354  036463                        .WORD   MSCNT3                   ; INCONSISTANT INTERNAL CONTRGL
3624 036356  036521                        .WORD   MSCNT4                   ; INTERNAL EDC ERROR
```

```
3625 036360  036544                        .WORD   MSCNT5          ; LESI ADAPTOR CARD PARITY ERROR INPUT
3626 036362  036611                        .WORD   MSCNT6          ; LESI ADAPTOR CARD PARITY ERROR,OUTPUT
3627 036364  036657                        .WORD   MSCNT7          ; "CABLE IN PLACE" NOT ASSERTED
3628 036366  036715                        .WORD   MSCNT8          ; CONTROLLER OVERRUN OR UNDERRUN
3629 036370  036754                        .WORD   MSCNT9          ; CONTROLLER MEMORY ERROR
3630
3631 036372  103  117  116  MSCNT0: .ASCIZ /CONTROLLER ERROR - /
3632 036416  123  105  122  MSCNT1: .ASCIZ /SERDES OVERRUN OR UNDERRUN/
3633 036451  105  104  103  MSCNT2: .ASCIZ /EDC ERROR/
3634 036463  111  116  103  MSCNT3: .ASCIZ /INCONSISTANT INTERNAL CONTROL/
3635 036521  111  116  124  MSCNT4: .ASCIZ /INTERNAL EDC ERROR/
3636 036544  114  105  123  MSCNT5: .ASCIZ /LESI ADAPTOR CARD PARITY ERROR INPUT/
3637 036611  114  105  123  MSCNT6: .ASCIZ /LESI ADAPTOR CARD PARITY ERROR,OUTPUT/
3638 036657  042  103  101  MSCNT7: .ASCIZ /"CABLE IN PLACE" NOT ASSERTED/
3639 036715  103  117  116  MSCNT8: .ASCIZ /CONTROLLER OVERRUN OR UNDERRUN/
3640 036754  103  117  116  MSCNT9: .ASCIZ /CONTROLLER MEMORY ERROR/
3641                                        .EVEN
3642
3643
3644                        ;++
3645                        ;       STADRV - "DRIVE ERROR" STATUS RETURNED
3646                        ;--
3647 037004               STADRV: POP    <R2>             ; RESTORE REGISTERS
     037004  012602                MOV    (SP)+,R2         ; POP STACK INTO R2
3648 037006  042702  000037         BIC    #ST.MSK,R2       ; GET SUBCODE
3649 037012  005000                 CLR    R0               ; INITIALIZE INDEX
3650 037014  026002  037122  2$:    CMP    NUMDRV(R0),R2    ; IS THIS THE SUBCODE?
3651 037020  001411                 BEQ    10$              ;
3652 037022  062700  000002         ADD    #2,R0            ; GET NEXT SUBCODE
3653 037026  022700  000026         CMP    #SIZDRV,R0       ; ANY SUBCODES LEFT?
3654 037032  002370                 BGE    2$               ;
3655 037034  012737  010221  002320 MOV    #BLANK,STAMS1    ; SUBCODE NOT IN LIST
3656 037042  000403                 BR     11$              ;
3657 037044  016037  037152  002320 10$:   MOV    ADRDRV(R0),STAMS1  ; GET ADDRESS OF MESSAGE STRING
3658 037052  012737  037202  002316 11$:   MOV    #MSDRV0,STAMS0   ;
3659 037060  032737  020000  002204 BIT    #IERL,IFLAGS     ; IF ERROR LOG,
3660 037066  001012                 BNE    99$              ;   DON'T PRINT ERROR
3661 037070  005261  000112         INC    D.HERR(R1)       ; INCREMENT HARD ERROR COUNT
3662 037074                         ERRDF  41,.ERR041       ; REPORT ERROR STATUS
     037074  104455                TRAP   C$ERDF
     037076  000051                .WORD  41
     037100  000000                .WORD  0
     037102  021240                .WORD  ERR041
3663 037104  004737  027424         CALL   DRPUNT           ; DROP DRIVE
3664 037110  004737  026006  98$:   CALL   RTNME            ; RETURN M.E. & CLEAR RESPONSE RING
3665 037114  000261         99$:    SEC                     ; INDICATE ERROR STATUS
3666 037116  000240                 NOP             ; [DBG] INSERT HALT FOR DEBUG
3667 037120  000207                 RETURN                  ;
3668
3669 037122  000040         NUMDRV: .WORD  SC.CTO           ; DRIVE COMMAND TIME OUT
3670 037124  000100                 .WORD  SC.XME           ; CONTROLLER DETECTED XMISSION ERROR
3671 037126  000140                 .WORD  SC.POE           ; POSITIONER ERROR
3672 037130  000200                 .WORD  SC.LRW           ; LOST RESD/WRITE READY
3673 037132  000240                 .WORD  SC.DCD           ; DRIVE CLOCK DROPOUT
3674 037134  000300                 .WORD  SC.LRR           ; LOST RECEIVER READY
```

```
3675 037136  000340                         .WORD   SC.DDE          ; DRIVE DETECTED ERROR
3676 037140  000400                         .WORD   SC.PSE          ; PULSE OR STATE PARITY ERROR
3677 037142  000500                         .WORD   SC.PRO          ; CONTROLLER DETECTED PROTOCOL ERROR
3678 037144  000550                         .WORD   SC.DFI          ; DRIVE FAILED INITIALIZATION
3679 037146  000600                         .WORD   SC.DII          ; DRIVE IGNORED INITIALIZATION
3680 037150  000640                         .WORD   SC.RRC          ; RECEIVER READY COLLISION
3681
3682         000026             SIZDRV  =           <.-NUMDRV-2>    ; LEGAL NUMBERS ARE LOWER THAN THIS
3683
3684 037152  037221            ADRDRV: .WORD   MSDRV1          ; DRIVE COMMAND TIME OUT
3685 037154  037250                     .WORD   MSDRV2          ; CONTROLLER DETECTED XMISSION ERROR
3686 037156  037317                     .WORD   MSDRV3          ; POSITIONER ERROR
3687 037160  037340                     .WORD   MSDRV4          ; LOST RESD/WRITE READY
3688 037162  037366                     .WORD   MSDRV5          ; DRIVE CLOCK DROPOUT
3689 037164  037412                     .WORD   MSDRV6          ; LOST RECEIVER READY
3690 037166  037436                     .WORD   MSDRV7          ; DRIVE DETECTED ERROR
3691 037170  037463                     .WORD   MSDRV8          ; PULSE OR STATE PARITY ERROR
3692 037172  037517                     .WORD   MSDRV9          ; CONTROLLER DETECTED PROTOCOL ERROR
3693 037174  037562                     .WORD   MSDRVA          ; DRIVE FAILED INITIALIZATION
3694 037176  037616                     .WORD   MSDRVB          ; DRIVE IGNORED INITIALIZATION
3695 037200  037653                     .WORD   MSDRVC          ; RECEIVER READY COLLISION
3696
3697 037202  104  122  111  MSDRV0: .ASCIZ  /DRIVE ERROR - /
3698 037221  104  122  111  MSDRV1: .ASCIZ  /DRIVE COMMAND TIME OUT/
3699 037250  103  117  116  MSDRV2: .ASCIZ  /CONTROLLER DETECTED TRANSMISSION ERROR/
3700 037517  120  117  123  MSDRV3: .ASCIZ  /POSITIONER ERROR/
3701 037340  114  117  123  MSDRV4: .ASCIZ  \LOST READ/WRITE READY\
3702 037366  104  122  111  MSDRV5: .ASCIZ  /DRIVE CLOCK DROPOUT/
3703 037412  114  117  123  MSDRV6: .ASCIZ  /LOST RECEIVER READY/
3704 037436  104  122  111  MSDRV7: .ASCIZ  /DRIVE DETECTED ERROR/
3705 037463  120  125  114  MSDRV8: .ASCIZ  /PULSE OR STATE PARITY ERROR/
3706 037517  103  117  116  MSDRV9: .ASCIZ  /CONTROLLER DETECTED PROTOCOL ERROR/
3707 037562  104  122  111  MSDRVA: .ASCIZ  /DRIVE FAILED INITIALIZATION/
3708 037616  104  122  111  MSDRVB: .ASCIZ  /DRIVE IGNORED INITIALIZATION/
3709 037653  122  105  103  MSDRVC: .ASCIZ  /RECEIVER READY COLLISION/
3710                                        .EVEN
```

```
3712
3713
3714                              ;++              RSPATN - RESPOND TO ATTENTION MESSAGE.
3715                              ;                INPUTS:
3716                              ;                        R1 - DRIVE UNIT NUMBER
3717                              ;                        R2 - STATUS/EVENT CODE
3718                              ;                        R3 - ENDCODE
3719                              ;                        R4 - POINTER TO RESPONSE PACKET
3720                              ;                        R5 - CONTROLLER TABLE ADDRESS
3721                              ;                OUTPUTS:
3722                              ;                        RESPONSE PACKET CLEARED, REPONSE RING OWNED BY CONTROLLER,
3723                              ;                        CONTENTS OF REGISTERS R1,R2,R3,R4 UNDEFINED
3724                              ;                        R5 UNCHANGED
3725                              ;--
3726
3727 037704                      RSPATN:
3729 037704  012760  023005              MOV    #DBGATN,R0              ; [DEBUG] PRINT RECEIVED MSG TYPE
3730 037710  004737  024132              CALL   DBGRSP                  ; [DEBUG] & BUFFER CONTENTS
3732 037714  022703  000100              CMP    #100,R3                 ; 100 - AVAILABLE ATTENTION MESSAGE
3733 037720  001410                      BEQ    ATNAVA                  ;
3734 037722  022703  000101              CMP    #101,R3                 ; 101 - DUPLICATE UNIT NUMBER
3735 037726  001425                      BEQ    ATNDUP                  ;
3736 037730  022703  000102              CMP    #102,R3                 ; 102 - ACCESS PATH ATTENTION MESSAGE
3737 037734  001436                      BEQ    ATNACC                  ;
3738 037736  000137  027744              JMP    INVCMD                  ; INVALID END MESSAGE RECEIVED
3739
3740                              ;++              ATNAVA - "AVAILABLE" ATTENTION MESSAGE RECEIVED.
3741                              ;
3742                              ;                IF UNIT SPECIFIED IN THE ATTENTION MESSAGE IS TO BE TESTED,
3743                              ;                        PRINT A DEVICE FATAL ERROR AND DROP UNIT FROM TEST
3744                              ;                ELSE, IGNORE THE ATTENTION MESSAGE
3745                              ;--
3746
3747 037742  004737  047164      ATNAVA: CALL   GTDRVT                  ; GET DRIVE TABLE
3748 037746  103413                      BCS    30$                     ;   IF TABLE NOT FOUND, IGNORE MESSAGE
3749 037750  032761  001000  000002      BIT    #OT.ONL,D.UNIT(R1)      ; IF DRIVE SHOULDN'T BE ONLINE,
3750 037756  001407                      BEQ    30$                     ;   IGNORE MESSAGE
3751 037760                              EMROF  60,,ERR060              ; DRIVE UNDER TEST WENT AVAILABLE ERROR
     037760  104455                      TRAP   CUEMOF
     037762  000074                      .WORD  60
     037764  000000                      .WORD  0
     037766  022472                      .WORD  ERR060
3752 037770  004737  027424              CALL   DRPUNT                  ; DROP UNIT FROM TEST
3753 037774  000207                      RETURN                         ;
3754 037776                      30$:
3755 037776  000240                      NOP                            ; [DBG] INSERT HALT FOR DEBUG
3756 040000  000207                      RETURN                         ;
3757
3758                              ;++              ATNDUP - "DUPLICATE UNIT NUMBER" ATTENTION MESSAGE RECEIVED
3759                              ;
3760                              ;                IF UNIT SPECIFIED IN THE ATTENTION MESSAGE IS TO BE TESTED,
3761                              ;                        PRINT A DEVICE FATAL ERROR AND DROP UNIT FROM TEST
3762                              ;                ELSE, IGNORE THE ATTENTION MESSAGE
3763                              ;--
3764
```

```
3765
3766  040002  004737  047164    ATNCUP: CALL    GTDRVT              ; GET DRIVE TABLE
3767  040006  103407                    BCS     30$                ;   IF TABLE NOT FOUND, IGNORE MESSAGE
3768  040010                            ERRDF   61,,ERRO61         ; DRIVE UNDER TEST DUPLICATE UNIT #
      040010  104455                    TRAP    C$ERDF
      040012  000075                    .WORD   61
      040014  000000                    .WORD   0
      040016  022544                    .WORD   ERRO61
3769  040020  004737  027424            CALL    DRPUNT             ; DROP CONTROLLER FROM TEST
3770  040024  000207                    RETURN                     ;
3771  040026            30$:
3772  040026  000240            NOP                      ; [DBG] INSERT HALT FOR DEBUG
3773  040030  000207                    RETURN                     ;
3774
3775                            ;++
3776                            ;       ATNACC - "ACCESS PATH" ATTENTION MESSAGE RECEIVED
3777                            ;
3778                            ;       THIS ATTENTION MESSAGE IS ALWAYS IGNORED
3779                            ;--
3780
3781  040032            ATNACC:
3782  040032  000240            NOP                      ; [DBG] INSERT HALT FOR DEBUG
3783  040034  000207                    RETURN                     ;
3784
```

```
3786
3787
3788                                     ;++
3789                                     ;        RSPERL - RESPOND TO ERROR LOG MESSAGE.
3790                                     ;
3791                                     ;        INPUTS:
3792                                     ;                R1 - DRIVE UNIT NUMBER
3793                                     ;                R2 - STATUS/EVENT CODE
3794                                     ;                R3 - ENDCODE
3795                                     ;                R4 - POINTER TO RESPONSE PACKET
3796                                     ;                R5 - CONTROLLER TABLE ADDRESS
3797                                     ;        OUTPUTS:
3798                                     ;                RESPONSE PACKET CLEARED, REPONSE RING OWNED BY CONTROLLER.
3799                                     ;                CONTENTS OF REGISTERS R1,R2,R3,R4 UNDEFINED
3800                                     ;                R5 UNCHANGED
3801                                     ;--
3802 040036  116404  000017     RSPERL: MOVB    P.FLGS+RE.RP(R4),R4     ; GET END
3803 040042  042704  177400             BIC     #177400,R4             ;   MESSAGE FLAGS
3804 040046  005703                     TST     R3                     ;
3805 040050  001420                     BEQ     ERRCNT                 ; 000 - CONTROLLER ERROR
3806 040052  022703  000001             CMP     #1,R3                  ;
3807 040056  001457                     BEQ     ERRHMA                 ; 001 - HOST MEMORY ACCESS ERROR
3808 040060  022703  000002             CMP     #2,R3                  ;
3809 040064  001516                     BEQ     ERRDTE                 ; 002 - DISK TRANSFER ERROR
3810 040066  022703  000003             CMP     #3,R3                  ;
3811 040072  001565                     BEQ     ERRSDI                 ; 003 - SDI ERROR
3812 040074  022703  000004             CMP     #4,R3                  ;
3813 040100  001002                     BNE     2$                     ;
3814 040102  000137  040572             JMP     ERRSDE                 ; 004 - SMALL DISK ERROR
3815 040106  000137  027744     2$:     JMP     INVCMD                 ; INVALID END MESSAGE RECEIVED
3816
3817
3818                                     ;++
3819                                     ;        ERRCNT - "CONTROLLER" ERROR ERROR LOG MESSAGE RECEIVED
3820                                     ;
3821                                     ;--
3822 040112  052737  020000  002204 ERRCNT: BIS  #IERL,IFLAGS          ; INDICATE THIS IS AN ERROR LOG
3823 040120  005001                     CLR     R1                     ; TABLE POINTER SANITY CHECK ON CHKSTA
3824 040122  004737  032332             CALL    CHKSTA                 ; CHECK RETURN STATUS
3825 040126  042737  020000  002204     BIC     #IERL,IFLAGS           ; CLEAR ERROR LOG FLAG
3826 040134  032704  000300             BIT     #<LF.CON+LF.SUC>,R4    ; IF OPERATION
3827 040140  001004                     BNE     1$                     ;   FAILED,
3828 040142  012737  000002  002156     MOV     #2,ERRTYP              ;   PRINT HARD ERROR
3829 040150  000407                     BR      2$                     ; ELSE,
3830 040152  032737  000002  002154 1$:  BIT    #SM.PSE,SO.BIT+SFPTBL  ; IF SOFT ERRORS ARE SUPPRESSED,
3831 040160  001414                     BEQ     4$                     ;   SKIP ERROR REPORT.
3832 040162  012737  000003  002156     MOV     #3,ERRTYP              ; ELSE, PRINT SOFT ERROR
3833 040170  012737  000062  002160 2$:  MOV    #50.,ERRNBR            ; SAVE ERROR NUMBER
3834 040176  005037  002162             CLR     ERRMSG                 ; CLEAR ERROR MESSAGE ADDRESS
3835 040202  012737  021654  002164     MOV     #ERR050,ERRBLK         ; SAVE ERROR SUBROUTINE ADDRESS
3836 040210                             ERROR                          ; PRINT MESSAGE
     040210  104460                     TRAP    C$ERROR
3837 040212                     4$:
3838 040212  000240             NOP                                    ; [DBG] INSERT HALT FOR DEBUG
3839 040214  000207                     RETURN                         ;
```

```
3840
3841
3842                                    ;++
3843                                    ;           ERRHMA - "HOST MEMORY ERROR" ERROR LOG MESSAGE RECEIVED
3844                                    ;
3845                                    ;--
3846  040216  052737  020000  002204  ERRHMA: BIS     #IERL,IFLAGS          ; INDICATE THIS IS AN ERROR LOG
3847  040224  005001                          CLR     R1                    ; TABLE POINTER SANITY CHECK ON CHKSTA
3848  040226  004737  032332                  CALL    CHKSTA                ; CHECK RETURN STATUS
3849  040232  042737  020000  002204          BIC     #IERL,IFLAGS          ; CLEAR ERROR LOG FLAG
3850  040240  032704  000300                  BIT     #<LF.CON+LF.SUC>,R4   ; IF OPERATION
3851  040244  001004                          BNE     1$                    ;     FAILED,
3852  040246  012737  000002  002156          MOV     #2,ERRTYP             ;     PRINT HARD ERROR
3853  040254  000407                          BR      2$                    ; ELSE,
3854  040256  032737  000002  002154  1$:     BIT     #SM.PSE,SO.BIT+SFPTBL ; IF SOFT ERRORS ARE SUPPRESSED,
3855  040264  001414                          BEQ     4$                    ;   SKIP ERROR REPORT.
3856  040266  012737  000003  002156          MOV     #3,ERRTYP             ; ELSE, PRINT SOFT ERROR
3857  040274  012737  000063  002160  2$:     MOV     #51.,ERRNBR           ; SAVE ERROR NUMBER
3858  040302  005037  002162                  CLR     ERRMSG                ; CLEAR ERROR MESSAGE ADDRESS
3859  040306  012737  021706  002164          MOV     #ERRO51,ERRBLK        ; SAVE ERROR SUBROUTINE ADDRESS
3860  040314                                  ERROR                         ; PRINT MESSAGE
      040314  104460                          TRAP    C$ERROR
3861  040316                          4$:
3862  040316  000240                          NOP                  ; [DBG] INSERT HALT FOR DEBUG
3863  040320  000207                          RETURN                        ;
3864
3865                                    ;++
3866                                    ;           ERRDTE - "DATA TRANSFER ERROR" ERROR LOG MESSAGE RECEIVED
3867                                    ;
3868                                    ;--
3869
3870  040322  004737  047164          ERRDTE: CALL    GTDRVT                ; ELSE, GET DRIVE TABLE
3871  040326  103445                          BCS     4$                    ;    IF TABLE NOT FOUND, PRINT ERROR
3872  040330  032761  100000  000002          BIT     #DT.AVL,D.UNIT(R1)    ; IF DRIVE DROPPED,
3873  040336  001041                          BNE     4$                    ;    IGNORE RESPONSE PACKET
3874  040340  052737  020000  002204          BIS     #IERL,IFLAGS          ; INDICATE THIS IS AN ERROR LOG
3875  040346  004737  032332                  CALL    CHKSTA                ; CHECK RETURN STATUS
3876  040352  042737  020000  002204          BIC     #IERL,IFLAGS          ; CLEAR ERROR LOG FLAG
3877  040360  032704  000300                  BIT     #<LF.CON+LF.SUC>,R4   ; IF OPERATION
3878  040364  001004                          BNE     2$                    ;     FAILED,
3879  040366  012737  000002  002156          MOV     #2,ERRTYP             ;     PRINT HARD ERROR
3880  040374  000411                          BR      3$                    ; ELSE,
3881  040376  005261  000114          2$:     INC     D.SERR(R1)            ; INCREMENT SOFT ERROR COUNT
3882  040402  032737  000002  002154          BIT     #SM.PSE,SO.BIT+SFPTBL ; IF SOFT ERRORS ARE SUPPRESSED,
3883  040410  001414                          BEQ     4$                    ;   SKIP ERROR REPORT.
3884  040412  012737  000003  002156          MOV     #3,ERRTYP             ; ELSE, PRINT SOFT ERROR
3885  040420  012737  000064  002160  3$:     MOV     #52.,ERRNBR           ; SAVE ERROR NUMBER
3886  040426  005037  002162                  CLR     ERRMSG                ; CLEAR ERROR MESSAGE ADDRESS
3887  040432  012737  022004  002164          MOV     #ERRO52,ERRBLK        ; SAVE ERROR SUBROUTINE ADDRESS
3888  040440                                  ERROR                         ; PRINT MESSAGE
      040440  104460                          TRAP    C$ERROR
3889  040442                          4$:
3890  040442  000240                          NOP                  ; [DBG] INSERT HALT FOR DEBUG
3891  040444  000207                          RETURN                        ;
3892
```

```
3893                                    ;++
3894                                    ;           ERRSDI - "SDI ERROR" ERROR LOG MESSAGE RECEIVED
3895                                    ;
3896                                    ;--
3897
3898 040446  004737  047164    ERRSDI: CALL    GTDRVT                  ; ELSE, GET DRIVE TABLE
3899 040452  103445                    BCS     4$                      ;   IF TABLE NOT FOUND, PRINT ERROR
3900 040454  032761  100000  000002    BIT     #DT.AVL,D.UNIT(R1)      ; IF DRIVE DROPPED,
3901 040462  001041                    BNE     4$                      ;   IGNORE RESPONSE PACKET
3902 040464  052737  020000  002204    BIS     #IERL,IFLAGS            ; INDICATE THIS IS AN ERROR LOG
3903 040472  004737  032332            CALL    CHKSTA                  ; CHECK RETURN STATUS
3904 040476  042737  020000  002204    BIC     #IERL,IFLAGS            ; CLEAR ERROR LOG FLAG
3905 040504  032704  000300            BIT     #<LF.CON+LF.SUC>,R4     ; IF OPERATION
3906 040510  001004                    BNE     2$                      ;   FAILED,
3907 040512  012737  000002  002156    MOV     #2,ERRTYP               ;   PRINT HARD ERROR
3908 040520  000411                    BR      3$                      ; ELSE,
3909 040522  005261  000114    2$:     INC     D.SERR(R1)              ; INCREMENT SOFT ERROR COUNT
3910 040526  032737  000002  002154    BIT     #SM.PSE,SO.BIT+SFPTBL   ; IF SOFT ERRORS ARE SUPPRESSED,
3911 040534  001414                    BEQ     4$                      ;   SKIP ERROR REPORT.
3912 040536  012737  000003  002156    MOV     #3,ERRTYP               ; ELSE, PRINT SOFT ERROR
3913 040544  012737  000065  002160  3$:     MOV     #53.,ERRNBR             ; SAVE ERROR NUMBER
3914 040552  005037  002162            CLR     ERRMSG                  ; CLEAR ERROR MESSAGE ADDRESS
3915 040556  012737  022122  002164    MOV     #ERRO53,ERRBLK          ; SAVE ERROR SUBROUTINE ADDRESS
3916 040564                            ERROR                           ; PRINT MESSAGE
     040564  104460                    TRAP    C$ERROR
3917 040566                    4$:
3918 040566  000240            NOP                     ; [DBG] INSERT HALT FOR DEBUG
3919 040570  000207                    RETURN                          ;
3920
3921
3922                                    ;++
3923                                    ;           ERRSDE - "SMALL DESK ERROR" ERROR LOG MESSAGE RECEIVED
3924                                    ;
3925                                    ;--
3926 040572  004737  047164    ERRSDE: CALL    GTDRVT                  ; ELSE, GET DRIVE TABLE
3927 040576  103445                    BCS     4$                      ;   IF TABLE NOT FOUND, PRINT ERROR
3928 040600  032761  100000  000002    BIT     #DT.AVL,D.UNIT(R1)      ; IF DRIVE DROPPED,
3929 040606  001041                    BNE     4$                      ;   IGNORE RESPONSE PACKET
3930 040610  052737  020000  002204    BIS     #IERL,IFLAGS            ; INDICATE THIS IS AN ERROR LOG
3931 040616  004737  032332            CALL    CHKSTA                  ; CHECK RETURN STATUS
3932 040622  042737  020000  002204    BIC     #IERL,IFLAGS            ; CLEAR ERROR LOG FLAG
3933 040630  032704  000300            BIT     #<LF.CON+LF.SUC>,R4     ; IF OPERATION
3934 040634  001004                    BNE     2$                      ;   FAILED,
3935 040636  012737  000002  002156    MOV     #2,ERRTYP               ;   PRINT HARD ERROR
3936 040644  000411                    BR      3$                      ; ELSE,
3937 040646  005261  000114    2$:     INC     D.SERR(R1)              ; INCREMENT SOFT ERROR COUNT
3938 040652  032737  000002  002154    BIT     #SM.PSE,SO.BIT+SFPTBL   ; IF SOFT ERRORS ARE SUPPRESSED,
3939 040660  001414                    BEQ     4$                      ;   SKIP ERROR REPORT.
3940 040662  012737  000003  002156    MOV     #3,ERRTYP               ; ELSE, PRINT SOFT ERROR
3941 040670  012737  000066  002160  3$:     MOV     #54.,ERRNBR             ; SAVE ERROR NUMBER
3942 040676  005037  002162            CLR     ERRMSG                  ; CLEAR ERROR MESSAGE ADDRESS
3943 040702  012737  022236  002164    MOV     #ERRO54,ERRBLK          ; SAVE ERROR SUBROUTINE ADDRESS
3944 040710                            ERROR                           ; PRINT MESSAGE
     040710  104460                    TRAP    C$ERROR
3945 040712                    4$:
```

```
3946 040712  000240                    NOP                       ; [DBG] INSERT HALT FOR DEBUG
3947 040714  000207                    RETURN                    ;
3948
```

```
3950
3951
3952                                      ;++              T1PGEN - GENERATE TEST PACKETS FOR TEST 1 (CONTROLLER VERIFICATION
3953                                      ;                TEST). THE FOLLOWING STEPS ARE PERFORMED:
3954                                      ;                      1. INITIALIZE EACH CONTROLLER
3955                                      ;                      2. FOR EACH OF THE 16 STANDARD DATA PATTERNS, WRITE ALL
3956                                      ;                         LOCATIONS IN CONTROLLER RAM, READ BACK THE DATA, AND DO
3957                                      ;                         A DATA COMPARE.
3958                                      ;                      3. RE-INITIALIZE CONTROLLER
3959                                      ;                      4. ISSUE A SET CONTROLLER CHARACTERISTICS COMMAND
3960                                      ;--
3961
3962 040716  040764              T1DSP:   .WORD    T1S0                 ; STEP 0 ROUTINE
3963 040720  041052                       .WORD    T1S1                 ; STEP 1 ROUTINE
3964 040722  041170                       .WORD    T1W1                 ; WAIT 1
3965 040724  041212                       .WORD    T1S2                 ; STEP 2 ROUTINE
3966 040726  041300                       .WORD    T1S3                 ; STEP 3 ROUTINE
3967 040730  041416                       .WORD    T1W2                 ; WAIT 2
3968 040732  041500                       .WORD    T1S4                 ; STEP 4 ROUTINE
3969
3970 040734  016500  000030     T1PGEN:  MOV      C.STEP(R5),R0        ; DISPATCH
3971 040740  022700  000006              CMP      #6,R0                ;     TO
3972 040744  103403                       BLO      3$                   ;        CURRENT
3973 040746  006300                       ASL      R0                   ;           STEP
3974 040750  000170  040716              JMP      @T1DSP(R0)           ;              ROUTINE
3975 040754  012701  040752     3$:      MOV      #.-2,R1              ; SAVE CURRENT LOCATION
3976 040760  000137  024310              JMP      BADMEM               ; PRINT SYSTEM FATAL ERROR
3977                                      ;
3978                                      ;       STEP 0 - INITIALIZE THE CONTROLLER
3979                                      ;
3980 040764  004737  052220     T1S0:    CALL     CNTINT               ; INITIALIZE CONTROLLER
3981 040770  103475                       BCS      T1LOE                ; IF ERROR, EXIT
3982 040772  005765  000034              TST      C.MSIZ(R5)           ; IF RAM SIZE = 0,
3983 040776  001005                       BNE      2$                   ;
3984 041000  012765  000006  000030      MOV      #6.,C.STEP(R5)       ;    DO STEP
3985 041006  000137  041500              JMP      T1S4                 ;       4 ONLY
3986 041012  052765  000400  000014 2$:  BIS      #CT.MRW,C.FLG(R5)    ; SET MAINTENANCE OPERATION BIT
3987 041020  012765  000001  000152      MOV      #1,C.CRED(R5)        ; CREDITS SET TO 1 FOR MAINT MODE
3988 041026  012765  000041  000040      MOV      #33.,C.TPAT(R5)      ; INIT TEST PATTERN
3989 041034  012765  000040  000042      MOV      #32.,C.NEXT(R5)      ;
3990 041042  005265  000030              INC      C.STEP(R5)           ; SET STEP COUNT
3991 041046  005065  000036              CLR      C.ROFF(R5)           ; INIT REGION OFFSET
3992
3993                                      ;       STEP 1 - WRITE DATA PATTERN TO CONTROLLER MEMORY
3994                                      ;
3995 041052  012701  000001     T1S1:    MOV      #1,R1                ; GET 1 BLOCK
3996 041056  016503  000040              MOV      C.TPAT(R5),R3        ; GET PATTERN NUMBER
3997 041062  004737  046474              CALL     PATBLD               ; GET BUFFER CONTAINING DATA PATTERN
3998 041066  005701                       TST      R1                   ; IF BUFFER SIZE = 0,
3999 041070  001435                       BEQ      T1LOE                ;    EXIT TEST
4000 041072  016501  000036              MOV      C.ROFF(R5),R1        ; GET REGION OFFSET
4001 041076  016503  000034              MOV      C.MSIZ(R5),R3        ; CALCULATE
4002 041102  160103                       SUB      R1,R3                ;    BYTES
4003 041104  006303                       ASL      R3                   ;       TO
4004 041106  023703  004406              CMP      BCLO,R3              ;          SEND
```

```
4005 041112  103402             BLO     11$
4006 041114  010337  004406     MOV     R3,BCLO                 ; SET BYTE COUNT
4007 041120  005003     11$:    CLR     R3                      ; CLEAR RESERVED
4008 041122  005004             CLR     R4                      ;   FIELDS
4009 041124  004737  051122     CALL    MNTWR                   ; BUILD MAINTENANCE WRITE PACKET
4010 041130  103415             BCS     T1LOE                   ; IF ERROR, EXIT
4011
4012 041132  006037  004406     ROR     BCLO                    ; CONVERT TO WORDS
4013 041136  063701  004406     ADD     BCLO,R1                 ; ADJUST
4014 041142  010165  000036     MOV     R1,C.ROFF(R5)           ;   REGION OFFSET
4015 041146  020165  000034     CMP     R1,C.MSIZ(R5)           ; DONE WRITING?
4016 041152  103404             BLO     T1LOE
4017 041154  005265  000030     INC     C.STEP(R5)              ; NEXT STEP
4018 041160  005065  000036     CLR     C.ROFF(R5)              ; RESET REGION OFFSET
4019 041164             T1LOE:
4020 041164  000240     NOP                     ; [DBG] INSERT HALT FOR DEBUG
4021 041166  000207             RETURN                          ;
4022
4023                    ;       WAIT FOR LAST STEP TO COMPLETE
4024                    ;
4025 041170  005765  000164 T1W1: TST     C.TBSF(R5)            ; IF TBS LIST ISN'T EMPTY,
4026 041174  001003             BNE     1$                      ;   KEEP WAITING
4027 041176  005765  000160     TST     C.CMFL(R5)              ; IF CMD LIST IS EMPTY,
4028 041202  001401             BEQ     2$                      ;   START NEXT STEP
4029 041204  000207     1$:     RETURN                          ;
4030 041206  005265  000030 2$: INC     C.STEP(R5)              ; START NEXT STEP
4031                    ;
4032                    ;       STEP 2 - READ PATTERN FROM MEMORY AND COMPARE DATA
4033                    ;
4034 041212  012701  000001 T1S2: MOV     #1,R1                 ; GET INPUT
4035 041216  004737  024354     CALL    GETBUF                  ;   BUFFER
4036 041222  005701             TST     R1                      ; IF ZERO,
4037 041224  001523             BEQ     T1L1E                   ;   WAIT FOR FREE BUFFER
4038 041226  016501  000036     MOV     C.ROFF(R5),R1           ; GET REGION OFFSET
4039 041232  016503  000034     MOV     C.MSIZ(R5),R3           ; CALCULATE
4040 041236  160103             SUB     R1,R3                   ;   BYTES
4041 041240  006303             ASL     R3                      ;     TO
4042 041242  023703  004406     CMP     BCLO,R3                 ;       SEND
4043 041246  103402             BLO     11$
4044 041250  010337  004406     MOV     R3,BCLO                 ; SET BYTE COUNT
4045 041254  005003     11$:    CLR     R3                      ; CLEAR RESERVED
4046 041256  005004             CLR     R4                      ;   FIELDS
4047 041260  004737  051112     CALL    MNTRD                   ; BUILD MAINTENANCE READ PACKET
4048 041264  103003             BCC     22$                     ; IF ERROR,
4049 041266  004737  025420     CALL    RETBUF                  ;   RETURN BUFFER AND
4050 041272  000500             BR      T1L1E                   ;     EXIT
4051 041274  005265  000030 22$: INC     C.STEP(R5)             ; NEXT STEP
4052                    ;
4053                    ;       STEP 3 - WRITE NEXT PATTERN TO LAST MEMORY BLOCK READ
4054                    ;
4055 041300  012701  000001 T1S3: MOV     #1,R1                 ; GET 1 BLOCK
4056 041304  016503  000042     MOV     C.NEXT(R5),R3           ; GET PATTERN NUMBER
4057 041310  004737  046474     CALL    PATBLD                  ; GET BUFFER CONTAINING DATA PATTERN
4058 041314  005701             TST     R1                      ; IF BUFFER SIZE = 0,
4059 041316  001466             BEQ     T1L1E                   ;   EXIT TEST
```

```
4060 041320  016501  000036                  MOV    C.ROFF(R5),R1         ; GET REGION OFFSET
4061 041324  016503  000034                  MOV    C.MSIZ(R5),R3         ; CALCULATE
4062 041330  160103                           SUB    R1,R3                 ;    BYTES
4063 041332  006303                           ASL    R3                    ;      TO
4064 041334  023703  004406                   CMP    BCLO,R3               ;        SEND
4065 041340  103402                           BLO    11$
4066 041342  010337  004406                   MOV    R3,BCLO               ; SET BYTE COUNT
4067 041346  005003                    11$:   CLR    R3                    ; CLEAR RESERVED
4068 041350  005004                           CLR    R4                    ;    FIELDS
4069 041352  004737  051122                   CALL   MNTWR                 ; BUILD "MAINTENANCE WRITE" PACKET
4070 041356  103446                           BCS    T1L1E                 ; IF ERROR, EXIT
4071
4072 041360  012765  000003  000030           MOV    #3.,C.STEP(R5)        ; NEXT STEP
4073 041366  006037  004406                   ROR    BCLO                  ; CONVERT TO WORDS
4074 041372  063701  004406                   ADD    BCLO,R1               ; ADJUST
4075 041376  010165  000036                   MOV    R1,C.ROFF(R5)         ;    REGION OFFSET
4076 041402  020165  000034                   CMP    R1,C.MSIZ(R5)         ; DONE WRITING?
4077 041406  103432                           BLO    T1L1E                 ;
4078 041410  012765  000005  000030           MOV    #5,C.STEP(R5)         ; SET NEXT STEP
4079                                    ;
4080                                    ;      WAIT FOR LAST STEP TO COMPLETE
4081                                    ;
4082 041416  005765  000164           T1W2:   TST    C.TBSF(R5)            ; IF TBS LIST ISN'T EMPTY,
4083 041422  001024                           BNE    T1L1E                 ;    KEEP WAITING
4084 041424  005765  000160                   TST    C.CMFL(R5)            ; IF CMD LIST ISN'T EMPTY,
4085 041430  001021                           BNE    T1L1E                 ;    KEEP WAITING
4086 041432  012765  000003  000030           MOV    #3,C.STEP(R5)         ; SET NEXT STEP
4087 041440  005065  000036                   CLR    C.ROFF(R5)            ; RESET REGION OFFSET
4088 041444  005365  000042                   DEC    C.NEXT(R5)            ; SAVE LAST PATTERN WRITTEN
4089 041450  002003                           BGE    21$                   ;
4090 041452  012765  000041  000042           MOV    #33.,C.NEXT(R5)       ;
4091 041460  005365  000040           21$:    DEC    C.TPAT(R5)            ;
4092 041464  002003                           BGE    T1L1E                 ; IF NOT NEGATIVE, EXIT LOOP
4093 041466  012765  000006  000030           MOV    #6,C.STEP(R5)         ; SET NEXT STEP
4094 041474                           T1L1E:
4095 041474  000240                    NOP                   ; [DBG] INSERT HALT FOR DEBUG
4096 041476  000207                           RETURN                       ;
4097                                    ;
4098                                    ;
4099                                    ;      STEP 4 - RE-INIT CONTROLLER AND ISSUE SCC COMMAND
4100                                    ;
4101 041500  005765  000164           T1S4:   TST    C.TBSF(R5)            ; DON'T START UNTIL
4102 041504  001025                           BNE    T1L2E                 ;    TBS LIST IS EMPTY
4103 041506  005765  000160                   TST    C.CMFL(R5)            ; DON'T START UNTIL
4104 041512  001022                           BNE    T1L2E                 ;    COMMAND LIST IS EMPTY
4105 041514  042765  000400  000014           BIC    #CT.MRW,C.FLG(R5)     ; CLEAR MAINTENANCE OPERATION BIT
4106 041522  004737  052220                   CALL   CNTINT                ; INITIALIZE CONTROLLER
4107 041526  103414                           BCS    T1L2E                 ; IF ERROR, EXIT
4108 041530  012701  000360                   MOV    #<CF.ATN+CF.MSC+CF.OTH+CF.THS>,R1 ; SET CONTROLLER FLAGS
4109 041534  004737  050210                   CALL   SCC                   ; BUILD SCC COMMAND
4110 041540  103407                           BCS    T1L2E                 ;    IF NO PACKETS, EXIT
4111                                    ;
4112                                    ;      MARK CONTROLLER AS DONE TESTING
4113                                    ;
4114 041542  052765  040000  000014           BIS    #CT.DUN,C.FLG(R5)     ; SET TEST DONE FLAG
```

```
4115 041550  005065  000030              CLR     C.STEP(R5)                  ; RESET STEP COUNTER
4116 041554  005065  000042              CLR     C.NEXT(R5)                  ; RESET NEXT UNIT COUNTER
4117 041560                      T1L2E:
4118 041560  000240              NOP                      ; [DBG] INSERT HALT FOR DEBUG
4119 041562  000207              RETURN                              ;
```

```
4121
4122
4123            ;++                T2PGEN - GENERATE TEST PACKETS FOR TEST 2 (SUBSYSTEM VERIFICATION
4124            ;                   TEST). THE FOLLOWING STEPS ARE PERFORMED:
4125            ;                      0. INIT CONTROLLER & "SET CONTROLLER CHARACTERISTICS"
4126            ;                      1. ISSUE AN "ONLINE" COMMAND FOR EACH DRIVE
4127            ;                      2. ISSUE A "SET UNIT CHARACTERISTICS" COMMAND FOR EACH DRIVE
4128            ;                         WRITE PROTECT ANY DRIVE THAT IS NOT TO BE WRITTEN
4129            ;                      3. ISSUE A "READ" COMMAND WITH THE DATA COMPARE FLAG SET
4130            ;                      4. ISSUE AN "ACCESS" COMMAND
4131            ;                      5. ISSUE A "GET COMMAND STATUS" COMMAND
4132            ;                      6. IF WRITABLE, WRITE 1 BLOCK WITH DATA COMPARE.
4133            ;                      7.   COMPARE THE BLOCK WRITTEN WITH THE PATTERN WRITTEN
4134            ;                      8.   COMPARE THE BLOCK WRITTEN WITH A DIFFERENT DATA PATTERN
4135            ;                      9. SET DRIVES AVAILABLE WITH SPIN DOWN
4136            ;                     10. ISSUE AND "ONLINE" COMMAND FOR EACH DRIVE
4137            ;                     11. SET ALL DRIVES AVAILABLE
4138            ;--
4139
4140 041564 041646          T2DSP:  .WORD   T2S0             ;  1 STEP 0 ROUTINE
4141 041566 041666                  .WORD   T2S1             ;  2 STEP 1 ROUTINE
4142 041570 041720                  .WORD   T2S2             ;  3 STEP 2 ROUTINE
4143 041572 042026                  .WORD   T2W1             ;  4 WAIT 1
4144 041574 042050                  .WORD   T2S3             ;  5 STEP 3 ROUTINE
4145 041576 042122                  .WORD   T2S4             ;  6 STEP 4 ROUTINE
4146 041600 042216                  .WORD   T2S5             ;  7 STEP 5 ROUTINE
4147 041602 042264                  .WORD   T2W2             ;  8 WAIT 2
4148 041604 042306                  .WORD   T2S6             ;  9 STEP 6 ROUTINE
4149 041606 042416                  .WORD   T2W3             ; 10 WAIT 3
4150 041610 042440                  .WORD   T2S7             ; 11 STEP 7 ROUTINE
4151 041612 042570                  .WORD   T2W4             ; 12 WAIT 4
4152 041614 042612                  .WORD   T2S8             ; 13 STEP 8 ROUTINE
4153
4154 041616 016500 000030   T2PGEN: MOV     C.STEP(R5),R0    ; DISPATCH
4155 041622 022700 000014           CMP     #12.,R0          ;   TO
4156 041626 002403                   BLT    3$               ;     CURRENT
4157 041630 006300                   ASL    R0               ;       STEP
4158 041632 000170 041564           JMP     @T2DSP(R0)       ;         ROUTINE
4159 041636 012701 041634   3$:      MOV     #.-2,R1          ; SAVE CURRENT LOCATION
4160 041642 000137 024310           JMP     BADMEM           ; PRINT SYSTEM FATAL ERROR
4161                        ;
4162                        ;       STEP 0 - INITIALIZE THE CONTROLLER AND
4163                        ;       ISSUE A SET CONTROLLER CHARACTERISTICS COMMAND
4164
4165 041646 004737 052220   T2S0:   CALL    CNTINT           ; INITIALIZE CONTROLLER
4166 041652 103463                   BCS    T2L1E            ; IF ERROR, EXIT
4167 041654 012701 000360           MOV     #<CF.ATN+CF.MSC+CF.OTH+CF.THS>,R1; SET CONTROLLER FLAGS
4168 041660 004737 050210           CALL    SCC              ; BUILD SCC COMMAND
4169 041664 103456                   BCS    T2L1E            ;   IF NO PACKETS, EXIT
4170                        ;
4171                        ;       STEP 1 - ISSUE AN "ONLINE" COMMAND FOR EACH DRIVE
4172
4173 041666 004737 045526   T2S1:   CALL    GTNXTD           ; GET POINTER TO DRIVE TABLE
4174 041672 001446                   BEQ    T2S2E            ;   IF ZERO, DONE
4175 041674 032760 100000 000002     BIT     #DT.AVL,D.UNIT(R0) ; CHECK IF DRIVE AVAILABLE
```

```
4176 041702  001030                       BNE     T2S2N              ; IF NOT, SKIP SUBTEST
4177 041704  005001                       CLR     R1                 ; SET UNIT FLAGS
4178 041706  004737  050150               CALL    ONLINE             ; BUILD ONLINE COMMAND
4179 041712  103443                        BCS     T2L1E              ;   IF NO PACKETS, EXIT
4180 041714  005265  000030               INC     C.STEP(R5)         ; GO TO STEP 2
4181
4182                             ;         STEP 2 - ISSUE A "SET UNIT CHARACTERISTICS" COMMAND FOR EACH DRIVE
4183                             ;         WRITE PROTECT ANY DRIVE THAT IS NOT TO BE WRITTEN
4184
4185 041720  004737  045526     T2S2:     CALL    GTNXTD             ; GET POINTER TO DRIVE TABLE
4186 041724  001431                       BEQ     T2S2E              ;   IF ZERO, DONE
4187 041726  032760  100000 000002        BIT     #DT.AVL,D.UNIT(R0) ; CHECK IF DRIVE AVAILABLE
4188 041734  001013                       BNE     T2S2N              ; IF NOT, SKIP SUBTEST
4189 041736  032760  010000 000002        BIT     #DT.WRT,D.UNIT(R0) ; CHECK IF DRIVE WRITEABLE
4190 041744  001003                       BNE     21$                ; IF NOT,
4191 041746  012701  010000               MOV     #UF.WPS,R1         ;   SET SOFTWARE WRITE PROTECT
4192 041752  000401                       BR      22$                ;
4193 041754  005001            21$:       CLR     R1                 ; ELSE, SET UNIT FLAGS
4194 041756  004737  050246    22$:       CALL    SUCHAR             ; BUILD "SUC" COMMAND
4195 041762  103417                       BCS     T2L1E              ;   IF NO PACKETS, EXIT
4196 041764  022765  000003 000042 T2S2N: CMP     #3,.C.NEXT(R5)     ; CHECK FOR END OF LIST
4197 041772  001406                       BEQ     T2S2E              ; IF SO, GO TO NEXT SECTION
4198 041774  005265  000042               INC     C.NEXT(R5)         ; ELSE,
4199 042000  012765  000001 000030        MOV     #1,C.STEP(R5)      ;   DO NEXT
4200 042006  000727                       BR      T2S1               ;     DRIVE
4201
4202 042010  012765  000003 000030 T2S2E: MOV     #3,.C.STEP(R5)     ; SET STEP
4203 042016  005065  000042               CLR     C.NEXT(R5)         ;   NEXT DRIVE IS 1ST IN LIST
4204 042022                     T2L1E:
4205 042022  000240            NOP                         ; [DBG] INSERT HALT FOR DEBUG
4206 042024  000207                       RETURN                     ; EXIT TEST
4207                             ;
4208                             ;         WAIT FOR LAST STEP TO COMPLETE
4209
4210 042026  005765  000164     T2W1:     TST     C.TBSF(R5)         ; IF TBS LIST ISN'T EMPTY,
4211 042032  001003                       BNE     1$                 ;   KEEP WAITING
4212 042034  005765  000160               TST     C.CMFL(R5)         ; IF CMD LIST IS EMPTY,
4213 042040  001401                       BEQ     2$                 ;   START NEXT STEP
4214 042042  000207            1$:        RETURN                     ;
4215 042044  005265  000030    2$:        INC     C.STEP(R5)         ; START NEXT STEP
4216                             ;
4217                             ;         STEP 3 - ISSUE A "READ" COMMAND WITH THE DATA COMPARE FLAG SET
4218
4219 042050  004737  045526     T2S3:     CALL    GTNXTD             ; GET POINTER TO DRIVE TABLE
4220 042054  001453                       BEQ     T2S4E              ;   IF ZERO, DONE
4221 042056  032760  100000 000002        BIT     #DT.AVL,D.UNIT(R0) ; CHECK IF DRIVE AVAILABLE
4222 042064  001035                       BNE     T2S4N              ; IF NOT, SKIP SUBTEST
4223 042066  012701  000001               MOV     #1,.R1             ; GET A 1 BLOCK
4224 042072  004737  024354               CALL    GETBUF             ;   DATA BUFFER
4225 042076  005701                       TST     R1                 ; IF ZERO,
4226 042100  001467                       BEQ     T2L2E              ;   EXIT
4227 042102  004737  050556               CALL    READ               ; BUILD READ COMMAND
4228 042106  103003                       BCC     22$                ; IF ERROR,
4229 042110  004737  025420               CALL    RETBUF             ;   RETURN BUFFER AND
4230 042114  000461                       BR      T2L2E              ;     EXIT
```

```
4231 042116 005265 000030        22$:   INC    C.STEP(R5)              ; GOTO NEXT STEP
4232                              ;
4233                              ;      STEP 4 - ISSUE AN "ACCESS" COMMAND
4234
4235 042122 004737 045526        T2S4:  CALL   GTNXTD                 ; GET POINTER TO DRIVE TABLE
4236 042126 001426                      BEQ    T2S4E                  ;   IF ZERO, DONE
4237 042130 032760 100000 000002        BIT    #DT.AVL.D.UNIT(R0)     ; CHECK IF DRIVE AVAILABLE
4238 042136 001010                      BNE    T2S4N                  ; IF NOT, SKIP SUBTEST
4239 042140 012737 024000 004406        MOV    #10240..BCLO           ; SET BYTECOUNT
4240 042146 005037 004410               CLR    BCHI                   ;   TO 20 BLOCKS
4241 042152 004737 050306               CALL   ACCESS                 ; BUILD "ACCESS" COMMAND
4242 042156 103440                       BCS    T2L2E                  ;   IF NO PACKETS, EXIT
4243
4244 042160 022765 000003 000042 T2S4N: CMP    #3..C.NEXT(R5)         ; CHECK FOR END OF LIST
4245 042166 001406                      BEQ    T2S4E                  ; IF SO, GO TO NEXT SECTION
4246 042170 005265 000042               INC    C.NEXT(R5)             ; ELSE,
4247 042174 012765 000004 000030        MOV    #4,C.STEP(R5)          ;   DO NEXT
4248 042202 000722                       BR     T2S3                   ;     DRIVE
4249
4250 042204 012765 000006 000030 T2S4E: MOV    #6..C.STEP(R5)         ;   DO NEXT STEP
4251 042212 005065 000042               CLR    C.NEXT(R5)             ;   NEXT DRIVE IS 1ST IN LIST
4252                              ;
4253                              ;      STEP 5 - ISSUE A "GET COMMAND STATUS" COMMAND
4254
4255 042216 005765 000160        T2S5:  TST    C.CMFL(R5)             ; IF CMD LIST NOT EMPTY,
4256 042222 001004                      BNE    1$                     ;   DO GCS COMMAND
4257 042224 005765 000164               TST    C.TBSF(R5)             ; IF TBS & CMD LIST EMPTY,
4258 042230 001013                      BNE    T2L2E                  ;   SKIP GCS COMMAND
4259 042232 000410                       BR     2$                     ;
4260 042234 016504 000160        1$:    MOV    C.CMFL(R5),R4          ; GET COMMAND REFERENCE NUMBER
4261 042240 016401 000004               MOV    ME.CRF(R4),R1          ;   OF NEWEST COMMAND
4262 042244 005002                      CLR    R2                     ; CLEAR HIGH WORD
4263 042246 004737 050056               CALL   GCSTAT                 ; BUILD "GCS" COMMAND
4264 042252 103402                       BCS    T2L2E                  ;   IF NO PACKETS, EXIT
4265 042254 005265 000030        2$:    INC    C.STEP(R5)             ;   DO NEXT STEP
4266
4267 042260                      T2L2E:
4268 042260 000240               NOP                                  ; [DBG] INSERT HALT FOR DEBUG
4269 042262 000207                      RETURN                        ; EXIT TEST
4270
4271                              ;      WAIT FOR LAST STEP TO COMPLETE
4272
4273 042264 005765 000164        T2W2:  TST    C.TBSF(R5)             ; IF TBS LIST ISN'T EMPTY,
4274 042270 001003                      BNE    1$                     ;   KEEP WAITING
4275 042272 005765 000160               TST    C.CMFL(R5)             ; IF CMD LIST IS EMPTY,
4276 042276 001401                      BEQ    2$                     ;   START NEXT STEP
4277 042300 000207               1$:    RETURN
4278 042302 005265 000030        2$:    INC    C.STEP(R5)             ; START NEXT STEP
4279                              ;
4280                              ;      STEP 6 - IF WRITEABLE, ISSUE AN "ERASE" COMMAND
4281
4282 042306 032737 000100 002204 T2S6:  BIT    #IPAT16,IFLAGS         ; IF PATTERN 16 CHANGED,
4283 042314 001032                      BNE    T2S6E                  ;   SKIP THIS TEST
4284 042316 004737 045526               CALL   GTNXTD                 ; GET POINTER TO DRIVE TABLE
4285 042322 001427                      BEQ    T2S6E                  ;   IF ZERO, DONE
```

```
4286 042324  032760 100000 000002        BIT    #DT.AVL.D.UNIT(R0)   ; CHECK IF DRIVE AVAILABLE
4287 042332  001014                       BNE    T2S6N                ; IF NOT, SKIP SUBTEST
4288 042334  032760 010000 000002        BIT    #DT.WRT.D.UNIT(R0)   ; CHECK IF DRIVE WRITEABLE
4289 042342  001410                       BEQ    T2S6N                ; IF NOT, SKIP WRITE/COMPARE TEST
4290 042344  012737 001000 004406        MOV    #512..BCLO           ; SET BYTECOUNT
4291 042352  005037 004410               CLR    BCHI                 ;   TO 1 BLOCK
4292 042356  004737 050532               CALL   ERASE                ; BUILD "ERASE" COMMAND
4293 042362  103413                       BCS    T2S6R                ;   IF NO PACKETS, EXIT
4294
4295 042364  022765 000003 000042  T2S6N: CMP   #3..C.NEXT(R5)       ; CHECK FOR END OF LIST
4296 042372  001403                       BEQ    T2S6E                ; IF SO, GO TO NEXT SECTION
4297 042374  005265 000042               INC    C.NEXT(R5)           ; ELSE,
4298 042400  000742                       BR     T2S6                 ;   NEXT DRIVE
4299
4300 042402  005265 000030        T2S6E:  INC    C.STEP(R5)           ; SET STEP
4301 042406  005065 000042               CLR    C.NEXT(R5)           ;   NEXT DRIVE IS 1ST IN LIST
4302 042412                        T2S6R:
4303 042412  000240               NOP                                ; [DBG] INSERT HALT FOR DEBUG
4304 042414  000207                       RETURN                     ; EXIT TEST
4305                               ;
4306                               ;       WAIT FOR LAST STEP TO COMPLETE
4307                               ;
4308 042416  005765 000164        T2W3:   TST    C.TBSF(R5)           ; IF TBS LIST ISN'T EMPTY,
4309 042422  001003                       BNE    1$                   ;   KEEP WAITING
4310 042424  005765 000160               TST    C.CMFL(R5)           ; IF CMD LIST IS EMPTY,
4311 042430  001401                       BEQ    2$                   ;   START NEXT STEP
4312 042432  000207               1$:     RETURN
4313 042434  005265 000030        2$:     INC    C.STEP(R5)           ; START NEXT STEP
4314
4315                               ;       STEP 7 - IF WRITABLE, WRITE 16 BLOCKS WITH DATA COMPARE. ALSO
4316                               ;       DO "COMPARE HOST DATA" ON BLOCKS WRITTEN
4317                               ;
4318 042440  004737 045526        T2S7:   CALL   GTNXTD               ; GET POINTER TO DRIVE TABLE
4319 042444  001443                       BEQ    T2S7E                ;   IF ZERO, DONE
4320 042446  032760 100000 000002        BIT    #DT.AVL.D.UNIT(R0)   ; CHECK IF DRIVE AVAILABLE
4321 042454  001030                       BNE    T2S7N                ; IF NOT, SKIP SUBTEST
4322          032760 010000 000002        BIT    #DT.WRT.D.UNIT(R0)   ; CHECK IF DRIVE WRITEABLE
4323          001424                       BEQ    T2S7N                ; IF NOT, SKIP WRITE/COMPARE TEST
4324 042472  012701 000020               MOV    #16..R1              ; GET A 16 BLOCK
4325          012703 000021               MOV    #17..R3              ;   DATA BUFFER
4326          004737 046474               CALL   PATBLD               ;     CONTAINING ALL PATTERNS
4327          005701                       TST    R1                   ; IF ZERO,
4328          001427                       BEQ    T2S7R                ;   EXIT
4329          052760 004000 000002        BIS    #DT.CMP.D.UNIT(R0)   ; SET COMPARE PENDING BIT
4330          004737 050646               CALL   WRITE                ; BUILD "WRITE" COMMAND
4331          103006                       BCC    T2S7N                ; IF ERROR,
4332          042760 004000 000002        BIC    #DT.CMP.D.UNIT(R0)   ;   CLEAR DATA CHECK FLAG,
4333          004737 025420               CALL   RETBUF               ;     RETURN BUFFER, &
4334 042534  000413                       BR     T2S7R                ;       EXIT
4335
4336 042536  022765 000003 000042  T2S7N: CMP   #3..C.NEXT(R5)       ; CHECK FOR END OF LIST
4337 042544  001403                       BEQ    T2S7E                ; IF SO, GO TO NEXT SECTION
4338 042546  005265 000042               INC    C.NEXT(R5)           ; ELSE,
4339 042552  000732                       BR     T2S7                 ;   NEXT DRIVE
4340
```

```
4341 042554 005265 000030      T2S7E:  INC    C.STEP(R5)                 ; SET STEP
4342 042560 005065 000042              CLR    C.NEXT(R5)                 ;   NEXT DRIVE IS 1ST IN LIST
4343 042564                    T2S7R:
4344 042564 000240             NOP                               ; [DBG] INSERT HALT FOR DEBUG
4345 042566 000207                     RETURN                    ; EXIT TEST
4346                            ;
4347                            ;      WAIT FOR LAST STEP TO COMPLETE
4348                            ;
4349 042570 005765 000164      T2W4:   TST    C.TBSF(R5)                 ; IF TBS LIST ISN'T EMPTY,
4350 042574 001003                     BNE    1$                         ;   KEEP WAITING
4351 042576 005765 000160              TST    C.CMFL(R5)                 ; IF CMD LIST IS EMPTY,
4352 042602 001401                     BEQ    2$                         ;   START NEXT STEP
4353 042604 000207             1$:     RETURN                            ;
4354 042606 005265 000030      2$:     INC    C.STEP(R5)                 ; START NEXT STEP
4355                            ;
4356                            ;      STEP 8 - SET DRIVES AVAILABLE
4357                            ;
4358 042612 004737 045526      T2S8:   CALL   GTNXTD                     ; GET POINTER TO DRIVE TABLE
4359 042616 001422                     BEQ    T2S8E                      ; IF ZERO, DONE
4360 042620 032750 100000 000002       BIT    #DT.AVL,D.UNIT(R0)         ; CHECK IF DRIVE AVAILABLE
4361 042626 001004                     BNE    10$                        ; IF NOT, SKIP SUBTEST
4362 042630 005003                     CLR    R3                         ; CLEAR COMMAND MODIFIERS
4363 042632 004737 050020              CALL   AVAIL                      ; BUILD "AVAILABLE" COMMAND
4364 042636 103421                     BCS    T2S8R                      ;   IF NO PACKETS, EXIT
4365 042640 022765 000003 000042 10$:  CMP    #3.,C.NEXT(R5)            ; CHECK FOR END OF LIST
4366 042646 001406                     BEQ    T2S8E                      ; IF SO, GO TO NEXT SECTION
4367 042650 005265 000042              INC    C.NEXT(R5)                 ; ELSE,
4368 042654 012765 000013 000030       MOV    #11.,C.STEP(R5)           ;   DO NEXT
4369 042662 000753                     BR     T2S8                       ;      DRIVE
4370                            ;
4371                            ;      MARK CONTROLLER AS DONE TESTING
4372                            ;
4373 042664 005065 000030      T2S8E:  CLR    C.STEP(R5)                 ; RESET TEST STEP COUNTER
4374 042670 005065 000042              CLR    C.NEXT(R5)                 ;   NEXT DRIVE IS 1ST IN LIST
4375 042674 052765 040000 000014       BIS    #CT.DUN,C.FLG(R5)         ; SET TEST DONE FLAG
4376 042702                    T2S8R:
4377 042702 000240             NOP                               ; [DBG] INSERT HALT FOR DEBUG
4378 042704 000207                     RETURN                            ;
```

```
4380
4381
4382            ;++
4383            ;        T3PGEN - GENERATE TEST PACKETS FOR TEST 3 (SUBSYSTEM EXERCISER
4384            ;        TEST). THE FOLLOWING STEPS ARE PERFORMED:
4385            ;              0.  INIT EACH CONTROLLER & "SET CONTROLLER CHARACTERISTICS"
4386            ;              1.  SET EACH DRIVE "ONLINE"
4387            ;              2.  "SET UNIT CHARACTERISTICS" FOR EACH DRIVE
4388            ;                  WRITE PROTECT ANY DRIVE THAT IS NOT TO BE WRITTEN
4389            ;              3.  "WRITE" EACH DISK USING ALL DATA PATTERNS IF WRITABLE
4390            ;              4.  "READ" FIRST 512 BLOCKS ON EACH DISK
4391            ;              5.  "ACCESS" REMAINDER OF DISK
4392            ;              6.  "READ" & "WRITE" DISK AS PER SOFTWARE QUESTIONS
4393            ;              7.  SET EACH DRIVE "AVAILABLE"
4394            ;--
4395 042706 042766    T3DSP:  .WORD  T3S0            ; STEP 0 ROUTINE
4396 042710 043026            .WORD  T3S1            ; STEP 1 ROUTINE
4397 042712 043060            .WORD  T3S2            ; STEP 2 ROUTINE
4398 042714 043166            .WORD  T3W1            ; WAIT 1
4399 042716 043350            .WORD  T3S3            ; STEP 3 ROUTINE
4400 042720 043634            .WORD  T3W2            ; WAIT 2
4401 042722 043656            .WORD  T3S4            ; STEP 4 ROUTINE
4402 042724 044140            .WORD  T3S5            ; STEP 5 ROUTINE
4403 042726 044352            .WORD  T3W4            ; WAIT 4
4404 042730 044524            .WORD  T3S6            ; STEP 6 ROUTINE
4405 042732 045366            .WORD  T3W5            ; WAIT 5
4406 042734 045410            .WORD  T3S7            ; STEP 7 ROUTINE
4407
4408 042736 016500 000030  T3PGEN: MOV   C.STEP(R5),R0         ; DISPATCH
4409 042742 022700 000013          CMP   #11.,R0              ;    TO
4410 042746 103403                 BLO   3$                   ;       CURRENT
4411 042750 006300                 ASL   R0                   ;          STEP
4412 042752 000170 042706          JMP   @T3DSP(R0)           ;             ROUTINE
4413 042756 012701 042754  3$:     MOV   #.-2,R1              ; SAVE CURRENT LOCATION
4414 042762 000137 024310          JMP   @ADMEM               ; PRINT SYSTEM FATAL ERROR
4415
4416            ;        STEP 0 - INITIALIZE THE CONTROLLER AND
4417            ;        ISSUE A SET CONTROLLER CHARACTERISTICS COMMAND
4418
4419 042766 004737 052220  T3S0:   CALL  CNTINT               ; INITIALIZE THE CONTROLLER
4420 042772 103473                 BCS   T3L1E                ; IF ERROR, EXIT
4421 042774 012701 000360          MOV   #<CF.ATN+CF.MSC+CF.OTH+CF.THS>,R1; SET CONTROLLER FLAGS
4422 043000 005003                 CLR   R3                   ; NO MODIFIERS ALLOWED
4423 043002 005004                 CLR   R4                   ; UNIT MUST BE ZERO
4424 043004 004737 050210          CALL  SCC                  ; BUILD SCC COMMAND
4425 043010 103464                 BCS   T3L1E                ;    IF NO PACKETS, EXIT
4426 043012 005265 000030          INC   C.STEP(R5)           ; SET STEP COUNT
4427 043016 005065 000042          CLR   C.NEXT(R5)           ; INIT NEXT DRIVE INDEX
4428 043022 005065 000032          CLR   C.UCNT(R5)           ; INIT DRIVE UNDER TEST COUNT
4429            ;
4430            ;        STEP 1 - ISSUE AN "ONLINE" COMMAND FOR EACH DRIVE
4431
4432 043026 004737 045526  T3S1:   CALL  G*NXTD               ; GET POINTER TO DRIVE TABLE
4433 043032 001446                 BEQ   T3S2E                ; IF ZERO, DONE
4434 043034 032760 100000 000002   BIT   #DT.AVL,D.UNIT(R0)   ; CHECK IF DRIVE AVAILABLE
```

```
4435 043042  001030                        BNE    T3S2N              ; IF NOT, SKIP SUBTEST
4436 043044  005001                        CLR    R1                 ; SET UNIT FLAGS
4437 043046  004737  050150                CALL   ONLINE             ; BUILD ONLINE COMMAND
4438 043052  103443                        BCS    T3L1E              ;   IF NO PACKETS, EXIT
4439 043054  005265  000030                INC    C.STEP(R5)         ; GO TO STEP 2
4440
4441                             ;          STEP 2 - ISSUE A "SET UNIT CHARACTERISTICS" COMMAND FOR EACH DRIVE
4442                             ;          WRITE PROTECT ANY DRIVE THAT IS NOT TO BE WRITTEN
4443
4444 043060  004737  045526     T3S2:       CALL   GTNXTD             ; GET POINTER TO DRIVE TABLE
4445 043064  001431                         BEQ    T3S2E              ; IF ZERO, DONE
4446 043066  032760  100000 000002          BIT    #DT.AVL,D.UNIT(R0) ; CHECK IF DRIVE AVAILABLE
4447 043074  001013                          BNE    T3S2N              ; IF NOT, SKIP SUBTEST
4448 043076  032760  010000 000002          BIT    #DT.WRT,D.UNIT(R0) ; CHECK IF DRIVE WRITEABLE
4449 043104  001003                          BNE    21$                ; IF NOT,
4450 043106  012701  010000                  MOV    #UF.WPS,R1         ;   SET SOFTWARE WRITE PROTECT
4451 043112  000401                           BR     22$                ;
4452 043114  005001               21$:       CLR    R1                 ; ELSE, SET UNIT FLAGS
4453 043116  004737  050246      22$:       CALL   SUCHAR             ; BUILD "SUC" COMMAND
4454 043122  103417                          BCS    T3L1E              ;   IF NO PACKETS, EXIT
4455 043124  022765  000003 000042 T3S2N:   CMP    #3.,C.NEXT(R5)     ; CHECK FOR END OF LIST
4456 043132  001406                          BEQ    T3S2E              ; IF SO, GO TO NEXT SECTION
4457 043134  005265  000042                  INC    C.NEXT(R5)         ; ELSE,
4458 043140  012765  000001 000030           MOV    #1,C.STEP(R5)      ;   NEXT
4459 043146  000727                           BR     T3S1               ;     DRIVE
4460
4461 043150  012765  000003 000030 T3S2E:   MOV    #3,C.STEP(R5)      ; SET NEXT STEP
4462 043156  005065  000042                  CLR    C.NEXT(R5)         ; RESET NEXT DRIVE POINTER
4463 043162              T3L1E:
4464 043162  000240     NOP                                  ; [DBG] INSERT HALT FOR DEBUG
4465 043164  000207                          RETURN                    ; EXIT TEST
4466
4467                             ;          WAIT FOR LAST STEP TO COMPLETE
4468                             ;
4469 043166  005765  000164     T3W1:       TST    C.TBSF(R5)         ; IF TBS LIST ISN'T EMPTY,
4470 043172  001003                          BNE    1$                 ;   KEEP WAITING
4471 043174  005765  000160                  TST    C.CMFL(R5)         ; IF CMD LIST IS EMPTY,
4472 043200  001401                          BEQ    2$                 ;   START NEXT STEP
4473 043202  000207               1$:        RETURN
4474 043204  032765  002000 000014 2$:       BIT    #CT.DET,C.FLG(R5)  ; IF DETERMINISTIC PHASE NOT DONE,
4475 043212  001005                          BNE    3$                 ;   DO DETERMINISTIC PHASE
4476 043214  012765  000011 000030           MOV    #9.,C.STEP(R5)     ; ELSE,
4477 043222  000137  044524                  JMP    T3S6               ;   DO USER DEFINED PHASE
4478 043226  005265  000030     3$:          INC    C.STEP(R5)         ; DO NEXT STEP
4479 043232  005065  000042                  CLR    C.NEXT(R5)         ; POINT TO FIRST DRIVE
4480 043236  004737  045526     10$:         CALL   GTNXTD             ; GET POINTER TO NEXT DRIVE TABLE
4481 043242  001435                          BEQ    15$                ; IF ZERO, DONE
4482 043244  042760  040000 000002           BIC    #DT.DUN,D.UNIT(R0) ; CLEAR SUBTEST DONE FLAG
4483 043252  022737  000001 002202           CMP    #1,CTRLRS          ; IF MORE THAN 1 CONTROLLER
4484 043260  001004                          BNE    11$                ;   CAN'T HOG ALL THE BUFFERS FOR WRITE
4485 043262  012760  000200 000050           MOV    #128.,D.BLKS(R0)   ; SET BLOCK SIZE TO ENTIRE BUFFER
4486 043270  000403                           BR     12$                ;
4487 043272  012760  000140 000050 11$:      MOV    #96.,D.BLKS(R0)    ; LEAVE 32 BLOCKS FOR INITIAL READ
4488 043300  016060  000006 000052 12$:      MOV    D.BGN1(R0),D.LBN(R0)   ; SET NEXT LBN TO ACCESS TO
4489 043306  016060  000010 000054           MOV    D.BGN1+2(R0),D.LBN+2(R0); FIRST BLOCK IN FIRST BEGIN/END SET
```

```
4490 043314  005265  000032                       INC    C.UCNT(R5)                 ; INCREMENT UNIT UNDER TEST COUNT
4491 043320  022765  000003  000042                CMP    #3,.C.NEXT(R5)            ; CHECK FOR END OF LIST
4492 043326  001403                                BEQ    15$                       ; IF SO, GET NEXT PATTERN
4493 043330  005265  000042                        INC    C.NEXT(R5)                ; ELSE,
4494 043334  000740                                BR     10$                       ;   NEXT DRIVE
4495 043336  005065  000042            15$:        CLR    C.NEXT(R5)                ; INIT DRIVE POINTER
4496 043342  005765  000032                        TST    C.UCNT(R5)                ; IF NO DRIVES AVAILABLE,
4497 043346  001472                                BEQ    T3S3E                     ;    EXIT STEP
4498
4499                                    ;          STEP 3 - IF WRITABLE, SEQUENTIALLY WRITE ENTIRE DISK. WRITE EACH OF
4500                                    ;          THE DATA PATTERNS IN 128 BLOCKS SEGMENTS.
4501
4502 043350  004737  045526            T3S3:       CALL   GTNXTD                    ; GET POINTER TO DRIVE TABLE
4503 043354  001455                                BEQ    50$                       ; IF ZERO, DONE
4504 043356  032760  140000  000002                BIT    #DT.AVL!DT.DUN,D.UNIT(R0) ; CHECK IF DRIVE AVAILABLE
4505 043364  001037                                BNE    40$                       ; IF NOT, SKIP WRITE TEST
4506 043366  032760  010000  000002                BIT    #DT.WRT,D.UNIT(R0)        ; CHECK IF DRIVE WRITEABLE
4507 043374  001433                                BEQ    40$                       ; IF NOT, SKIP WRITE TEST
4508
4509                                    ;          GET LBN TO ACCESS
4510
4511 043376  004737  045550            32$:        CALL   GETLBN                    ; GENERATE NEXT LBN TO ACCESS
4512 043402  032760  002000  000002                BIT    #DT.EOM,D.UNIT(R0)        ; IF END OF MEDIA,
4513 043410  001025                                BNE    40$                       ;   EXIT SUBTEST
4514
4515                                    ;          GET WRITE BUFFER AND WRITE THE DISK.
4516
4517 043412  016001  000050            37$:        MOV    D.BLKS(R0),R1             ; GET BUFFER SIZE
4518 043416  012703  000021                        MOV    #17.,R3                   ; SET PATTERN NUMBER (ALL PATTERNS)
4519 043422  004737  046474                        CALL   PATBLD                    ; BUILD DATA PATTERN
4520 043426  005701                                TST    R1                        ; IF PATTERN LENGTH ZERO,
4521 043430  001477                                BEQ    T3L2E                     ;   EXIT
4522 043432                                        PUSH   <R1>                      ; SAVE BLOCK COUNT
     043432  010146                                MOV    R1,-(SP)                  ; PUSH R1 ON STACK
4523 043434  004737  050646                        CALL   WRITE                     ; BUILD "WRITE" COMMAND
4524 043440                                        POP    <R1>                      ; RESTORE BLOCK COUNT
     043440  012601                                MOV    (SP)+,R1                  ; POP STACK INTO R1
4525 043442  103003                                BCC    39$                       ; IF ERROR,
4526 043444  004737  025420                        CALL   RETBUF                    ;   RETURN BUFFER AND
4527 043450  000467                                BR     T3L2E                     ;     EXIT
4528
4529 043452  060160  000052            39$:        ADD    R1,D.LBN(R0)              ; SET NEXT LBN
4530 043456  005560  000054                        ADC    D.LBN+2(R0)               ;   TO ACCESS
4531 043462  000412                                BR     50$                       ; DO NEXT DRIVE
4532
4533                                    ;          PATTERN DONE ON THIS DRIVE, DROP IT FROM LOOP
4534
4535 043464  032760  040000  000002    40$:        BIT    #DT.DUN,D.UNIT(R0)        ; IF ALREADY MARKED DONE,
4536 043472  001006                                BNE    50$                       ;   FINISH OTHER DRIVES
4537 043474  052760  040000  000002                BIS    #DT.DUN,D.UNIT(R0)        ; SET DONE BIT
4538 043502  005365  000032                        DEC    C.UCNT(R5)                ; DECREMENT UNIT UNDER TEST COUNT
4539 043506  001412                                BEQ    T3S3E                     ; IF ZERO, EXIT
4540
4541                                    ;          GET NEXT DRIVE IN SEQUENCE
4542                                    ;
```

```
4543 043510  022765  000003  000042  50$:    CMP     #3.,C.NEXT(R5)          ; IF NOT END
4544 043516  001403                          BEQ     51$                     ;   OF DRIVE LIST,
4545 043520  005265  000042                  INC     C.NEXT(R5)              ; THEN, GET NEXT DRIVE &
4546 043524  000711                          BR      T3S3                    ;   BUILD MORE PACKETS
4547 043526  005065  000042  51$:    CLR     C.NEXT(R5)              ; ELSE, RESET NEXT DRIVE POINTER &
4548 043532  000436                          BR      T3L2E                   ;   GO SEND WHAT WE HAVE BUILT
4549                                  ;
4550                                  ;       END OF STEP. CLEANUP AND GO TO NEXT STEP
4551                                  ;
4552 043534  005065  000042  T3S3E:  CLR     C.NEXT(R5)              ; POINT TO FIRST DRIVE
4553 043540  004737  045526  10$:    CALL    GTNXTD                  ; GET POINTER TO NEXT DRIVE TABLE
4554 043544  001425                          BEQ     15$                     ; IF ZERO, DONE
4555 043546  042760  042000  000002          BIC     #<DT.EOM+DT.DUN>,D.UNIT(R0); CLEAR "END OF MEDIA" & "DONE"
4556 043554  012760  001000  000050          MOV     #512.,D.BLKS(R0)        ; GET LENGTH OF I/O
4557 043562  016060  000006  000052          MOV     D.BGN1(R0),D.LBN(R0)    ; SET NEXT LBN TO ACCESS TO
4558 043570  016060  000010  000054          MOV     D.BGN1+2(R0),D.LBN+2(R0); FIRST BLOCK IN FIRST BEGIN/END SET
4559 043576  005265  000032          INC     C.UCNT(R5)              ; INCREMENT UNIT UNDER TEST COUNT
4560 043602  022765  000003  000042          CMP     #3.,C.NEXT(R5)          ; CHECK FOR END OF LIST
4561 043610  001403                          BEQ     15$                     ; IF SO, END CLEANUP
4562 043612  005265  000042                  INC     C.NEXT(R5)              ; ELSE,
4563 043616  000750                          BR      10$                     ;   NEXT DRIVE
4564 043620  005065  000042  15$:    CLR     C.NEXT(R5)              ; RESET NEXT DRIVE POINTER
4565 043624  005265  000030          INC     C.STEP(R5)              ; SET STEP
4566 043630                  T3L2E:
4567 043630  000240          NOP                     ; [DBG] INSERT HALT FOR DEBUG
4568 043632  000207                  RETURN                          ; EXIT TEST
4569                                  ;
4570                                  ;       WAIT FOR LAST STEP TO COMPLETE
4571                                  ;
4572 043634  005765  000164  T3W2:   TST     C.TBSF(R5)              ; IF TBS LIST ISN'T EMPTY,
4573 043640  001003                          BNE     1$                      ;   KEEP WAITING
4574 043642  005765  000160                  TST     C.CMFL(R5)              ; IF CMD LIST IS EMPTY,
4575 043646  001401                          BEQ     2$                      ;   START NEXT STEP
4576 043650  000207          1$:     RETURN
4577 043652  005265  000030  2$:     INC     C.STEP(R5)              ; DO NEXT STEP
4578                                  ;
4579                                  ;       STEP 4 - "READ" THE FIRST 512 BLOCKS OF EACH DISK. PERFORM
4580                                  ;       DATA VERIFICATION ON THE DATA READ.
4581                                  ;
4582 043656  004737  045526  T3S4:   CALL    GTNXTD                  ; GET POINTER TO DRIVE TABLE
4583 043662  001456                          BEQ     50$                     ; IF ZERO, DONE
4584 043664  032760  140000  000002          BIT     #DT.AVL!DT.DUN,D.UNIT(R0); CHECK IF DRIVE AVAILABLE FOR TEST
4585 043672  001040                          BNE     40$                     ; IF NOT, SKIP SUBTEST
4586                                  ;
4587                                  ;       GET LBN TO ACCESS
4588                                  ;
4589 043674  004737  045550  32$:    CALL    GETLBN                  ; GENERATE NEXT LBN TO ACCESS
4590 043700  032760  002000  000002          BIT     #DT.EOM,D.UNIT(R0)      ; IF END OF MEDIA,
4591 043706  001032                          BNE     40$                     ;   EXIT SUBTEST
4592                                  ;
4593                                  ;       GET INPUT BUFFER, DO I/O AND UPDATE COUNTERS
4594                                  ;
4595 043710  012701  000010          MOV     #8.,R1                  ; GET AN
4596 043714  020160  000050          CMP     R1,D.BLKS(R0)           ;   INPUT
4597 043720  101402                          BLOS    33$                     ;     DATA
```

```
4598 043722  016001  000050           MOV      D.BLKS(RO),R1           ;          BUFFER
4599 043726  004737  024354   33$:     CALL     GETBUF
4600 043732  005701                    TST      R1                      ; IF ZERO,
4601 043734  001477                    BEQ      T3L3E                   ;    EXIT
4602 043736                            PUSH     <R1>                    ; SAVE BLOCK COUNT
     043736  010146                    MOV      R1,-(SP)                ; PUSH R1 ON STACK
4603 043740  004737  050556            CALL     READ                    ; BUILD "READ" COMMAND
4604 043744                            POP      <R1>                    ; RESTORE SAVED REGISTERS
     043744  012601                    MOV      (SP)+,R1                ; POP STACK INTO R1
4605 043746  103003                    BCC      34$                     ; IF ERROR,
4606 043750  004737  025420            CALL     RETBUF                  ;    RETURN BUFFER AND
4607 043754  000467                    BR       T3L3E                   ;    EXIT
4608 043756  060160  000052   34$:     ADD      R1,D.LBN(RO)            ; SET NEXT LBN
4609 043762  005560  000054            ADC      D.LBN+2(RO)             ;    TO ACCESS
4610 043766  160160  000050            SUB      R1,D.BLKS(RO)           ; DECREMENT BLOCKS REMAINING
4611 043772  001012                    BNE      50$                     ; IF NOT ZERO, KEEP GOING
4612
4613                           ;        SUBTEST DONE ON THIS DRIVE, DROP IT FROM LOOP
4614                           ;
4615 043774  032760  040000  000002 40$:  BIT   #DT.DUN,D.UNIT(RO)      ; IF ALREADY MARKED DONE,
4616 044002  001006                    BNE      50$                     ;    FINISH OTHER DRIVES
4617 044004  052760  040000  000002      BIS   #DT.DUN,D.UNIT(RO)      ; SET DONE BIT
4618 044012  005365  000032            DEC      C.UCNT(R5)              ; DECREMENT UNIT UNDER TEST COUNT
4619 044016  001412                    BEQ      T3S4E                   ; IF UNITS LEFT,DO NEXT UNIT
4620                           ;
4621                           ;        GET NEXT DRIVE IN SEQUENCE
4622                           ;
4623 044020  022765  000003  000042 50$:  CMP   #3.,C.NEXT(R5)          ; IF NOT END
4624 044026  001403                    BEQ      51$                     ;    OF DRIVE LIST,
4625 044030  005265  000042            INC      C.NEXT(R5)              ; THEN, GET NEXT DRIVE &
4626 044034  000710                    BR       T3S4                    ;    BUILD MORE PACKETS
4627 044036  005065  000042   51$:     CLR      C.NEXT(R5)              ; ELSE, RESET NEXT DRIVE POINTER &
4628 044042  000434                    BR       T3L3E                   ;    GO SEND WHAT WE HAVE BUILT
4629                           ;
4630                           ;        END OF STEP. CLEANUP AND GO TO NEXT STEP
4631                           ;
4632 044044  005065  000042   T3S4E:   CLR      C.NEXT(R5)              ; POINT TO FIRST DRIVE
4633 044050  004737  045526   10$:     CALL     GTNXTD                  ; GET POINTER TO NEXT DRIVE TABLE
4634 044054  001423                    BEQ      15$                     ; IF ZERO, DONE
4635 044056  032760  002000  000002      BIT   #DT.EOM,D.UNIT(RO)      ; IF END OF MEDIA,
4636 044064  001010                    BNE      11$                     ;    DON'T COUNT THIS UNIT
4637 044066  042760  040000  000002      BIC   #DT.DUN,D.UNIT(RO)      ; CLEAR "END OF MEDIA" & "DONE"
4638 044074  012760  004000  000050      MOV   #2048.,D.BLKS(RO)       ; SET BLOCKS TO ACCESS
4639 044102  005265  000032            INC      C.UCNT(R5)              ; INCREMENT UNIT UNDER TEST COUNT
4640 044106  022765  000003  000042 11$:  CMP   #3.,C.NEXT(R5)          ; CHECK FOR END OF LIST
4641 044114  001403                    BEQ      15$                     ; IF SO, END CLEANUP
4642 044116  005265  000042            INC      C.NEXT(R5)              ; ELSE,
4643 044122  000752                    BR       10$                     ;    NEXT DRIVE
4644 044124  005065  000042   15$:     CLR      C.NEXT(R5)              ; RESET NEXT DRIVE POINTER
4645 044130  005265  000030            INC      C.STEP(R5)              ; SET STEP
4646 044134                   T3L3E:
4647 044134  000240           NOP                       ; [DBG] INSERT HALT FOR DEBUG
4648 044136  000207                    RETURN                          ; EXIT TEST
4649                           ;
4650                           ;        STEP 5 - "ACCESS" THE REMAINDER OF EACH DISK.
```

```
4651
4652 044140  004737  045526           T3S5:   CALL    GTNXTD                      ; GET POINTER TO DRIVE TABLE
4653 044144  001441                            BEQ     50$                         ; IF ZERO, DONE
4654 044146  032760  140000  000002            BIT     #DT.AVL!DT.DUN,D.UNIT(R0)   ; CHECK IF DRIVE AVAILABLE FOR TEST
4655 044154  001023                            BNE     40$                         ; IF NOT, SKIP SUBTEST
4656                                    ;
4657                                    ;       GET NUMBER OF BLOCKS TO ACCESS AND BYTE COUNT
4658                                    ;
4659 044156  004737  045550                    CALL    GETLBN                      ; GENERATE NEXT LBN TO ACCESS
4660 044162  032760  002000  000002            BIT     #DT.EOM,D.UNIT(R0)          ; IF END OF MEDIA,
4661 044170  001015                            BNE     40$                         ;    EXIT SUBTEST
4662 044172  016001  000050                    MOV     D.BLKS(R0),R1               ; CALCULATE
4663 044176  004737  045502                    CALL    GETBC                       ;    BYTE COUNT
4664                                    ;
4665                                    ;       DO "ACCESS" COMMAND
4666                                    ;
4667 044202  004737  050306                    CALL    ACCESS                      ; BUILD "ACCESS" COMMAND
4668 044206  103457                             BCS     T3L4E                       ;    IF NO PACKETS, EXIT
4669 044210  066060  000050  000052            ADD     D.BLKS(R0),D.LBN(R0)        ; SET NEXT LBN
4670 044216  005560  000054                    ADC     D.LBN+2(R0)                 ;    TO ACCESS
4671 044222  000412                            BR      50$                         ; GET NEXT DRIVE
4672                                    ;
4673                                    ;       SUBTEST DONE ON DRIVE, DROP IT FROM SUBTEST
4674                                    ;
4675 044224  032760  040000  000002    40$:    BIT     #DT.DUN,D.UNIT(R0)          ; IF ALREADY MARKED DONE,
4676 044232  001006                            BNE     50$                         ;    FINISH OTHER DRIVES
4677 044234  052760  040000  000002            BIS     #DT.DUN,D.UNIT(R0)          ; SET DONE BIT
4678 044242  005365  000032                    DEC     C.UCNT(R5)                  ; DECREMENT UNIT UNDER TEST COUNT
4679 044246  001412                            BEQ     T3S5E                       ; IF NO UNITS LEFT, EXIT
4680                                    ;
4681                                    ;       GET NEXT DRIVE IN SEQUENCE
4682                                    ;
4683 044250  022765  000003  000042    50$:    CMP     #3.,C.NEXT(R5)              ; IF NOT END
4684 044256  001403                            BEQ     51$                         ;    OF DRIVE LIST,
4685 044260  005265  000042                    INC     C.NEXT(R5)                  ; THEN, GET NEXT DRIVE &
4686 044264  000725                            BR      T3S5                        ;    BUILD MORE PACKETS
4687 044266  005065  000042    51$:    CLR     C.NEXT(R5)                  ; ELSE, RESET NEXT DRIVE POINTER &
4688 044272  000425                            BR      T3L4E                       ;    GO SEND WHAT WE HAVE BUILT
4689                                    ;
4690                                    ;       END OF STEP. CLEANUP AND GO TO NEXT STEP
4691                                    ;
4692 044274  005065  000042    T3S5E:  CLR     C.NEXT(R5)                  ; POINT TO FIRST DRIVE
4693 044300  004737  045526    10$:    CALL    GTNXTD                      ; GET POINTER TO NEXT DRIVE TABLE
4694 044304  001412                            BEQ     15$                         ; IF ZERO, DONE
4695 044306  042760  042000  000002            BIC     #<DT.EOM+DT.DUN>,D.UNIT(R0) ; CLEAR "END OF MEDIA" & "DONE"
4696 044314  022765  000003  000042            CMP     #3.,C.NEXT(R5)              ; CHECK FOR END OF LIST
4697 044322  001403                            BEQ     15$                         ; IF SO, END CLEANUP
4698 044324  005265  000042                    INC     C.NEXT(R5)                  ; ELSE,
4699 044330  000763                            BR      10$                         ;    NEXT DRIVE
4700 044332  005065  000032    15$:    CLR     C.UCNT(R5)                  ; RESET UNIT COUNT
4701 044336  005065  000042                    CLR     C.NEXT(R5)                  ; RESET NEXT DRIVE POINTER
4702 044342  005265  000030                    INC     C.STEP(R5)                  ; SET STEP
4703 044346                           T3L4E:
4704 044346  000240                    NOP                     ; [DBG] INSERT HALT FOR DEBUG
4705 044350  000207                            RETURN                      ; EXIT TEST
```

```
4706                                    ;
4707                                    ;        WAIT FOR LAST STEP TO COMPLETE
4708                                    ;
4709 044352  005765  000164    T3W4:    TST     C.TBSF(R5)              ; IF TBS LIST ISN'T EMPTY,
4710 044356  001003             BNE     1$                             ;    KEEP WAITING
4711 044360  005765  000160     TST     C.CMFL(R5)                     ; IF CMD LIST IS EMPTY,
4712 044364  001401             BEQ     2$                             ;    START NEXT STEP
4713 044366  000207    1$:      RETURN                                 ;
4714 044370  005265  000030    2$:      INC     C.STEP(R5)             ; DO NEXT STEP
4715 044374  042765  002000  000014     BIC     #CT.DET,C.FLG(R5)      ; INDICATE DETERMINISTIC PHASE DONE
4716 044402                             PRINTF  #DETHDR,(R5)           ; PRINT CONTROLLER INFO MESSAGE
     044402  011546                      MOV     (R5),-(SP)
     044404  012746  010341              MOV     #DETHDR,-(SP)
     044410  012746  000002              MOV     #2,-(SP)
     044414  010600                      MOV     SP,R0
     044416  104417                      TRAP    C$PNTF
     044420  062706  000006              ADD     #6,SP
4717 044424  004737  053550             CALL    RNTIME                 ; GET RUN TIME
4718 044430                             PRINTF  #RNTIM,RNTHH,RNTMM,RNTSS; PRINT RUN TIME
     044430  013746  002272              MOV     RNTSS,-(SP)
     044434  013746  002270              MOV     RNTMM,-(SP)
     044440  013746  002266              MOV     RNTHH,-(SP)
     044444  012746  010303              MOV     #RNTIM,-(SP)
     044450  012746  000004              MOV     #4,-(SP)
     044454  010600                      MOV     SP,R0
     044456  104417                      TRAP    C$PNTF
     044460  062706  000012              ADD     #12,SP
4719 044464                             PRINTF  #CRLF                  ; TERMINATE LINE
     044464  012746  010300              MOV     #CRLF,-(SP)
     044470  012746  000001              MOV     #1,-(SP)
     044474  010600                      MOV     SP,R0
     044476  104417                      TRAP    C$PNTF
     044500  062706  000004              ADD     #4,SP
4720 044504             \               PRINTF  #DETDUN                ; PRINT "DETERMINISTIC PHASE DONE"
     044504  012746  010415              MOV     #DETDUN,-(SP)
     044510  012746  000001              MOV     #1,-(SP)
     044514  010600                      MOV     SP,R0
     044516  104417                      TRAP    C$PNTF
     044520  062706  000004              ADD     #4,SP
4721                                    ;
4722                                    ;        STEP 6 - GENERATE I/O TO THE DRIVES AS PER THE SOFTWARE AND
4723                                    ;        MANUAL INTERVENTION QUESTIONS.
4724                                    ;
4725 044524  005765  000164    T3S6:    TST     C.TBSF(R5)             ; IF TBS LIST IS EMPTY,
4726 044530  001401             BEQ     2$                             ;    BUILD MORE PACKETS
4727 044532  000207             RETURN                                 ;
4728 044534  005765  000032    2$:      TST     C.UCNT(R5)             ; IF MORE DRIVES AVAILABLE,
4729 044540  001035             BNE     20$                            ;    GET NEXT DRIVE
4730                                    ;
4731                                    ;        DO STEP SETUP
4732                                    ;
4733 044542  005065  000042             CLR     C.NEXT(R5)             ; POINT TO FIRST DRIVE
4734 044546  004737  045526    10$:     CALL    GTNXTD                 ; GET POINTER TO NEXT DRIVE TABLE
4735 044552  001421             BEQ     15$                            ; IF ZERO, DONE
4736 044554  005060  000050             CLR     D.BLKS(R0)             ; CLEAR BLOCK SIZE
```

```
4737 044560  016060  000006  000052              MOV    D.BGN1(R0),D.LBN(R0)        ; SET NEXT LBN TO ACCESS TO
4738 044566  016060  000010  000054              MOV    D.BGN1+2(R0),D.LBN+2(R0);     FIRST BLOCK IN FIRST BEGIN/END SET
4739 044574  005265  000032                       INC    C.UCNT(R5)                  ; INCREMENT UNIT UNDER TEST COUNT
4740 044600  022765  000003  000042   11$:        CMP    #3.,C.NEXT(R5)              ; CHECK FOR END OF LIST
4741 044606  001403                                BEQ    15$                         ; IF SO, GET NEXT PATTERN
4742 044610  005265  000042                       INC    C.NEXT(R5)                  ; ELSE,
4743 044614  000754                                BR     10$                         ;   NEXT DRIVE
4744 044616  005065  000042   15$:        CLR    C.NEXT(R5)                  ; INIT DRIVE POINTER
4745 044622  005765  000032                       TST    C.UCNT(R5)                  ; IF MORE DRIVES AVAILABLE,
4746 044626  001002                                BNE    20$                         ;   GET NEXT DRIVE
4747 044630  000137  045310                       JMP    T3S6E                       ; ELSE, EXIT STEP
4748
4749 044634  004737  045526   20$:        CALL   GTNXTD                      ; GET POINTER TO DRIVE TABLE
4750 044640  001002                                BNE    24$                         ; IF NOT ZERO, CONTINUE
4751 044642  000137  045262                       JMP    50$                         ; ELSE, NO MORE DRIVES
4752 044646  032760  140000  000002   24$:        BIT    #DT.AVL!DT.DUN,D.UNIT(R0); CHECK IF DRIVE AVAILABLE
4753 044654  001170                                BNE    40$                         ; IF NOT AVAILABLE, SKIP TEST
4754                                     ;
4755                                     ;         GET SIZE OF TRANSFER
4756                                     ;
4757 044656  013701  002152   30$:        MOV    SFPTBL+SO.MTS,R1            ; GENERATE
4758 044662  032737  000020  002154              BIT    #SM.VLT,SFPTBL+SO.BIT      ;   TRANSFER
4759 044670  001405                                BEQ    31$                         ;     LENGTH
4760 044672  005301                                DEC    R1                          ;       IN
4761 044674  005002                                CLR    R2                          ;         BLOCKS
4762 044676  004737  026250                       CALL   RANDOM                      ;
4763 044702  005201                                INC    R1                          ;
4764 044704  010160  000050   31$:        MOV    R1,D.BLKS(R0)               ;
4765                                     ;
4766                                     ;         GET NEXT LBN TO ACCESS AND DETERMINE I/O OPERATION TYPE
4767                                     ;
4768 044710  004737  045550              CALL   GETLBN                      ; GENERATE NEXT LBN TO ACCESS
4769 044714  032760  010000  000002              BIT    #DT.WRT,D.UNIT(R0)          ; CHECK IF DRIVE WRITEABLE
4770 044722  001513                                BEQ    37$                         ; IF NOT, DO READS ONLY
4771 044724  012701  177777                       MOV    #-1,R1                      ; DO
4772 044730  005002                                CLR    R2                          ;   READ
4773 044732  004737  026250                       CALL   RANDOM                      ;     OR
4774 044736  032701  000010                       BIT    #10,R1                      ;       WRITE?
4775 044742  001503                                BEQ    37$                         ;
4776 044744  005760  000060                       TST    D.CCNT(R0)                  ; IF "COMPARE HOST DATA" OUTSTANDING,
4777 044750  001100                                BNE    37$                         ;   DO "READ"
4778                                     ;
4779                                     ;         GET DATA PATTERN TO WRITE
4780                                     ;
4781 044752  013701  002260              MOV    TS.PAT,R1                   ; GENERATE
4782 044756  001006                                BNE    33$                         ;   PATTERN
4783 044760  012701  000017                       MOV    #15.,R1                     ;     TO
4784 044764  005002                                CLR    R2                          ;       USE
4785 044766  004737  026250                       CALL   RANDOM                      ;
4786 044772  000404                                BR     34$                         ;
4787 044774  022701  000020   33$:        CMP    #16.,R1                     ; IF PATTERN = 16,
4788 045000  001001                                BNE    34$                         ;   USE PATTERN 0
4789 045002  005001                                CLR    R1                          ;
4790 045004  010165  000040   34$:        MOV    R1,C.TPAT(R5)               ;
4791                                     ;
```

```
4792                                    ;           CHECK WHETHER TO DO "WRITE" OR "ERASE" OPERATION
4793                                    ;
4794 045010  016001  000050                MOV     D.BLKS(R0),R1           ; GET NUMBER OF BLOCKS TO WRITE
4795 045014  016503  000040                MOV     C.TPAT(R5),R3           ; GET DATA PATTERN
4796 045020  005760  000056                TST     D.WCNT(R0)             ; IF NO WRITES OUTSTANDING,
4797 045024  001420                         BEQ     35$                    ;   DO "WRITE"
4798 045026  032760  004000  000002         BIT     #DT.CMP,D.UNIT(R0)     ; IF "COMPARE HOST DATA" PENDING,
4799 045034  001046                         BNE     37$                    ;   DO "READ"
4800 045036  005703                         TST     R3                     ; IF DATA PATTERN IS NOT ZERO,
4801 045040  001012                         BNE     35$                    ;   WRITE NEEDED PATTERN
4802 045042  032737  000100  002204         BIT     #IPAT16,IFLAGS         ; IF SET,
4803 045050  001006                         BNE     35$                    ;   DON'T USE ERASE FOR PATTERN 0
4804 045052  004737  045502                 CALL    GETBC                  ; CALCULATE BYTE COUNT
4805 045056  004737  050532                 CALL    ERASE                  ; BUILD "ERASE" COMMAND
4806 045062  103537                          BCS     T3L5E                  ; IF NO PACKETS, EXIT
4807 045064  000456                          BR      39$                    ; ELSE, DO NEXT COMMAND
4808
4809 045066  004737  046474         35$:     CALL    PATBLD                 ; BUILD DATA PATTERN
4810 045072  005701                          TST     R1                     ; IF PATTERN SIZE ZERO,
4811 045074  001444                          BEQ     38$                    ;   DO "ACCESS"
4812 045076  010160  000050                  MOV     R1,D.BLKS(R0)          ; SAVE ACTUAL BLOCK COUNT
4813 045102  032737  000010  002154          BIT     #SM.CMP,SFPTBL+SO.BIT  ; IF NOT SET,
4814 045110  001412                           BEQ     36$                    ;   DON'T DO PATTERN VERIFICATION
4815 045112  012701  000037                  MOV     #31.,R1                ; GIVE PATTERN
4816 045116  005002                          CLR     R2                     ;   VERIFICATION A
4817 045120  004737  026250                  CALL    RANDOM                 ;     1 IN 8 CHANCE
4818 045124  005701                          TST     R1                     ; IF ZERO,
4819 045126  001003                          BNE     36$                    ;   DO PATTERN VERIFICATION
4820 045130  052760  004000  000002          BIS     #DT.CMP,D.UNIT(R0)     ;
4821 045136  004737  050646         36$:     CALL    WRITE                  ; BUILD WRITE COMMAND
4822 045142  103027                          BCC     39$                    ; IF ERROR,
4823 045144  004737  025420                  CALL    RETBUF                 ;   RETURN BUFFER &
4824 045150  000504                          BR      T3L5E                  ;     EXIT
4825
4826                                    ;           DO "READ" OPERATION
4827                                    ;
4828 045152  016001  000050         37$:     MOV     D.BLKS(R0),R1          ; GET INPUT
4829 045156  004737  024354                  CALL    GETBUF                 ;   BUFFER
4830 045162  005701                          TST     R1                     ; IF ZERO,
4831 045164  001410                          BEQ     38$                    ;   DO "ACCESS"
4832 045166  010160  000050                  MOV     R1,D.BLKS(R0)          ; SAVE ACTUAL BLOCK COUNT
4833 045172  004737  050556                  CALL    READ                   ; BUILD READ COMMAND
4834 045176  103011                          BCC     39$                    ; IF ERROR,
4835 045200  004737  025420                  CALL    RETBUF                 ;   RETURN BUFFER AND
4836 045204  000466                          BR      T3L5E                  ;     EXIT
4837
4838                                    ;           DO "ACCESS" IF NO BUFFERS AVAILABLE
4839                                    ;
4840 045206  016001  000050         38$:     MOV     D.BLKS(R0),R1          ; CALCULATE
4841 045212  004737  045502                  CALL    GETBC                  ;   BYTE COUNT
4842 045216  004737  050306                  CALL    ACCESS                 ; BUILD ACCESS COMMAND
4843
4844 045222  066060  000050  000052 39$:     ADD     D.BLKS(R0),D.LBN(R0)   ; SET NEXT LBN
4845 045230  005560  000054                  ADC     D.LBN+2(R0)            ;   TO ACCESS
4846 045234  000412                          BR      50$                    ;
```

```
4847
4848                              ;          DRIVE NOT AVAILABLE, DROP IT FROM SUBTEST
4849
4850 045236 032760 040000 000002 40$:  BIT     #DT.DUN,D.UNIT(R0)     ; IF ALREADY MARKED DONE,
4851 045244 001006                      BNE     50$                    ;   FINISH OTHER DRIVES
4852 045246 052760 040000 000002        BIS     #DT.DUN,D.UNIT(R0)     ; SET DONE BIT
4853 045254 005365 000032               DEC     C.UCNT(R5)             ; DECREMENT UNIT UNDER TEST COUNT
4854 045260 001413                      BEQ     T3S6E                  ; IF NO UNITS LEFT, EXIT
4855                              ;
4856                              ;          GET NEXT DRIVE IN SEQUENCE
4857
4858 045262 022765 000003 000042 50$:  CMP     #3.,C.NEXT(R5)         ; IF NOT END
4859 045270 001404                      BEQ     51$                    ;   OF DRIVE LIST,
4860 045272 005265 000042               INC     C.NEXT(R5)             ; THEN, GET NEXT DRIVE &
4861 045276 000137 044634               JMP     20$                    ;   BUILD MORE PACKETS
4862 045302 005065 000042        51$:  CLR     C.NEXT(R5)             ; ELSE, RESET NEXT DRIVE POINTER &
4863 045306 000425                      BR      T3L5E                  ;   GO SEND WHAT WE HAVE BUILT
4864                              ;
4865                              ;          END OF STEP. CLEANUP AND GO TO NEXT STEP
4866
4867 045310 005065 000042        T3S6E: CLR     C.NEXT(R5)             ; POINT TO FIRST DRIVE
4868 045314 004737 045526        10$:  CALL    GTNXTD                 ; GET POINTER TO NEXT DRIVE TABLE
4869 045320 001412                      BEQ     15$                    ; IF ZERO, DONE
4870 045322 042760 040000 000002        BIC     #DT.DUN,D.UNIT(R0)     ; CLEAR "END OF MEDIA" & "DONE"
4871 045330 022765 000003 000042        CMP     #3.,C.NEXT(R5)         ; CHECK FOR END OF LIST
4872 045336 001403                      BEQ     15$                    ; IF SO, END CLEANUP
4873 045340 005265 000042               INC     C.NEXT(R5)             ; ELSE,
4874 045344 000763                      BR      10$                    ;   NEXT DRIVE
4875 045346 005065 000032        15$:  CLR     C.UCNT(R5)             ; RESET UNIT COUNT
4876 045352 005065 000042               CLR     C.NEXT(R5)             ; RESET NEXT DRIVE POINTER
4877 045356 005265 000030               INC     C.STEP(R5)             ; SET STEP
4878 045362                       T3L5E:
4879 045362 000240               NOP                     ; [DBG] INSERT HALT FOR DEBUG
4880 045364 000207                      RETURN                         ; EXIT TEST
4881
4882                              ;          WAIT FOR LAST STEP TO COMPLETE
4883
4884 045366 005765 000164        T3W5: TST     C.TBSF(R5)             ; IF TBS LIST ISN'T EMPTY,
4885 045372 001003                      BNE     1$                     ;   KEEP WAITING
4886 045374 005765 000160               TST     C.CMFL(R5)             ; IF CMD LIST IS EMPTY,
4887 045400 001401                      BEQ     2$                     ;   START NEXT STEP
4888 045402 000207               1$:  RETURN                          ;
4889 045404 005265 000030        2$:  INC     C.STEP(R5)             ; DO NEXT STEP
4890                              ;
4891                              ;          STEP 7 - SET ALL DRIVES AVAILABLE
4892
4893 045410 004737 045526        T3S7: CALL    GTNXTD                 ; GET POINTER TO DRIVE TABLE
4894 045414 001417                      BEQ     79$                    ;   IF ZERO, DONE
4895 045416 032760 100000 000002        BIT     #DT.AVL,D.UNIT(R0)     ; CHECK IF DRIVE AVAILABLE
4896 045424 001004                      BNE     75$                    ; IF NOT, SKIP SUBTEST
4897 045426 005003                      CLR     R3                     ; CLEAR COMMAND MODIFIERS
4898 045430 004737 050020               CALL    AVAIL                  ; BUILD "AVAILABLE" COMMAND
4899 045434 103420                      BCS     T3L6E                  ;   IF NO PACKETS, EXIT
4900 045436 022765 000003 000042 75$:  CMP     #3.,C.NEXT(R5)         ; CHECK FOR END OF LIST
4901 045444 001403                      BEQ     79$                    ; IF SO, GO TO NEXT SECTION
```

```
4902 045446  005265  000042              INC    C.NEXT(R5)              ; ELSE,
4903 045452  000756                      BR     T3S7                   ;   NEXT DRIVE
4904                              ;
4905                              ;      MARK CONTROLLER AS DONE TESTING
4906                              ;
4907 045454  005065  000032      79$:    CLR    C.UCNT(R5)             ; RESET UNIT UNDER TEST COUNT
4908 045460  005065  000030              CLR    C.STEP(R5)             ; RESET TEST STEP COUNTER
4909 045464  005065  000042              CLR    C.NEXT(R5)             ; NEXT DRIVE IS 1ST IN LIST
4910 045470  052765  040000  000014      BIS    #CT.DUN,C.FLG(R5)      ; SET TEST DONE FLAG
4911 045476                      T3L6E:
4912 045476  000240              NOP                     ; [DBG] INSERT HALT FOR DEBUG
4913 045500  000207              RETURN                  ; EXIT TEST
```

```
4915
4916
4917                        ;++
4918                        ;         GETBC - CONVERT FROM BLOCKS TO BYTES
4919                        ;
4920                        ;         INPUTS:
4921                        ;                 R1 - NUMBER OF BLOCKS
4922                        ;         OUTPUTS:
4923                        ;                 BCLO, BCHI - BYTE COUNT
4924                        ;--
4925 045502 012702 001000  GETBC:  MOV     #512.,R2             ; CALCULATE
4926 045506 004737 047420          CALL    MULT                 ;     COUNT
4927 045512 010337 004406          MOV     R3,BCLO              ;       FOR
4928 045516 010437 004410          MOV     R4,BCHI              ;       I/O
4929 045522 000240          NOP             ; [DBG] INSERT HALT FOR DEBUG
4930 045524 000207          RETURN                              ;
4931
4932                        ;++
4933                        ;         GTNXTD - GET POINTER TO NEXT DRIVE TABLE
4934                        ;
4935                        ;         INPUTS:
4936                        ;                 R5 - ADDRESS OF CONTROLLER TABLE
4937                        ;         OUTPUTS:
4938                        ;                 R0 - POINTER TO DRIVE TABLE
4939                        ;--
4940
4941 045526 016500 000042  GTNXTD: MOV     C.NEXT(R5),R0        ; GET
4942 045512 006300          ASL     R0                          ;    POINTER
4943 045534 060500          ADD     R5,R0                       ;       TO DRIVE
4944 045536 062700 000016   ADD     #C.DR0,R0                    ;        TABLE
4945 045542 011000          MOV     (R0),R0                     ; GET START OF TABLE
4946 045544 000240          NOP             ; [DBG] INSERT HALT FOR DEBUG
4947 045546 000207          RETURN                              ;
```

```
4949
4950
4951                              ;++
4952                              ;           GETLBN - GET LBN TO TEST IF BEGIN/END.SETS ARE USED
4953                              ;
4954                              ;           INPUTS:
4955                              ;                   R0 - POINTER TO DRIVE TABLE
4956                              ;                   R5 - POINTER TO CONTROLLER TABLE
4957                              ;                   D.LBN(R0) - LBN AFTER LBN LAST ACCESSED
4958                              ;                   D.BLKS(R0) - NUMBER OF BLOCKS REQUESTED
4959                              ;                   D.BES(R0) - LAST BEGIN/END SET
4960                              ;           OUTPUTS:
4961                              ;                   D.LBN(R0) - LBN TO ACCESS NEXT
4962                              ;                   D.BLKS(R0) - NUMBER OF BLOCKS TO ACCESS
4963                              ;                   D.BES(R0) - CURRENT BEGIN/END SET
4964                              ;                   DT.EOM FLAG SET IF END OF MEDIA REACHED
4965                              ;--
4966 045550                      GETLBN: PUSH    <R1,R2>                 ; SAVE REGISTERS
4967 045554 032765 002000 000014         BIT     #CT.DET,C.FLG(R5)       ; IF DETERMINISTIC PHASE (BIT SET),
4968 045562 001040                        BNE     20$                     ;   DO SEQUENTIAL ACCESS
4969 045564 032737 000200 002154          BIT     #SM.RAN,SFPTBL+SO.BIT   ; DO SEQUENTIAL
4970 045572 001434                        BEQ     20$                     ;   ACCESS?
4971
4972 045574 016001 000004                 MOV     D.BEC(R0),R1            ; GET COUNT OF
4973 045600 005301                         DEC     R1                      ;   BEGIN/END SETS
4974 045602 005002                         CLR     R2                      ; IF SO,
4975 045604 004737 026250                  CALL    RANDOM                  ;    A BEGIN/END
4976 045610 010104                         MOV     R1,R4
4977 045612 004737 046060                  CALL    GETBES                  ; GET POINTER TO D.BGN?(R0)
4978 045616 016401 000004                  MOV     4(R4),R1                ; CALCULATE
4979 045622 016402 000006                  MOV     6(R4),R2                ;   SIZE
4980 045626 161401                         SUB     (R4),R1                 ;    OF
4981 045630 005602                         SBC     R2                      ;     BEGIN/END
4982 045632 166402 000002                  SUB     2(R4),R2                ;      SET
4983 045636 004737 026250                  CALL    RANDOM                  ; CALCULATE
4984 045642 061401                         ADD     (R4),R1                 ;   NEXT
4985 045644 005502                         ADC     R2                      ;    LBN
4986 045646 066402 000002                  ADD     2(R4),R2
4987 045652 010160 000052                  MOV     R1,D.LBN(R0)            ; SAVE NEXT
4988 045656 010260 000054                  MOV     R2,D.LBN+2(R0)          ;   LBN
4989 045662 000447                         BR      30$                     ;
4990
4991 045664 016004 000046            20$:  MOV     D.BES(R0),R4            ; GET CURRENT BEGIN/END SET
4992 045670 004737 046060                  CALL    GETBES                  ;
4993 045674 026064 000054 000006           CMP     D.LBN+2(R0),6(R4)       ; ARE
4994 045702 103437                         BLO     30$                     ;   WE AT
4995 045704 101004                         BHI     25$                     ;    END OF
4996 045706 026064 000052 000004           CMP     D.LBN(R0),4(R4)         ;     BEGIN/END
4997 045714 101432                         BLOS    30$                     ;      SET?
4998 045716 026060 000046 000004     25$:  CMP     D.BES(R0),D.BEC(R0)     ; GET
4999 045724 103003                         BHIS    26$                     ;   NEXT
5000 045726 005260 000046                  INC     D.BES(R0)               ;    BEGIN/END
5001 045732 000412                         BR      28$                     ;     SET
5002 045734 005060 000046            26$:  CLR     D.BES(R0)               ; RE-INIT CURRENT B/E SET POINTER
5003 045740 032765 002000 000014          BIT     #CT.DET,C.FLG(R5)       ; IF USER DEFINED PHASE (BIT NOT SET),
```

```
5004 045746 001404                              BEG      28$                         ;   GET FIRST LBN IN FIRST B/E SET
5005 045750 052760 002000 000002                BIS      #DT.EOM,D.UNIT(R0)          ; SET END OF MEDIA FLAG
5006 045756 000434                              BR       120$                        ;
5007
5008 045760 016004 000046         28$:          MOV      D.BES(R0),R4                ; GET POINTER
5009 045764 004737 046060                       CALL     GETBES                      ;   TO BEGIN/END SET
5010 045770 011460 000052                       MOV      (R4),D.LBN(R0)              ; SAVE NEXT LBN
5011 045774 016460 000002 000054                MOV      2(R4),D.LBN+2(R0)           ;   TO ACCESS
5012
5013 046002 016401 000004         30$:          MOV      4(R4),R1                    ; GET
5014 046006 016402 000006                       MOV      6(R4),R2                    ;   REMAINING
5015 046012 062701 000001                       ADD      #1,R1                       ;     BEGIN/END
5016 046016 005502                              ADC      R2                          ;       SET
5017 046020 166001 000052                       SUB      D.LBN(R0),R1                ;         LENGTH
5018 046024 005602                              SBC      R2                          ;           IN
5019 046026 166002 000054                       SUB      D.LBN+2(R0),R2              ;             BLOCKS
5020 046032 005702                              TST      R2                          ; MAKE SURE
5021 046034 001005                              BNE      120$                        ;   D.BLKS(R0)
5022 046036 026001 000050                       CMP      D.BLKS(R0),R1               ;     IS NOT GREATER
5023 046042 101402                              BLOS     120$                        ;       THAN REMAINING
5024 046044 010160 000050                       MOV      R1,D.BLKS(R0)               ;         BEGIN/END SET
5025
5026 046050                       120$:         POP      <R2,R1>                     ;
5027 046054 000240                              NOP                   ; [DBG] INSERT HALT FOR DEBUG
5028 046056 000207                              RETURN                               ;
5029
5030                              ;++
5031                              ;       GETBES - GENERATE POINTER TO CURRENT BEGIN/END SET
5032                              ;
5033                              ;       INPUT:
5034                              ;               R4 - BEGIN/END SET INDEX
5035                              ;       OUTPUT:
5036                              ;               R4 - POINTER TO BEGIN/END SET IN DISK TABLE
5037                              ;
5038                              ;--
5039 046060 006304               GETBES: ASL     R4                          ; GET
5040 046062 006304                       ASL     R4                          ;   OFFSET
5041 046064 006304                       ASL     R4                          ;     TO FIRST
5042 046066 060004                       ADD     R0,R4                       ;       LBN IN
5043 046070 062704 000006                ADD     #D.BGM1,R4                  ;         BEGIN/END SET
5044 046074 000240               NOP                   ; [DBG] INSERT HALT FOR DEBUG
5045 046076 000207                       RETURN                               ;
```

```
5047
5048
5049                                        ;**          PATCHK - VERIFY THAT THE DATA BLOCK READ CONTAINS A VALID
5050                                        ;            DATA PATTERN. THIS ROUTINE ASSUMES THE BLOCK READ IS 64KB
5051                                        ;            OR SMALLER.
5052                                        ;
5053                                        ;
5054                                        ;            INPUTS:
5055                                        ;                   C.RHDR(R5) - POINTER TO RESPONSE PACKET
5056                                        ;                   R5 - POINTER TO CONTROLLER TABLE
5057                                        ;            OUTPUTS:
5058                                        ;                   CARRY CLEAR IF DATA PATTERN GOOD
5059                                        ;                   CARRY SET IF DATA PATTERN INVALID
5060                                        ;--
5061  046100                        PATCHK: PUSH     <R0,R1,R2,R3,R4>          ; SAVE REGISTERS
5062  046112  016504  000104                MOV      C.RHDR(R5),R4             ; GET
5063  046116  016401  000006                MOV      P.CRF+RE.RP(R4),R1        ;   POINTER TO
5064  046122  004737  025550                CALL     GETME                    ;     MESSAGE ENVELOPE
5065  046126  005037  004374                CLR      BUFBLK                   ; INIT BLOCK COUNT
5066  046132  005037  004370                CLR      BUFBA                    ; INIT DATA WORD OFFSET
5067  046136  016400  000042                MOV      P.BCNT+ME.CP(R4),R0      ; INITIALIZE
5068  046142  016401  000044                MOV      P.BCNT+2+ME.CP(R4),R1    ;   DATA
5069  046146  000241                        CLC                               ;
5070  046150  006001                        ROR      R1                       ;       BUFFER
5071  046152  006000                        ROR      R0                       ;         WORD
5072  046154  010037  004406                MOV      R0,BCLO                  ;           COUNT
5073  046160  016437  000020  002242         MOV      ME.OFF(R4),TSTOFF        ; GET POINTER TO DATA BUFFER
5074  046166                                PUSH     <R4>                     ; SAVE POINTER TO M.E.
5075
5076  046170  012701  002364         10$:   MOV      #OBUFF,R1                ; GET POINTER TO LOW MEMORY BUFFER
5077  046174  013703  002242                MOV      TSTOFF,R3                ; GET OFFSET TO DATA BUFFER
5078  046200  004737  025154                CALL     MMTOBF                   ; COPY INPUT BUFFER TO LOW MEMORY
5079  046204  011103                        MOV      (R1),R3                  ; GET CURRENT
5080  046206  042703  177760                BIC      #^C<17>,R3               ;   PATTERN NUMBER
5081  046212  006303                        ASL      R3                       ; CONVERT TO WORD OFFSET
5082  046214  016300  006776                MOV      PATDSP(R3),R0            ; GET POINTER TO DATA PATTERN
5083  046220  012702  003364                MOV      #CBUFF,R2                ; POINT TO COMPARE BUFFER AREA
5084  046224  011022                        MOV      (R0),(R2)+               ; PUT PATTERN NUMBER IN BUFFER
5085  046226  012704  000377                MOV      #255.,R4                 ; GET BUFFER SIZE - 1
5086  046232  016003  000002         12$:   MOV      2(R0),R3                 ; GET COUNT OF WORDS IN PATTERN
5087  046236  010001                        MOV      R0,R1                    ; GET POINTER
5088  046240  062701  000004                ADD      #4,R1                    ;   FIRST WORD IN PATTERN
5089  046244  012122                 13$:   MOV      (R1)+,(R2)+              ; WRITE NEXT WORD TO BUFFER
5090  046246  005304                        DEC      R4                       ; DECREMENT BUFFER SIZE COUNT
5091  046250  001403                        BEQ      14$                      ; IF ZERO, GET OUT OF LOOP
5092  046252  005303                        DEC      R3                       ; DECREMENT PATTERN SIZE COUNT
5093  046254  001766                        BEQ      12$                      ; IF ZERO, GO TO TOP OF PATTERN
5094  046256  000772                        BR       13$                      ;   ELSE CONTINUE LOOP
5095  046260  012701  002364         14$:   MOV      #OBUFF,R1                ; GET POINTER TO BLOCK READ
5096  046264  012702  003364                MOV      #CBUFF,R2                ; GET POINTER TO COMPARE BUFFER
5097  046270  005003                        CLR      R3                       ; START AT FIRST WORD
5098  046272  000241                        CLC                               ;
5099  046274  006037  004406                ROR      BCLO                     ; CONVERT TO WORDS
5100  046300  012700  000400                MOV      #256.,R0                 ; CALCULATE
5101  046304  020037  004406                CMP      R0,BCLO                  ;   NUMBER
```

```
5102 046310  101402                   BLOS    15$                       ;       OF WORDS
5103 046312  013700  004406           MOV     BCLO,RO                   ;         TO CHECK
5104 046316  022122           15$:    CMP     (R1)+,(R2)+               ; COMPARE
5105 046320  001024                   BNE     30$                       ;   BUFFER
5106 046322  005203                   INC     R3                        ;     CONTENTS
5107 046324  020003                   CMP     RO,R3                     ; IF NOT AT END,
5108 046326  003373                   BGT     15$                       ;   CHECK NEXT WORD
5109 046330  005237  004374           INC     BUFBLK                    ; COUNT BLOCKS CHECKED
5110 046334  006303                   ASL     R3                        ; CALCULATE NEXT
5111 046336  060337  002242           ADD     R3,TSTOFF                 ;   MEMORY OFFSET
5112 046342  160037  004406           SUB     RO,BCLO                   ; IF NOT ZERO,
5113 046346  001310                   BNE     10$                       ;   GET NEXT BUFFER
5114 046350                           POP     <R4>                      ; RESTORE SAVED REGISTERS
5115 046352  000241                   CLC                               ; CLEAR CARRY TO INDICATE SUCCESS
5116 046354           20$:            POP     <R4,R3,R2,R1,RO>          ; RESTORE SAVED REGISTERS
5117 046366  000240                   NOP                               ; [DBG] INSERT HALT FOR DEBUG
5118 046370  000207                   RETURN                            ;
5119
5120 046372           30$:            POP     <R4>                      ; RESTORE POINTER TO M.E.
5121 046374  016437  000062  004370   MOV     P.LBN+ME.CP(R4),BUFBA     ; GET
5122 046402  016437  000064  004372   MOV     P.LBN+2+ME.CP(R4),BUFEA   ;   LBN
5123 046410  063737  004374  004370   ADD     BUFBLK,BUFBA              ;     CONTAINING BAD
5124 046416  005537  004372           ADC     BUFEA                     ;       DATA WORD
5125 046422  006303                   ASL     R3                        ; CONVERT TO BYTE OFFSET
5126 046424  010337  004406           MOV     R3,BCLO                   ;
5127 046430  016237  177776  004364   MOV     -2(R2),PCGOOD             ; POINT TO
5128 046436  016137  177776  004366   MOV     -2(R1),PCBAD              ;   FAILING DATA
5129 046444  016501  000104           MOV     C.RHDR(R5),R1             ; CALCULATE ADDRESS
5130 046450  062701  000012           ADD     @<P.UNIT+RE.RP>,R1        ;   CONTAINING UNIT NUMBER
5131 046454                           ERRHRD  38.,ERRO38                ; PRINT DATA COMPARE ERROR
     046454  104456                   TRAP    C$ERHRD
     046456  000046                   .WORD   38
     046460  000000                   .WORD   0
     046462  020576                   .WORD   ERRO38
5132 046464  004737  026006           CALL    RTNME                     ; RETURN M.E. & CLEAR RESPONSE RING
5133 046470  000261                   SEC                               ; SET CARRY TO INDICATE ERROR
5134 046472  000730                   BR      20$                       ;
```

```
5136
5137
5138        ;++           PATBLD - BUILD A MEMORY BUFFER CONTAINING A SPECIFIED DATA
5139        ;             PATTERN. IF A MEMORY SEGMENT CONTAINING THE NEEDED PATTERN
5140        ;             EXISTS IN THE MULTI-USE TABLE AND ITS SIZE IS BIG ENOUGH, USE
5141        ;             IT. OTHERWISE ALLOCATE A NEW MEMORY SEGMENT.
5142        ;
5143        ;             INPUTS:
5144        ;                 R1 - NUMBER OF BLOCKS NEEDED
5145        ;                 R3 - THE PATTERN NUMBER
5146        ;             OUTPUTS:
5147        ;                 R1 - BLOCKS ALLOCATED
5148        ;                 R3 - MAPPED MEMORY OFFSET
5149        ;                 BCLO,BCHI - BYTE COUNT
5150        ;                 BUFBA,BUFEA - PHYSICAL ADDRESS OF BUFFER
5151        ;                 TSTOFF - MEMORY OFFSET TO BUFFER
5152        ;--
5153
5154 046474                      PATBLD: PUSH   <R0,R2,R4,R5>        ; SAVE REGISTERS
     046474 010046                       MOV    R0,-(SP)            ; PUSH R0 ON STACK
     046476 010246                       MOV    R2,-(SP)            ; PUSH R2 ON STACK
     046500 010446                       MOV    R4,-(SP)            ; PUSH R4 ON STACK
     046502 010546                       MOV    R5,-(SP)            ; PUSH R5 ON STACK
5155 046504 042737 000040 002204         BIC    #ITEMP,IFLAGS       ; CLEAR TEMPORARY FLAG BIT
5156 046512 032765 000400 000014         BIT    #CT.MRW,C.FLG(R5)   ; IF MAINTENANCE OPERATION IN
5157 046520 001403                       BEQ    1$                  ;   SET TEMPORARY BIT TO
5158 046522 052737 000040 002204         BIS    #ITEMP,IFLAGS       ;     INDICATE MAINTENANCE MODE
5159 046530 006303              1$:      ASL    R3                  ; CONVERT TO WORD OFFSET
5160 046532 010305                       MOV    R3,R5               ; SAVE PATTERN OFFSET
5161 046534 026501 004516                CMP    MUTSIZ(R5),R1       ; IF SIZE NEEDED > EXISTING SIZE
5162 046540 103423                       BLO    10$                 ;   ALLOCATE A NEW SEGMENT
5163 046542 016503 004412      2$:      MOV    MUTOFF(R5),R3       ; ELSE, USE CURRENT SEGMENT
5164 046546 013704 004622               MOV    M.HDR,R4            ; FIND PACKET IN MEMORY LIST
5165 046552 001004              4$:      BNE    6$                  ; IF NOT END OF LIST, KEEP LOOKING
5166 046554 012701 046552               MOV    #.-2,R1             ; ELSE, THE PACKET WASN'T FOUND
5167 046560 000137 024310               JMP    BADMEM             ;     PRINT SYSTEM FATAL ERROR
5168 046564 026403 000004      6$:      CMP    4(R4),R3           ; IF MEMORY OFFSETS ARE SAME,
5169 046570 001402                       BEQ    8$                  ;   SEARCH DONE
5170 046572 011404                       MOV    (R4),R4             ; ELSE,
5171 046574 000766                       BR     4$                  ;   POINT TO NEXT PACKET
5172 046576 105264 000003      8$:      INCB   3(R4)               ; INCREMENT NUMBER OF USERS
5173 046602 004737 025006               CALL   MAPMEM             ; GET BYTE COUNT & PHYSICAL ADDRESS
5174 046606 000556                       BR     50$                 ;   EXIT
5175
5176 046610 004737 024354      10$:     CALL   GETBUF             ; GET NEEDED MEMORY
5177 046614 005701                       TST    R1                  ; IF RETURNED SIZE ZERO,
5178 046616 001407                       BEQ    11$                 ;   USE EXISTING BUFFER
5179 046620 026501 004516                CMP    MUTSIZ(R5),R1       ; IF EXISTING SIZE < PACKET RETURNED
5180 046624 103410                       BLO    20$                 ;   BUILD NEW PACKET
5181 046626 010337 002242                MOV    R3,TSTOFF           ; ELSE,
5182 046632 004737 025420                CALL   RETBUF             ;   RETURN UNEEDED BUFFER
5183 046636 016501 004516      11$:     MOV    MUTSIZ(R5),R1       ; IF EXISTING BUFFER SIZE NOT ZERO,
5184 046642 001337                       BNE    2$                  ;   USE EXISTING BUFFER
5185 046644 000537                       BR     50$                 ; ELSE, EXIT ROUTINE
5186
```

```
5187 046646                        20$:    PUSH    <R3.R1>             ; SAVE BLOCK COUNT & MEMORY OFFSET
     046646  010346                        MOV     R3,-(SP)            ; PUSH R3 ON STACK
     046650  010146                        MOV     R1,-(SP)            ; PUSH R1 ON STACK
5188 046652  010165  004516                MOV     R1,MUTSIZ(R5)       ; PUT NEW BUFFER IN
5189 046656  010365  004412                MOV     R3,MUTOFF(R5)       ;   MULTI-USE LIST
5190 046662  032737  000040  002204        BIT     #ITEMP,IFLAGS       ; IF MAINTENANCE OPERATION IN
5191 046670  001101                        BNE     40$                 ;   PROGRESS, USE SPECIAL PATTERNS
5192 046672  022705  000042                CMP     #<17.*2>,R5         ; IF PATTERN = 17,
5193 046676  001433                        BEQ     30$                 ;   PUT ALL PATTERNS IN BUFFER
5194 046700  016500  006776                MOV     PATDSP(R5),R0       ; GET POINTER TO DATA PATTERN
5195 046704  012702  002364                MOV     #OBUFF,R2           ; POINT TO OUTPUT BUFFER AREA
5196 046710  012022                        MOV     (R0)+,(R2)+         ; PUT PATTERN NUMBER IN BUFFER
5197 046712  012704  000377                MOV     #255.,R4            ; GET BUFFER SIZE - 1
5198 046716  010001                22$:    MOV     R0,R1               ; GET POINTER TO FIRST WORD IN PATTERN
5199 046720  012105                        MOV     (R1)+,R5            ; GET COUNT OF WORDS IN PATTERN
5200 046722  012122                23$:    MOV     (R1)+,(R2)+         ; WRITE NEXT WORD TO BUFFER
5201 046724  005304                        DEC     R4                  ; DECREMENT BUFFER SIZE COUNT
5202 046726  001403                        BEQ     24$                 ; IF ZERO, GET OUT OF LOOP
5203 046730  005305                        DEC     R5                  ; DECREMENT PATTERN SIZE COUNT
5204 046732  001771                        BEQ     22$                 ; IF ZERO, GO TO TOP OF PATTERN
5205 046734  000772                        BR      23$                 ;     ELSE CONTINUE LOOP
5206 046736  011602                24$:    MOV     (SP),R2             ; CREATE LOOP COUNTER
5207 046740  012701  002364                MOV     #OBUFF,R1           ; WRITE OUTPUT BUFFER TO
5208 046744  004737  025276        25$:    CALL    BFTOMM              ;   MAPPED MEMORY BUFFER
5209 046750  062703  001000                ADD     #512.,R3            ; INCREMENT OFFSET
5210 046754  005302                        DEC     R2                  ; DECREMENT LOOP COUNT
5211 046756  001372                        BNE     25$                 ; IF NOT ZERO, LOOP
5212 046760                                POP     <R1,R3>             ; RESTORE REGISTERS
     046760  012601                        MOV     (SP)+,R1            ; POP STACK INTO R1
     046762  012603                        MOV     (SP)+,R3            ; POP STACK INTO R3
5213 046764  000467                        BR      50$                 ; EXIT ROUTINE
5214
5215 046766  010102                30$:    MOV     R1,R2               ; CREATE LOOP COUNTER
5216 046770  012705  000036        31$:    MOV     #<15.*2>,R5         ; POINT TO DATA PATTERN 15
5217 046774                        32$:    PUSH    <R2>                ; SAVE LOOP COUNTER
     046774  010246                        MOV     R2,-(SP)            ; PUSH R2 ON STACK
5218 046776  016500  006776                MOV     PATDSP(R5),R0       ; GET POINTER TO DATA PATTERN
5219 047002  012702  002364                MOV     #OBUFF,R2           ; POINT TO OUTPUT BUFFER AREA
5220 047006  012022                        MOV     (R0)+,(R2)+         ; PUT PATTERN NUMBER IN BUFFER
5221 047010  012704  000377                MOV     #255.,R4            ; GET BUFFER SIZE - 1
5222 047014  010001                33$:    MOV     R0,R1               ; GET POINTER TO FIRST WORD IN PATTERN
5223 047016  012105                        MOV     (R1)+,R5            ; GET COUNT OF WORDS IN PATTERN
5224 047020  012122                34$:    MOV     (R1)+,(R2)+         ; WRITE NEXT WORD TO BUFFER
5225 047022  005304                        DEC     R4                  ; DECREMENT BUFFER SIZE COUNT
5226 047024  001403                        BEQ     35$                 ; IF ZERO, GET OUT OF LOOP
5227 047026  005305                        DEC     R5                  ; DECREMENT PATTERN SIZE COUNT
5228 047030  001771                        BEQ     33$                 ; IF ZERO, GO TO TOP OF PATTERN
5229 047032  000772                        BR      34$                 ;     ELSE CONTINUE LOOP
5230 047034  012701  002364        35$:    MOV     #OBUFF,R1           ; WRITE OUTPUT BUFFER TO
5231 047040  004737  025276                CALL    BFTOMM              ;   MAPPED MEMORY BUFFER
5232 047044  062703  001000                ADD     #512.,R3            ; INCREMENT OFFSET
5233 047050                                POP     <R2>                ; RESTORE LOOP COUNT
     047050  012602                        MOV     (SP)+,R2            ; POP STACK INTO R2
5234 047052  005302                        DEC     R2                  ; DECREMENT LOOP COUNT
5235 047054  001404                        BEQ     39$                 ; IF ZERO, EXIT LOOP
```

```
5236 047056  005305                    DEC    R5              ; ELSE
5237 047060  005305                    DEC    R5              ;   POINT TO NEXT PATTERN
5238 047062  002344                    BGE    32$             ;   IF LESS THAN ZERO,
5239 047064  000741                    BR     31$             ;     START WITH PATTERN 15 AGAIN
5240 047066               39$:         POP    <R1,R3>         ; RESTORE REGISTERS
     047066  012601                    MOV    (SP)+,R1        ; POP STACK INTO R1
     047070  012603                    MOV    (SP)+,R3        ; POP STACK INTO R3
5241 047072  000424                    BR     50$             ;
5242
5243 047074  016500  007036  40$:      MOV    MNTPAT(R5),R0   ; GET DATA PATTERN
5244 047100  012702  002364            MOV    #OBUFF,R2       ; POINT TO OUTPUT BUFFER AREA
5245 047104  012704  000400            MOV    #256.,R4        ; GET BUFFER SIZE
5246 047110  010022            43$:    MOV    R0,(R2)+        ; WRITE NEXT WORD TO BUFFER
5247 047112  005304                    DEC    R4              ; DECREMENT BUFFER SIZE COUNT
5248 047114  001375                    BNE    43$             ; IF ZERO, GET OUT OF LOOP
5249 047116  011602                    MOV    (SP),R2         ; CREATE LOOP COUNTER
5250 047120  012701  002364            MOV    #OBUFF,R1       ; WRITE OUTPUT BUFFER TO
5251 047124  004737  025276  45$:      CALL   BFTOMM          ;   MAPPED MEMORY BUFFER
5252 047130  062703  001000            ADD    #512.,R3        ; INCREMENT OFFSET
5253 047134  005302                    DEC    R2              ; DECREMENT LOOP COUNT
5254 047136  001372                    BNE    45$             ; IF NOT ZERO, LOOP
5255 047140                            POP    <R1,R3>         ; RESTORE REGISTERS
     047140  012601                    MOV    (SP)+,R1        ; POP STACK INTO R1
     047142  012603                    MOV    (SP)+,R3        ; POP STACK INTO R3
5256
5257 047144               50$:         POP    <R5,R4,R2,R0>   ;   REGISTERS
     047144  012605                    MOV    (SP)+,R5        ; POP STACK INTO R5
     047146  012604                    MOV    (SP)+,R4        ; POP STACK INTO R4
     047150  012602                    MOV    (SP)+,R2        ; POP STACK INTO R2
     047152  012600                    MOV    (SP)+,R0        ; POP STACK INTO R0
5258 047154  010337  002242            MOV    R3,TSTOFF       ; SAVE MEMORY OFFSET
5259 047160  000240               NOP                         ; [DBG] INSERT HALT FOR DEBUG
5260 047162  000207                    RETURN                 ;
5261
```

```
5263
5264
5265                            ;++
5266                            ;          GTDRVT - GET DRIVE TABLE POINTER
5267                            ;
5268                            ;          INPUTS:
5269                            ;                  R1 - DRIVE NUMBER
5270                            ;                  R5 - CONTROLLER TABLE ADDRESS
5271                            ;          OUTPUTS:
5272                            ;                  R1 - DRIVE TABLE ADDRESS
5273                            ;                  L$LUN - LOADED WITH UNIT NUMBER OF DRIVE
5274                            ;                  CARRY CLEAR IF DRIVE TABLE NOT FOUND AFTER ERROR PRINTED
5275                            ;--
5276 047164                    GTDRVT: PUSH   <R2,R4>               ; SAVE CONTENTS OF REGISTERS
     047164  010246                    MOV    R2,-(SP)              ; PUSH R2 ON STACK
     047166  010446                    MOV    R4,-(SP)              ; PUSH R4 ON STACK
5277 047170  010504                    MOV    R5,R4                 ; GET CONTROLLER TABLE ADRESS
5278 047172  062704  000016            ADD    #C.DRO,R4             ; ADD OFFSET TO DRIVE TABLE ADDRESS
5279 047176  012702  000004            MOV    #4.,R2                ; GET COUNT OF DRIVES
5280 047202  005714            1$:     TST    (R4)                  ; CHECK IF AN ADDRESS HERE
5281 047204  001406                    BEQ    3$                    ; IF NO ADDRESS, ERROR EXIT
5282 047206  027401  000000            CMP    @(R4),R1              ; COMPARE DRIVE NUMBERS
5283 047212  001410                    BEQ    4$                    ; BRANCH IF A MATCH
5284 047214  005724            2$:     TST    (R4)+                 ; BUMP ADDRESS
5285 047216  005302                    DEC    R2                    ; DECREMENT DRIVE COUNTER
5286 047220  001370                    BNE    1$                    ; LOOK AT NEXT TABLE
5287 047222                    3$:     POP    <R4,R2>               ; RESTORE REGISTERS
     047222  012604                    MOV    (SP)+,R4              ; POP STACK INTO R4
     047224  012602                    MOV    (SP)+,R2              ; POP STACK INTO R2
5288 047226  000261                    SEC                          ; SET CARRY AS ERROR FLAG
5289 047230  000240            NOP                                  ; [DBG] INSERT HALT FOR DEBUG
5290 047232  000207                    RETURN                       ;
5291
5292 047234  011401            4$:     MOV    (R4),R1               ; GET ADDRESS OF TABLE
5293 047236  116137  000002  002074    MOVB   D.UNIT(R1),L$LUN      ; GET LOGICAL UNIT NUMBER
5294 047244                            POP    <R4,R2>               ; RESTORE REGISTERS
     047244  012604                    MOV    (SP)+,R4              ; POP STACK INTO R4
     047246  012602                    MOV    (SP)+,R2              ; POP STACK INTO R2
5295 047250  000241                    CLC                          ; CLEAR CARRY FLAG
5296 047252  000240            NOP                                  ; [DBG] INSERT HALT FOR DEBUG
5297 047254  000207                    RETURN                       ;
```

```
5299
5300
5301                                    ;++
5302                                    ;          DIVIDE - DIVIDE A 48 BIT UNSIGNED NUMBER BY A 16 BIT NUMBER.
5303                                    ;          REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
5304                                    ;          WILL NOT CHECK FOR DIVIDE BY ZERO.
5305                                    ;
5306                                    ;          INPUTS:
5307                                    ;                    R1 - LOW  16 BITS OF DIVIDEND
5308                                    ;                    R2 - NEXT 16 BITS OF DIVIDEND
5309                                    ;                    R3 - HIGH 16 BITS OF DIVIDEND
5310                                    ;                    R4 - DIVISOR
5311                                    ;          OUTPUTS:
5312                                    ;                    R1 - LOW 16 BITS OF QUOTIENT
5313                                    ;                    R2 - NEXT 16 BITS OF QUOTIENT
5314                                    ;                    R3 - HIGH 16 BITS OF QUOTIENT
5315                                    ;                    R4 - REMAINDER
5316                                    ;--
5317  047256                 DIVIDE:   PUSH      <R0,R5>                 ; DIVIDEND IS IN <R3,R2,R1>
       047256   010046                 MOV       R0,-(SP)                ; PUSH R0 ON STACK
       047260   010546                 MOV       R5,-(SP)                ; PUSH R5 ON STACK
5318  047262   010405                 MOV       R4,R5                   ; SAVE DIVISOR
5319  047264   012700   000060        MOV       #48.,R0                 ; SET UP SHIFT COUNT
5320  047270   005004                 CLR       R4                      ; START WITH ZERO REMAINDER
5321  047272   006301       1$:       ASL       R1                      ;
5322  047274   006102                 ROL       R2                      ; SHIFT LEFT INTO REMAINDER
5323  047276   006103                 ROL       R3                      ;
5324  047300   006104                 ROL       R4                      ;
5325  047302   020504                 CMP       R5,R4                   ; WILL DIVISOR GO INTO REMAINDER?
5326  047304   101002                 BHI       2$                      ; ONLY SUBTRACT IF IT WILL
5327  047306   160504                 SUB       R5,R4                   ; SUBTRACT DIVISOR
5328  047310   005201                 INC       R1                      ; PUT A ONE INTO QUOTIENT
5329  047312   005300       2$:       DEC       R0                      ; COUNT THE SHIFTS
5330  047314   001366                 BNE       1$                      ;
5331  047316                          POP       <R5,R0>                 ; RESTORE SAVED REGISTERS
       047316   012605                 MOV       (SP)+,R5                ; POP STACK INTO R5
       047320   012600                 MOV       (SP)+,R0                ; POP STACK INTO R0
5332  047322   000240                 NOP                         ; [DBG] INSERT HALT FOR DEBUG
5333  047324   000207                 RETURN                          ;
```

```
5335
5336
5337                            ;++
5338                            ;        DIV48 - DIVIDE A 48 BIT UNSIGNED NUMBER BY A 32 BIT UNSIGNED NUMBER.
5339                            ;        REPLACE DIVIDEND WITH QUOTIENT AND RETURN REMAINDER.
5340                            ;        WILL NOT CHECK FOR DIVIDE BY ZERO.
5341                            ;
5342                            ;        INPUTS:
5343                            ;                R1 - LOW  16 BITS OF DIVIDEND
5344                            ;                R2 - MID 16 BITS OF DIVIDEND
5345                            ;                R3 - HIGH 16 BITS OF DIVIDEND
5346                            ;                R4 - LOW 16 BITS OF DIVISOR
5347                            ;                R5 - HIGH 16 BITS OF DIVISOR
5348                            ;        OUTPUTS:
5349                            ;                R1 - LOW 16 BITS OF QUOTIENT
5350                            ;                R2 - MID 16 BITS OF QUOTIENT
5351                            ;                R3 - HIGH 16 BITS OF QUOTIENT
5352                            ;                R4 - LOW 16 BITS OF REMAINDER
5353                            ;                R5 - HIGH 16 BITS OF REMAINDER
5354                            ;--
5355 047326            DIV48:  PUSH    <R0>                    ; SAVE REGISTERS
     047326 010046             MOV     R0,-(SP)                ; PUSH R0 ON STACK
5356 047330            PUSH    <R5,R4>                 ; SAVE DIVISOR
     047330 010546             MOV     R5,-(SP)                ; PUSH R5 ON STACK
     047332 010446             MOV     R4,-(SP)                ; PUSH R4 ON STACK
5357 047334 012700 000060      MOV     #48.,R0                 ; SET UP SHIFT COUNT
5358 047340 005004             CLR     R4                      ; START WITH
5359 047342 005005             CLR     R5                      ;    ZERO REMAINDER
5360 047344 006301     1$:     ASL     R1
5361 047346 006102             ROL     R2                      ; SHIFT LEFT INTO REMAINDER
5362 047350 006103             ROL     R3                      ;
5363 047352 006104             ROL     R4                      ;
5364 047354 006105             ROL     R5                      ;
5365 047356 026605 000002      CMP     2(SP),R5                ; IS
5366 047362 101010             BHI     10$                     ;    DIVISOR
5367 047364 103402             BLO     2$                      ;      LESS THAN
5368 047366 021604             CMP     (SP),R4                 ;        OR EQUAL TO
5369 047370 101005             BHI     10$                     ;          REMAINDER?
5370 047372 161604     2$:     SUB     (SP),R4                 ; IF SO,
5371 047374 005605             SBC     R5                      ;    SUBTRACT DIVISOR
5372 047376 166605 000002      SUB     2(SP),R5                ;      FROM REMAINDER
5373 047402 005201             INC     R1                      ;        & INCREMENT QUOTIENT
5374 047404 005300     10$:    DEC     R0                      ; COUNT THE SHIFTS
5375 047406 001356             BNE     1$
5376 047410 022626             CMP     (SP)+,(SP)+             ; THROW OUT SAVED DIVISOR
5377 047412            POP     <R0>                    ; RESTORE SAVED REGISTERS
     047412 012600             MOV     (SP)+,R0                ; POP STACK INTO R0
5378 047414 000240     NOP                             ; [DBG] INSERT HALT FOR DEBUG
5379 047416 000207             RETURN                          ;
```

```
5381
5382
5383              ;++
5384              ;          MULT - MULTIPLY A 16 BIT NUMBER BY ANOTHER 16 BIT NUMBER, YIELDING
5385              ;          A 32 BIT RESULT
5386              ;
5387              ;          INPUTS:
5388              ;                  R1 - MULTIPLIER
5389              ;                  R2 - MULTIPLICAND
5390              ;
5391              ;          OUTPUTS:
5392              ;                  R1 - ZERO
5393              ;                  R2 - MULTIPLICAND
5394              ;                  R3 - PRODUCT (LOW 16 BITS)
5395              ;                  R4 - PRODUCT (HIGH 16 BITS)
5396              ;--
5397 047420      MULT:     PUSH    <R0,R1,R2>              ; SAVE REGISTERS
     047420 010046         MOV     R0,-(SP)                ; PUSH R0 ON STACK
     047422 010146         MOV     R1,-(SP)                ; PUSH R1 ON STACK
     047424 010246         MOV     R2,-(SP)                ; PUSH R2 ON STACK
5398 047426 005000         CLR     R0
5399 047430 005003         CLR     R3                      ; CLEAR
5400 047432 005004         CLR     R4                      ;     PRODUCT
5401 047434 000241  1$:    CLC                             ;
5402 047436 006001         ROR     R1                      ; SHIFT MULTIPLIER TO RIGHT
5403 047440 103003         BCC     2$                      ; IF A ONE BIT SHIFTED OUT
5404 047442 060203         ADD     R2,R3                   ;     ADD MULTIPLICAND
5405 047444 005504         ADC     R4                      ;         TO PRODUCT
5406 047446 060004         ADD     R0,R4                   ;
5407 047450 006302  2$:    ASL     R2                      ; DOUBLE THE
5408 047452 006100         ROL     R0                      ;     MULTIPLICAND
5409 047454 005701         TST     R1                      ;
5410 047456 001366         BNE     1$                      ; CONTINUE UNTIL MULTIPLIER IS ZERO
5411 047460         POP     <R2,R1,R0>              ; RESTORE REGISTERS
     047460 012602         MOV     (SP)+,R2                ; POP STACK INTO R2
     047462 012601         MOV     (SP)+,R1                ; POP STACK INTO R1
     047464 012600         MOV     (SP)+,R0                ; POP STACK INTO R0
5412 047466 000240  NOP                            ; [DBG] INSERT HALT FOR DEBUG
5413 047470 000207         RETURN                          ;
5414
```

```
5416
5417
5418                                    ;++
5419                                    ;         ITOA - CONVERT A 48 BIT OR SMALLER BINARY NUMBER INTO A DECIMAL
5420                                    ;         ASCIZ STRING.
5421                                    ;
5422                                    ;         INPUTS:
5423                                    ;                  R1 - LOW 16 BITS OF NUMBER
5424                                    ;                  R2 - NEXT 16 BITS OF NUMBER
5425                                    ;                  R3 - HIGH 16 BITS OF NUMBER
5426                                    ;                  R4 - CHARACTER COUNT
5427                                    ;
5428                                    ;         OUTPUTS:
5429                                    ;                  TEMP - FIRST CHARACTER IN A ASCIZ OUTPUT STRING
5430                                    ;                  R1,R2,R3 - DESTROYED
5431                                    ;--
5432  047472                ITOA:    PUSH    <R0,R4>              ; SAVE REGISTERS
      047472 010046                  MOV     R0,-(SP)             ; PUSH R0 ON STACK
      047474 010446                  MOV     R4,-(SP)             ; PUSH R4 ON STACK
5433  047476 012700 002212           MOV     #TEMP,R0             ; GET POINTER TO STRING
5434  047502 112720 000060   1$:     MOVB    #'0,(R0)+            ; FILL ENTIRE
5435  047506 005304                  DEC     R4                   ;       STRING WITH
5436  047510 001374                  BNE     1$                   ;            SPACES
5437  047512 105010                  CLRB    (R0)                 ; TERMINATE STRING
5438  047514 012704 000012   2$:     MOV     #10.,R4              ; GET DIVISOR
5439  047520 004737 047256           CALL    DIVIDE               ; DIVIDE NUMBER BY 10
5440  047524 062704 000060           ADD     #'0,R4               ; CONVERT REMAINDER TO ASCII
5441  047530 110440                  MOVB    R4,-(R0)             ; STORE CHARACTER IN STRING
5442  047532 010146                  MOV     R1,-(SP)             ; CHECK
5443  047534 050216                  BIS     R2,(SP)              ;       IF QUOTIENT
5444  047536 050326                  BIS     R3,(SP)+             ;             IS ZERO
5445  047540 001365                  BNE     2$                   ; IF NOT, GET NEXT CHARACTER
5446  047542                POP     <R4,R0>              ; RESTORE SAVED REGISTERS
      047542 012604                  MOV     (SP)+,R4             ; POP STACK INTO R4
      047544 012600                  MOV     (SP)+,R0             ; POP STACK INTO R0
5447  047546 000240        NOP       RETURN           ; [DBG] INSERT HALT FOR DEBUG
5448  047550 000207                  RETURN                       ;
5449
```

```
5451
5452                              ;++
5453                              ;          BLDSTR - CONVERT 32 BIT NUMBER TO 10 CHARACTER ASCIZ STRING
5454                              ;
5455                              ;          INPUT:
5456                              ;                   R4 - POINTER TO 2 WORD DEFAULT NUMBER
5457                              ;          OUTPUT:
5458                              ;                   TEMP - ASCIZ STRING REPRESENTING DEFAULT NUMBER
5459                              ;--
5460
5461 047552                      BLDSTR: PUSH    <R1,R2,R3,R4>          ; SAVE REGISTERS
     047552   010146                     MOV     R1,-(SP)              ; PUSH R1 ON STACK
     047554   010246                     MOV     R2,-(SP)              ; PUSH R2 ON STACK
     047556   010346                     MOV     R3,-(SP)              ; PUSH R3 ON STACK
     047560   010446                     MOV     R4,-(SP)              ; PUSH R4 ON STACK
5462 047562   011401                     MOV     (R4),R1               ; GET LOW WORD OF NUMBER
5463 047564   016402   000002            MOV     2(R4),R2              ; GET NEXT WORD OF NUMBER
5464 047570   005003                     CLR     R3                    ; CLEAR HIGH WORD
5465 047572   012704   000012            MOV     #10.,R4               ; GET CHARACTER COUNT
5466 047576   004737   047472            CALL    ITOA                  ; CONVERT INTEGER TO ASCII
5467 047602                              POP     <R4,R3,R2,R1>         ; RESTORE SAVED REGISTERS
     047602   012604                     MOV     (SP)+,R4              ; POP STACK INTO R4
     047604   012603                     MOV     (SP)+,R3              ; POP STACK INTO R3
     047606   012602                     MOV     (SP)+,R2              ; POP STACK INTO R2
     047610   012601                     MOV     (SP)+,R1              ; POP STACK INTO R1
5468 047612   000240            NOP                                   ; [DBG] INSERT HALT FOR DEBUG
5469 047614   000207            RETURN                                ;
```

```
5471
5472                                    ;++
5473                                    ;        BLDLWD - CONVERT ASCIZ STRING TO 32-BIT NUMBER
5474                                    ;
5475                                    ;        INPUTS:
5476                                    ;              TEMP - ASCIZ STRING UP TO 10 CHARACTERS LONG
5477                                    ;              R4 - ADDRESS OF TWO WORD STORAGE
5478                                    ;        OUTPUTS:
5479                                    ;              IF STRING IS VALID NUMBER
5480                                    ;                 TWO WORDS AT R4 LOADED WITH NUMBER
5481                                    ;                 R4 POINTING TO WORD AFTER STORAGE
5482                                    ;                 CARRY CLEAR
5483                                    ;              IF STRING INVALID
5484                                    ;                 ERROR MESSAGE PRINTED
5485                                    ;                 CARRY SET
5486                                    ;--
5487
5488 047616                    BLDLWD:  PUSH    <R0,R1,R2,R3>         ; SAVE REGISTERS
     047616  010046                     MOV     R0,-(SP)             ; PUSH R0 ON STACK
     047620  010146                     MOV     R1,-(SP)             ; PUSH R1 ON STACK
     047622  010246                     MOV     R2,-(SP)             ; PUSH R2 ON STACK
     047624  010346                     MOV     R3,-(SP)             ; PUSH R3 ON STACK
5489 047626  005000                     CLR     R0                   ; START WITH ZEROS
5490 047630  005001                     CLR     R1
5491 047632  012702  002212             MOV     #TEMP,R2             ; GET ADDRESS OF STRING
5492 047636  112203            1$:      MOVB    (R2)+,R3             ; GET A DIGIT FROM STRING
5493 047640  001456                     BEQ     30$                  ; IF NULL CHARACTER, ALL DONE
5494 047642  022703  000040             CMP     #' ,R3               ; STRIP LEADING BLANKS
5495 047646  001003                     BNE     11$                  ;
5496 047650  000772                     BR      1$                   ;
5497 047652  112203            10$:     MOVB    (R2)+,R3             ; GET A DIGIT FROM STRING
5498 047654  001450                     BEQ     30$                  ; IF NULL CHARACTER, ALL DONE
5499 047656  162703  000060    11$:     SUB     #'0,R3               ; SUBTRACT CHARACTER 0
5500 047662  100433                     BMI     20$                  ;
5501 047664  022703  000011             CMP     #9.,R3               ;
5502 047670  103430                     BLO     20$                  ;
5503 047672  006300                     ASL     R0                   ; MULTIPLY BY 2
5504 047674  006101                     ROL     R1
5505 047676                             PUSH    <R1,R0>              ; SAVE N X 2
     047676  010146                     MOV     R1,-(SP)             ; PUSH R1 ON STACK
     047700  010046                     MOV     R0,-(SP)             ; PUSH R0 ON STACK
5506 047702  006300                     ASL     R0                   ; TIMES 2 AGAIN FOR N X 4
5507 047704  006101                     ROL     R1
5508 047706  006300                     ASL     R0                   ; TIMES 2 AGAIN FOR N X 8
5509 047710  006101                     ROL     R1
5510 047712  062600                     ADD     (SP)+,R0             ; ADD N X 2 TO GIVE N X 10
5511 047714  005501                     ADC     R1                   ;
5512 047716  062601                     ADD     (SP)+,R1             ;
5513 047720  060300                     ADD     R3,R0                ; ADD CURRENT DIGIT
5514 047722  005501                     ADC     R1                   ;
5515 047724  103352                     BCC     10$                  ; IF CARRY SET, # OVERFLOWS STORAGE
5516 047726                             PRINTF  #INP28A              ; PRINT PROPER RANGE
     047726  012746  065011             MOV     #INP28A,-(SP)
     047732  012746  000001             MOV     #1,-(SP)
     047736  010600                     MOV     SP,R0
```

```
        047740  104417                      TRAP    C$PNTF
        047742  062706  000004              ADD     #4,SP
5517 047746  000261                         SEC                        ; SET CARRY TO ASK AGAIN
5518 047750  000415                         BR      40$                ;
5519
5520 047752                      20$:       PRINTF  #INP28B            ; PRINT ILLEGAL CHARACTER
        047752  012746  065054              MOV     #INP28B,-(SP)
        047756  012746  000001              MOV     #1,-(SP)
        047762  010600                      MOV     SP,R0
        047764  104417                      TRAP    C$PNTF
        047766  062706  000004  .           ADD     #4,SP
5521 047772  000261                         SEC                        ;
5522 047774  000403                         BR      40$                ;
5523
5524 047776  010024             30$:        MOV     R0,(R4)+           ; MOVE NUMBER TO STORAGE AREA
5525 050000  010124                         MOV     R1,(R4)+           ;
5526 050002  000241                         CLC                        ; CLEAR CARRY TO INDICATE ALL IS WELL
5527 050004                      40$:        POP     <R3,R2,R1,R0>      ; RESTORE SAVED REGISTERS
5528 050014  000240                         NOP                        ; [DBG] INSERT HALT FOR DEBUG
5529 050016  000207                         RETURN
```

```
5531
5532                            ;++
5533                            ;           AVAIL - SEND MSCP AVAILABLE COMMAND PACKET
5534                            ;
5535                            ;           THIS COMMAND WILL CAUSE A DISK TO STOP SPINNING AND TO UNLOAD
5536                            ;           ITS HEADS.
5537                            ;
5538                            ;           INPUTS:
5539                            ;                   R0 - POINTER TO DRIVE TABLE
5540                            ;                   R3 - COMMAND MODIFIERS
5541                            ;                   R5 - ADDRESS OF CONTROLLER TABLE
5542                            ;
5543                            ;           OUTPUTS:
5544                            ;                   R4 - UNIT NUMBER
5545                            ;                   "TO BE SENT" LIST CONTAINS COMMAND PACKET
5546                            ;--
5547
5548 050020  011004             AVAIL:  MOV     (R0),R4                  ; GET DRIVE UNIT NUMBER
5549 050022  042760  001000 000002     BIC     #DT.ONL,D.UNIT(R0)       ; CLEAR UNIT ONLINE FLAG
5550 050030  012737  000010 002244     MOV     #OP.AVL,TSTOPC           ; SAVE OPCODE
5551 050036  004737  051214            CALL    BLDME                    ; BUILD MESSAGE ENVELOPE
5552 050042  103403                     BCS     1$                       ; IF SET, NO BUFFER AVAILABLE
5553 050044  012764  000014 000022     MOV     #12.,ME.CL(R4)           ;          LENGTHS
5554 050052                     1$:
5555 050052  000240             NOP                                      ; [DBG] INSERT HALT FOR DEBUG
5556 050054  000207             RETURN                                   ;
5557
5558                            ;++
5559                            ;           GCSTAT - SEND MSCP GET COMMAND STATUS COMMAND
5560                            ;
5561                            ;           THIS COMMAND WILL CAUSE THE CONTROLLER TO RETURN STATUS ON THE
5562                            ;           SPECIFIED MSCP COMMAND
5563                            ;
5564                            ;           INPUTS:
5565                            ;                   R1 - COMMAND REFERENCE NUMBER OF OUTSTANDING COMMAND
5566                            ;                   R5 - ADDRESS OF CONTROLLER TABLE
5567                            ;
5568                            ;           OUTPUTS:
5569                            ;                   R3 - CLEARED (NO COMMAND MODIFIERS)
5570                            ;                   R4 - CLEARED (NO UNIT SPECIFIED)
5571                            ;                   "TO BE SENT" LIST CONTAINS COMMAND PACKET
5572                            ;--
5573
5574 050056  005003             GCSTAT: CLR     R3                       ; NO MODIFIERS ALLOWED
5575 050060  005004                     CLR     R4                       ; UNIT MUST BE ZERO
5576 050062  012737  000002 002244     MOV     #OP.GCS,TSTOPC           ; SAVE OPCODE
5577 050070  004737  051214            CALL    BLDME                    ; BUILD MESSAGE ENVELOPE
5578 050074  103405                     BCS     1$                       ; IF SET, NO BUFFER AVAILABLE
5579 050076  010164  000042            MOV     R1,P.OTRF+ME.CP(R4)      ; SET OUTSTANDING REF #
5580 050102  012764  000020 000022     MOV     #16.,ME.CL(R4)           ;
5581 050110                     1$:
5582 050110  000240             NOP                                      ; [DBG] INSERT HALT FOR DEBUG
5583 050112  000207             RETURN                                   ;
5584
5585                            ;++
```

```
5586                                      ;      GUSTAT - SEND MSCP GET UNIT STATUS COMMAND
5587                                      ;
5588                                      ;      THIS COMMAND WILL CAUSE THE CONTROLLER TO RETURN UNIT STATUS ON THE
5589                                      ;      SPECIFIED UNIT
5590                                      ;
5591                                      ;      INPUTS:
5592                                      ;           R5 - ADDRESS OF CONTROLLER TABLE
5593                                      ;
5594                                      ;      OUTPUTS:
5595                                      ;           "TO BE SENT" LIST CONTAINS COMMAND PACKET
5596                                      ;--
5597
5598 050114  012703  000001      GUSTAT: MOV    #MD.NXU,R3              ; GET NEXT UNIT STATUS
5599 050120  005004                      CLR    R4                     ; SET UNIT TO ZERO
5600 050122  012737  000003 002244       MOV    #OP.GUS,TSTOPC         ; SAVE OPCODE
5601 050130  004737  051214              CALL   BLDME                  ; BUILD MESSAGE ENVELOPE
5602 050134  103403                      BCS    1$                     ; IF SET, NO BUFFER AVAILABLE
5603 050136  012764  000014 000022       MOV    #12.,ME.CL(R4)         ; SET COMMAND PACKET LENGTH
5604 050144                      1$:
5605 050144  000240              NOP                                   ; [DBG] INSERT HALT FOR DEBUG
5606 050146  000207                      RETURN                        ;
5607
5608                                      ;++
5609                                      ;      ONLINE - SEND MSCP ONLINE COMMAND
5610                                      ;
5611                                      ;      THIS COMMAND WILL CAUSE THE CONTROLLER TO SET THE SPECIFIED DRIVE
5612                                      ;      ONLINE AND RETURN DRIVE STATUS INFORMATION.
5613                                      ;
5614                                      ;      INPUTS:
5615                                      ;           R0 - POINTER TO DRIVE TABLE
5616                                      ;           R1 - UNIT FLAGS
5617                                      ;           R5 - ADDRESS OF CONTROLLER TABLE
5618                                      ;
5619                                      ;      OUTPUTS:
5620                                      ;           R3 - "SET WRITE PROTECT" CAMMAND MODIFIER SET
5621                                      ;           R4 - DRIVE UNIT NUMBER
5622                                      ;           "TO BE SENT" LIST CONTAINS COMMAND PACKET
5623                                      ;--
5624
5625 050150  012703  000004      ONLINE: MOV    #<MD.SWP>,R3           ; SET COMMAND MODIFIERS
5626 050154  011004                      MOV    (R0),R4                ; GET DRIVE UNIT NUMBER
5627 050156  012737  000011 002244       MOV    #OP.ONL,TSTOPC         ; SAVE OPCODE
5628 050164  004737  051214              CALL   BLDME                  ; BUILD MESSAGE ENVELOPE
5629 050170  103405                      BCS    1$                     ; IF SET, NO BUFFER AVAILABLE
5630 050172  010164  000044              MOV    R1,P.UNFL+ME.CP(R4)    ; SET THE UNIT FLAGS
5631 050176  012764  000044 000022       MOV    #36.,ME.CL(R4)         ; SET COMMAND PACKET LENGTH
5632 050204                      1$:
5633 050204  000240              NOP                                   ; [DBG] INSERT HALT FOR DEBUG
5634 050206  000207                      RETURN                        ;
5635
5636                                      ;++
5637                                      ;      SCC - SEND MSCP SET CONTROLLER CHARACTERISTICS COMMAND
5638                                      ;
5639                                      ;      THIS COMMAND WILL SET THE SPECIFIED CHARACTERISTICS OF THE
5640                                      ;      CONTROLLER
```

```
5641                                    ;          INPUTS:
5642                                    ;                  R1 - CONTROLLER FLAGS
5643                                    ;                  R5 - ADDRESS OF CONTROLLER TABLE
5644                                    ;
5645                                    ;
5646                                    ;          OUTPUTS:
5647                                    ;                  R3 - CLEARED (NO COMMAND MODIFIERS)
5648                                    ;                  R4 - CLEARED (NO UNIT SPECIFIED)
5649                                    ;                  "TO BE SENT" LIST CONTAINS COMMAND PACKET
5650                                    ;--
5651
5652 050210  005003              SCC:   CLR     R3                        ; NO MODIFIERS ALLOWED
5653 050212  005004                     CLR     R4                        ; UNIT MUST BE ZERO
5654 050214  012737  000004  002244     MOV     #OP.SCC,TSTOPC            ; SAVE OPCODE
5655 050222  004737  051214             CALL    BLDME                     ; BUILD MESSAGE ENVELOPE
5656 050226  103405                      BCS     1$                        ; IF SET, NO BUFFER AVAILABLE
5657 050230  010164  000044             MOV     R1,P.CNTF+ME.CP(R4)       ; SET THE CONTROLLER FLAGS
5658 050234  012764  000040  000022     MOV     #32.,ME.CL(R4)            ; SET UP COMMAND PACKET LENGTH
5659 050242                      1$:
5660 050242  000240              NOP                                      ; [DBG] INSERT HALT FOR DEBUG
5661 050244  000207              RETURN                                   ;
5662
5663                                    ;++
5664                                    ;          SUCHAR - SEND MSCP SET UNIT CHARACTERISTICS COMMAND
5665                                    ;
5666                                    ;          THIS COMMAND WILL CAUSE THE CONTROLLER TO SET THE APPROPRIATE
5667                                    ;          CHARACTERISTICS IN THE SPECIFIED DRIVE.
5668                                    ;
5669                                    ;          INPUTS:
5670                                    ;                  R0 - POINTER TO DRIVE TABLE
5671                                    ;                  R1 - UNIT FLAGS
5672                                    ;                  R5 - ADDRESS OF CONTROLLER TABLE
5673                                    ;
5674                                    ;          OUTPUTS:
5675                                    ;                  R3 - "SET WRITE PROTECT" CAMMAND MODIFIER SET
5676                                    ;                  R4 - DRIVE UNIT NUMBER
5677                                    ;                  "TO BE SENT" LIST CONTAINS COMMAND PACKET
5678                                    ;--
5679
5680 050246  012703  000004      SUCHAR: MOV    #<MD.SWP>,R3              ; SET COMMAND MODIFIERS
5681 050252  011004                     MOV     (R0),R4                   ; GET DRIVE UNIT NUMBER
5682 050254  012737  000012  002244     MOV     #OP.SUC,TSTOPC           ; SAVE OPCODE
5683 050262  004737  051214             CALL    BLDME                     ; BUILD MESSAGE ENVELOPE
5684 050266  103405                      BCS     1$                        ; IF SET, NO BUFFER AVAILABLE
5685 050270  010164  000044             MOV     R1,P.UNFL+ME.CP(R4)       ; SET THE UNIT FLAGS
5686 050274  012764  000044  000022     MOV     #36.,ME.CL(R4)            ; SET COMMAND PACKET LENGTH
5687 050302                      1$:
5688 050302  000240              NOP                                      ; [DBG] INSERT HALT FOR DEBUG
5689 050304  000207              RETURN                                   ;
```

```
5691
5692                          ;++
5693                          ;                 ACCESS - SEND MSCP ACCESS COMMAND PACKET
5694                          ;
5695                          ;        THIS COMMAND WILL CAUSE THE CONTROLLER TO READ DATA FROM A DRIVE,
5696                          ;        CHECK FOR ERRORS AND DISCARD THE DATA.
5697                          ;        THIS ROUTINE SETS THE OPCODE AND TRANSFERS CONTROL TO XFERPK TO
5698                          ;        BUILD THE TRANSFER COMMAND PACKET.
5699                          ;
5700                          ;        INPUTS:
5701                          ;                 R0 - POINTER TO DRIVE TABLE
5702                          ;                 R5 - ADDRESS OF CONTROLLER TABLE
5703                          ;        OUTPUTS:
5704                          ;                 R3 - COMMAND MODIFIERS
5705                          ;                 R4 - UNIT NUMBER
5706                          ;                 BUFBA,BUFEA - BUFFER DESCRIPTOR CLEARED
5707                          ;--
5708
5709 050306 005037 004370    ACCESS: CLR    BUFBA                     ; CLEAR BUFFER
5710 050312 005037 004372            CLR    BUFEA                     ;   DESCRIPTOR
5711 050316 005003                   CLR    R3                        ; SET COMMAND MODIFIERS
5712 050320 011004                   MOV    (R0),R4                   ; GET DRIVE UNIT NUMBER
5713 050322 012737 000020 002244     MOV    #OP.ACC,TSTOPC            ; SAVE OPCODE
5714 050330 022737 000003 002206     CMP    #3,TNUM                   ; IF NOT TEST 3,
5715 050336 001006                   BNE    2$                        ;   ALWAYS DO ECC
5716 050340 032737 000100 002154     BIT    #SM.ECC,SO.BIT+SFPTBL     ; ENABLE
5717 050346 001002                   BNE    2$                        ;   ECC?
5718 050350 052703 000100            BIS    #MD.SEC,R3                ; NO!
5719 050354 000137 050744    2$:     JMP    XFERPK                    ; BUILD TRANSFER PACKET
5720
5721                          ;++
5722                          ;                 CMPHD - SEND MSCP COMPARE HOST DATA COMMAND PACKET
5723                          ;
5724                          ;        THIS COMMAND WILL CAUSE THE CONTROLLER TO COMPARE THE DATA WRITTEN
5725                          ;        TO A DISK TO DATA IN A HOST MEMORY BUFFER.
5726                          ;        THIS ROUTINE REUSES THE MESSAGE ENVELOPE FROM A PREVIOUS
5727                          ;        WRITE OPERATION.
5728                          ;
5729                          ;        INPUTS:
5730                          ;                 R1 - DRIVE TABLE ADDRESS
5731                          ;                 C.RHDR(R5) - POINTER TO RESPONSE PACKET
5732                          ;                 R5 - ADDRESS OF CONTROLLER TABLE
5733                          ;        OUTPUTS:
5734                          ;                 MESSAGE ENVELOPE HAS BEEN ADDED TO THE TOP OF THE TBS LIST.
5735                          ;                 RESPONSE RING HAS BEEN CLEARED.
5736                          ;--
5737
5738 050360 005261 000060    CMPHD:  INC    D.CCNT(R1)                ; "COMPARE HOST DATA" OUTSTANDING COUNT
5739 050364 042761 004000 000002     BIC    #DT.CMP,D.UNIT(R1)        ; CLEAR "COMPARE HOST DATA" PENDING
5740 050372 016504 000104            MOV    C.RHDR(R5),R4             ; GET OLD COMMAND PACKET
5741 050376 016401 000006            MOV    P.CRF+RE.RP(R4),R1        ;   MESSAGE
5742 050402 004737 025550            CALL   GETME                    ;     ENVELOPE
5743 050406 016401 000002            MOV    ME.CBL(R4),R1             ; REMOVE
5744 050412 001004                   BNE    10$                      ;   MESSAGE
5745 050414 016465 000000 000160     MOV    ME.CFL(R4),C.CMFL(R5)     ;     ENVELOPE
```

```
5746 050422 000403                           BR      11$             ;           FROM
5747 050424 016461  000000 000000   10$:     MOV     ME.CFL(R4),ME.CFL(R1)  ;         COMMAND
5748 050432 016401  000000          11$:     MOV     ME.CFL(R4),R1   ;              LIST
5749 050436 001404                           BEQ     13$
5750 050440 016461  000002 000002            MOV     ME.CBL(R4),ME.CBL(R1)  ;
5751 050446 000403                           BR      14$             ;
5752 050450 016465  000002 000162   13$:     MOV     ME.CBL(R4),C.CMBL(R5)  ; ADD
5753 050456 016564  000164 000000   14$:     MOV     C.TBSF(R5),ME.CFL(R4)  ;     MESSAGE
5754 050464 010465  000164                   MOV     R4,C.TBSF(R5)   ;             ENVELOPE
5755 050470 005064  000002                   CLR     ME.CBL(R4)      ;               TO
5756 050474 016401  000000                   MOV     ME.CFL(R4),R1   ;                 TOP
5757 050500 001403                           BEQ     15$             ;                    OF
5758 050502 010461  000002                   MOV     R4,ME.CBL(R1)   ;                     TBS
5759 050506 000402                           BR      20$             ;                        LIST
5760 050510 010465  000166          15$:     MOV     R4,C.TBSB(R5)   ;
5761
5762 050514 012764  000040 000036   20$:     MOV     #OP.CMP,P.OPCD+ME.CP(R4); SET OPCODE
5763 050522 005064  000016                   CLR     ME.CFG(R4)      ; CLEAR M.E. FLAG FIELD
5764 050526 000240                           NOP             ; [DBG] INSERT HALT FOR DEBUG
5765 050530 000207                           RETURN          ;
5766
5767
5768                                 ;++
5769                                 ;       ERASE - SEND MSCP ERASE COMMAND PACKET
5770                                 ;
5771                                 ;       THIS COMMAND WILL CAUSE THE CONTROLLER TO ERASE DATA FROM THE
5772                                 ;       SELECTED LBNS ON A DREIVE. THE PATTERN WRITTEN IS THE SAME AS
5773                                 ;       THE DEFAULT FOR PATTERN 0.
5774                                 ;       THIS ROUTINE SETS THE OPCODE AND TRANSFERS CONTROL TO XFERPK TO
5775                                 ;       BUILD THE TRANSFER COMMAND PACKET.
5776                                 ;
5777                                 ;       INPUTS:
5778                                 ;               R0 - POINTER TO DRIVE TABLE
5779                                 ;               R5 - ADDRESS OF CONTROLLER TABLE
5780                                 ;       OUTPUTS:
5781                                 ;               R3 - COMMAND MODIFIERS
5782                                 ;               R4 - UNIT NUMBER
5783                                 ;               BUFBA,BUFEA - BUFFER DESCRIPTOR CLEARED
5784                                 ;--
5785 050532 005037  004370          ERASE:   CLR     BUFBA           ; CLEAR BUFFER
5786 050536 005037  004372                   CLR     BUFEA           ;     DESCRIPTOR
5787 050542 005003                           CLR     R3              ; SET COMMAND MODIFIERS
5788 050544 011004                           MOV     (R0),R4         ; GET DRIVE UNIT NUMBER
5789 050546 012737  000022 002244            MOV     #OP.ERS,TSTOPC  ; SAVE OPCODE
5790 050554 000473                           BR      XFERPK          ; BUILD TRANSFER PACKET
5791
5792                                 ;++
5793                                 ;       READ - SEND MSCP READ COMMAND
5794                                 ;
5795                                 ;       THIS COMMAND WILL CAUSE THE CONTROLLER TO READ DATA FROM A DRIVE
5796                                 ;       INTO HOST MEMORY AND CHECK FOR ERRORS.
5797                                 ;       THIS ROUTINE SETS THE OPCODE AND TRANSFERS CONTROL TO XFERPK TO
5798                                 ;       BUILD THE TRANSFER COMMAND PACKET.
5799                                 ;
5800                                 ;       INPUTS:
```

```
5801                                    ;                RO - POINTER TO DRIVE TABLE
5802                                    ;                R5 - ADDRESS OF CONTROLLER TABLE
5803                                    ;        OUTPUTS:
5804                                    ;                R3 - COMMAND MODIFIERS
5805                                    ;                R4 - UNIT NUMBER
5806                                    ;--
5807
5808 050556  005003           READ:    CLR     R3                      ; SET COMMAND MODIFIERS
5809 050560  011004                     MOV     (RO),R4                 ; GET DRIVE UNIT NUMBER
5810 050562  012737  000041  002244     MOV     #OP.RD,TSTOPC           ; SAVE OPCODE
5811 050570  022737  000003  002206     CMP     #3,TNUM                 ; IF NOT TEST 3,
5812 050576  001403                     BEQ     2$                      ;    ALWAYS DO
5813 050600  052703  040000             BIS     #MD.CMP,R3              ;      ECC AND
5814 050604  000417                     BR      10$                     ;        DATA COMPARE
5815
5816 050606  012701  000003    2$:      MOV     #3.,R1                  ; DO DATA
5817 050612  005002                     CLR     R2                      ;    COMPARES?
5818 050614  004737  026250             CALL    RANDOM                  ;
5819 050620  005701                     TST     R1                      ;
5820 050622  001002                     BNE     5$                      ;
5821 050624  052703  040000             BIS     #MD.CMP,R3              ;
5822 050630  032737  000100  002154 5$: BIT     #SM.ECC,SO.BIT+SFPTBL   ; ENABLE
5823 050636  001002                     BNE     10$                     ;    ECC?
5824 050640  052703  000100             BIS     #MD.SEC,R3              ; NO!
5825 050644  000437            10$:     BR      XFERPK                  ; BUILD THE TRANSFER PACKET
5826
5827                                    ;++
5828                                    ;
5829                                    ;        WRITE - SEND MSCP WRITE COMMAND
5830                                    ;
5831                                    ;        THIS COMMAND WILL CAUSE THE CONTROLLER TO WRITE DATA TO A DRIVE.
5832                                    ;        THIS ROUTINE SETS THE OPCODE AND TRANSFERS CONTROL TO XFERPK TO
5833                                    ;        BUILD THE TRANSFER COMMAND PACKET.
5834                                    ;
5835                                    ;        INPUTS:
5836                                    ;                RO - POINTER TO DRIVE TABLE
5837                                    ;                R5 - ADDRESS OF CONTROLLER TABLE
5838                                    ;        OUTPUTS:
5839                                    ;                R3 - COMMAND MODIFIERS
5840                                    ;                R4 - UNIT NUMBER
5841                                    ;--
5842 050646  005003           WRITE:   CLR     R3                      ; SET COMMAND MODIFIERS
5843 050650  011004                     MOV     (RO),R4                 ; GET DRIVE UNIT NUMBER
5844 050652  012737  000042  002244     MOV     #OP.WR,TSTOPC           ; SAVE OPCODE
5845 050660  022737  000003  002206     CMP     #3,TNUM                 ; IF NOT TEST 3,
5846 050666  001403                     BEQ     2$                      ;    ALWAYS DO
5847 050670  052703  040000             BIS     #MD.CMP,R3              ;      ECC AND
5848 050674  000423                     BR      10$                     ;        DATA COMPARE
5849
5850 050676  032765  002000  000014 2$: BIT     #CT.DET,C.FLG(R5)       ; IF DETERMINISTIC PHASE,
5851 050704  001017                     BNE     10$                     ;    DON'T DO DATA COMPARES
5852 050706  012701  000003             MOV     #3.,R1                  ; DO
5853 050712  005002                     CLR     R2                      ;    DATA
5854 050714  004737  026250             CALL    RANDOM                  ;      COMPARES?
5855 050720  005701                     TST     R1                      ;
```

```
5856 050722 001002                    BNE    4$                          ;
5857 050724 052703 040000             BIS    #MD.CMP,R3                  ;
5858 050730 032737 000100 002154 4$:  BIT    #SM.ECC,SO.BIT+SFPTBL       ; ENABLE
5859 050736 001002                    BNE    10$                         ;    ECC?
5860 050740 052703 000100             BIS    #MD.SEC,R3                  ; NO!
5861 050744                      10$:
5862
5863                              ;++
5864                              ;       XFERPK - BUILD TRANSFER COMMAND PACKET
5865                              ;
5866                              ;       THIS ROUTINE BUILDS A TRANSFER COMMAND PACKET.
5867                              ;
5868                              ; INPUTS:
5869                              ;               BUFEA - BUFFER DESCRIPTOR (LOW WORD)
5870                              ;               BUFBA - BUFFER DESCRIPTOR (HIGH WORD)
5871                              ;               BCLO - BYTE COUNT (LOW WORD)
5872                              ;               BCHI - BYTE COUNT (HIGH WORD)
5873                              ;               TSTOPC - OPCODE
5874                              ;               R3 - COMMAND MODIFIERS
5875                              ;               R4 - UNIT NUMBER
5876                              ;               R5 - ADDRESS OF CONTROLLER TABLE
5877                              ;
5878                              ; OUTPUTS:
5879                              ;               "TO BE SENT" LIST CONTAINS COMMAND PACKET
5880                              ;--
5881
5882 050744 004737 051214        XFERPK: CALL   BLDME                    ; BUILD MESSAGE ENVELOPE
5883 050750 103456                       BCS    20$                      ; IF SET, NO BUFFER AVAILABLE
5884 050752 022737 000042 002244         CMP    #OP.WR,TSTOPC            ; IF COMMAND
5885 050760 001404                        BEQ    1$                      ;    IS WRITE OR ERASE,
5886 050762 022737 000022 002244         CMP    #OP.ERS,TSTOPC           ;      INCREMENT
5887 050770 001002                        BNE    2$                      ;        OUTSTANDING
5888 050772 005260 000056          1$:   INC    D.WCNT(R0)               ;          WRITE COUNT
5889 050776 022737 000003 002206   2$:   CMP    #3,TNUM                  ; IF NOT TEST 3,
5890 051004 001012                        BNE    10$                     ;    ALWAYS DO ERROR RECOVERY
5891 051006 032765 002000 000014         BIT    #CT.DET,C.FLG(R5)        ; IF DETERMINISTIC PHASE,
5892 051014 001006                        BNE    10$                     ;    DO ERROR RECOVERY
5893 051016 032737 000040 002154         BIT    #SM.EER,SO.BIT+SFPTBL    ; DO
5894 051024 001002                        BNE    10$                     ;    ERROR RECOVERY?
5895 051026 052703 000400                 BIS    #MD.SER,R3              ; NO!
5896 051032 012764 000040 000022  10$:   MOV    #32.,ME.CL(R4)           ; SET COMMAND PACKET LENGTH
5897 051040 013764 004406 000042         MOV    BCLO,P.BCNT+ME.CP(R4)    ; STORE THE BUFFER SIZE IN WORDS
5898 051046 013764 004410 000044         MOV    BCHI,P.BCNT+2+ME.CP(R4)  ; SET READ BUFFER SIZE
5899 051054 013764 004372 000050         MOV    BUFEA,P.ADEA+ME.CP(R4)   ; SET THE BUFFER DESCRIPTOR
5900 051062 013764 004370 000046         MOV    BUFBA,P.ADBA+ME.CP(R4)   ;
5901 051070 016064 000052 000062         MOV    D.LBN(R0),P.LBN+ME.CP(R4); SET LBN (LOW WORD)
5902 051076 016064 000054 000064         MOV    D.LBN+2(R0),P.LBN+2+ME.CP(R4); SET LBN (HIGH WORD)
5903 051104 000241                        CLC                            ; CLEAR CARRY TO INDICATE SUCCESS
5904 051106                         20$:
5905 051106 000240                 NOP                    ; [DBG] INSERT HALT FOR DEBUG
5906 051110 000207                 RETURN                 ;
5907
5908                              ;++
5909                              ;       MNTRD - SEND UQSSP MAINTENANCE READ COMMAND
5910                              ;
```

```
5911                                        ;       THIS COMMAND WILL CAUSE DATA TO BE READ FROM THE SPECIFIED
5912                                        ;       LOCATIONS IN CONTROLLER MEMORY.
5913                                        ;       THIS ROUTINE SETS THE OPCODE AND TRANSFERS CONTROL TO MAINTP TO
5914                                        ;       BUILD THE MAINTENANCE COMMAND PACKET.
5915                                        ;--
5916
5917 051112  012737  000030  002244  MNTRD:  MOV     #OP.MRD,TSTOPC          ; SAVE OPCODE
5918 051120  000403                          BR      MAINTP                 ; BUILD THE TRANSFER PACKET
5919
5920                                        ;++
5921                                        ;       MNTWR - SEND UQSSP MAINTENANCE WRITE COMMAND
5922                                        ;
5923                                        ;       THIS COMMAND WILL CAUSE A BUFFER OF HOST DATA TO BE WRITTEN TO
5924                                        ;       THE SPECIFIED LOCATIONS IN CONTROLLER MEMORY.
5925                                        ;       THIS ROUTINE SETS THE OPCODE AND TRANSFERS CONTROL TO MAINTP TO
5926                                        ;       BUILD THE MAINTENANCE COMMAND PACKET.
5927                                        ;--
5928
5929 051122  012737  000031  002244  MNTWR:  MOV     #OP.MWR,TSTOPC         ; SAVE OPCODE
5930
5931                                        ;++
5932                                        ;       MAINTP - BUILD MAINTENANCE COMMAND PACKET
5933                                        ;
5934                                        ;       THIS ROUTINE BUILDS A MAINTENANCE COMMAND PACKET.
5935                                        ;
5936                                        ;       INPUTS:
5937                                        ;               BUFEA - BUFFER DESCRIPTOR (LOW WORD)
5938                                        ;               BCLO - BYTE COUNT
5939                                        ;               R1 - REGION OFFSET
5940                                        ;               R3 - COMMAND MODIFIERS
5941                                        ;               R4 - UNIT NUMBER
5942                                        ;               R5 - ADDRESS OF CONTROLLER TABLE
5943                                        ;
5944                                        ;       OUTPUTS:
5945                                        ;               "TO BE SENT" LIST CONTAINS COMMAND PACKET
5946                                        ;--
5947
5948 051130  005003                  MAINTP: CLR     R3                      ; COMMAND MODIFIERS NOT ALLOWED
5949 051132  004737  051214                  CALL    BLDME                   ; BUILD MESSAGE ENVELOPE
5950 051136  103424                          BCS     1$                      ; IF SET, NO BUFFER AVAILABLE
5951 051140  012764  000044  000022          MOV     #36.,ME.CL(R4)          ; SET COMMAND PACKET LENGTH
5952 051146  013764  004406  000042          MOV     BCLO,P.BCNT+ME.CP(R4)   ; STORE THE BUFFER SIZE IN BYTES
5953 051154  013764  004372  000050          MOV     BUFEA,P.ADEA+ME.CP(R4)  ; SET THE BUFFER DESCRIPTOR
5954 051162  013764  004370  000046          MOV     BUFBA,P.ADBA+ME.CP(R4)  ;
5955 051170  012764  000001  000062          MOV     #1,P.RGID+ME.CP(R4)     ; SET REGION ID (HIGH WORD 0)
5956 051176  010164  000066                  MOV     R1,P.RGOF+ME.CP(R4)     ; SET REGION OFFSET (HIGH WORD 0)
5957 051202  012764  177760  000024          MOV     #MAINT,ME.CV(R4)        ; CIRCUIT ID = 177400 FOR MAINTENANCE
5958 051210                          1$:
5959 051210  000240                          NOP                             ; [DBG] INSERT HALT FOR DEBUG
5960 051212  000207                          RETURN                          ;
```

```
5962
5963                                    ;++
5964                                    ;          BLDME - BUILD MESSAGE ENVELOPE AND DATA STRUCTURES FOR COMMAND
5965                                    ;                  PACKET
5966                                    ;
5967                                    ;          INPUTS:
5968                                    ;                  R3 - COMMAND MODIFIERS
5969                                    ;                  R4 - UNIT NUMBER
5970                                    ;                  R5 - ADDRESS OF CONTROLLER TABLE
5971                                    ;                  TSTOPC - OPCODE
5972                                    ;          OUTPUT:
5973                                    ;                  COMMAND AND RESPONSE PACKETS HAVE BEEN CLEARED
5974                                    ;                  COMMAND PACKET CONTAINS UNIT NUMBER, COMMAND REFERENCE
5975                                    ;                       NUMBER, OPCODE, AND COMMAND MODIFIERS
5976                                    ;                  R4 - ADDRESS OF COMMAND PACKET BUFFER
5977                                    ;                  CARRY CLEAR ON SUCCESS
5978                                    ;                  CARRY SET ON ERROR (NO BUFFERS AVAILABLE)
5979                                    ;--
5980
5981 051214                     BLDME:  PUSH    <R0>                    ; SAVE REGISTERS
5982 051216   005737   006236           TST     TBSSIZ                  ; IF NO PACKETS LEFT,
5983 051222   001526                     BEQ     10$                     ;    EXIT
5984 051224   032737   000070  002244    BIT     #70,TSTOPC              ; IF IMMEDIATE COMMAND
5985 051232   001404                     BEQ     1$                      ;    BUILD IT
5986 051234   023737   002202  006236    CMP     CTRLRS,TBSSIZ           ; ALWAYS SAVE 1 PACKET
5987 051242   002116                     BGE     10$                     ;    PER CONTROLLER FOR IMMED. CMD
5988 051244                     1$:      PUSH    <R4>                    ; SAVE COMMAND UNIT NUMBER
5989 051246   013704   006242           MOV     T.HDR,R4                ; R4 POINTS TO PACKET
5990 051252   016400   000000           MOV     ME.CFL(R4),R0           ; GET POINTER TO NEXT PACKET
5991 051256   010037   006242           MOV     R0,T.HDR                ; HEADER POINTS TO NEXT PACKET
5992 051262   005060   000002           CLR     ME.CBL(R0)              ; FIX BACK LINK OF NEXT PACKET
5993 051266   005337   006236           DEC     TBSSIZ                  ; DECREMENT AVAILABLE COUNT
5994
5995 051272   032737   000070  002244    BIT     #70,TSTOPC              ; IF IMMEDIATE, PUT AT TOP OF LIST
5996 051300   001414                     BEQ     2$                      ; ELSE, PUT AT END OF LIST
5997 051302   016500   000166           MOV     C.TBSB(R5),R0           ; GET BACK LINK TO TBS LIST
5998 051306   001425                     BEQ     3$                      ; CHECK FOR EMPTY LIST
5999 051310   010460   000000           MOV     R4,ME.CFL(R0)           ; LAST PACKET POINTS TO NEW PACKET
6000 051314   010064   000002           MOV     R0,ME.CBL(R4)           ; NEW PACKET POINTS BACK TO LIST
6001 051320   005064   000000           CLR     ME.CFL(R4)              ; NEW PACKET POINTS FORWARD TO HEADER
6002 051324   010465   000166           MOV     R4,C.TBSB(R5)           ; HEADER POINTS TO NEW PACKET
6003 051330   000424                     BR      4$                      ; PUT NEW BUFFER IN MAPPED MEMORY
6004
6005 051332   016500   000164    2$:     MOV     C.TBSF(R5),R0           ; GET FORWARD LINK TO CMD LIST
6006 051336   001411                     BEQ     3$                      ; CHECK FOR EMPTY LIST
6007 051340   010460   000002           MOV     R4,ME.CBL(R0)           ; FIRST PACKET POINTS TO NEW PACKET
6008 051344   010064   000000           MOV     R0,ME.CFL(R4)           ; NEW PACKET POINTS FORWARD TO LIST
6009 051350   005064   000002           CLR     ME.CBL(R4)              ; NEW PACKET POINTS BACK TO HEADER
6010 051354   010465   000164           MOV     R4,C.TBSF(R5)           ; HEADER POINTS TO NEW PACKET
6011 051360   000410                     BR      4$                      ; PUT NEW BUFFER IN MAPPED MEMORY
6012
6013 051362   005064   000000    3$:     CLR     ME.CFL(R4)              ; LIST WAS
6014 051366   005064   000002           CLR     ME.CBL(R4)              ;    EMPTY
6015 051372   010465   000164           MOV     R4,C.TBSF(R5)           ; HEADER POINTS TO
6016 051376   010465   000166           MOV     R4,C.TBSB(R5)           ;    NEXT PACKET
```

```
6017
6018 051402  012700  000046           4$:    MOV     #<CMDSIZ-3>,R0              ; R0 = # OF WORDS TO CLEAR
6019 051406                                   PUSH    <R4>                       ; SAVE ADDRESS OF TOP
6020 051410  062704  000006                   ADD     #ME.CTO,R4                 ; R4 -> 1ST WORD TO CLEAR
6021 051414  005024                    5$:    CLR     (R4)+                      ; CLEAR WORD
6022 051416  005300                           DEC     R0                         ; DECREMENT WORD COUNT
6023 051420  001375                           BNE     5$                         ; IF NOT ZERO, LOOP
6024 051422                                   POP     <R4>                       ; RESTORE POINTER TO TOP
6025 051424  013764  002242  000020           MOV     TSTOFF,ME.OFF(R4)          ; SET MEMORY PACKET OFFSET
6026 051432  012737  177777  002242           MOV     #-1,TSTOFF                 ; -1 INDICATES NO BUFFER USED
6027 051440  005064  000016                   CLR     ME.CFG(R4)                 ; SET M. E. FLAGS
6028 051444  016464  000004  000026           MOV     ME.CRF(R4),P.CRF+ME.CP(R4) ; SET COMMAND REF NUMBER
6029 051452  012664  000032                   MOV     (SP)+,P.UNIT+ME.CP(R4)     ; SET UNIT NUMBER
6030 051456  013764  002244  000036           MOV     TSTOPC,P.OPCD+ME.CP(R4)    ; SET OPCODE
6031 051464  010364  000040                   MOV     R3,P.MOD+ME.CP(R4)         ; SET COMMAND MODIFIERS
6032 051470  000241                           CLC                                ; SUCCESS RETURN
6033 051472  000240                    NOP                        ; [DBG] INSERT HALT FOR DEBUG
6034 051474                            9$:    POP     <R0>                       ; RESTORE SAVED REGISTERS
6035 051476  000207                           RETURN                             ; RETURN
6036
6037 051500  000261                    10$:   SEC                                ; INDICATE ERROR
6038 051502  000774                           BR      9$                         ;
```

```
6040
6041                              ;++
6042                              ;          SNDMSG - SEND A PACKET TO THE CONTROLLER
6043                              ;
6044                              ;          SEND FIRST PACKET IN THE 'TO BE SENT' LIST
6045                              ;          SET OWN, CLEAR FLAG IN THE COMMAND RING (FOR CONTROLLER).
6046                              ;          SET OWN & FLAG IN MESSAGE RING (FOR INTERRUPTS BY CONTROLLER).
6047                              ;          FORCE POLLING TO START.
6048                              ;
6049                              ;          INPUT:
6050                              ;               R5 - ADDRESS OF CONTROLLER TABLE
6051                              ;          OUTPUT:
6052                              ;--
6053
6054 051504                      SNDMSG: PUSH    <R1,R3,R4>              ; SAVE REGISTERS
6055 051512  016503  000070      1$:     MOV     C.CPTR(R5),R3           ; POINT TO COMMAND RING ENTRY
6056 051516  032763  100000  000002      BIT     #RG.OWN,2(R3)          ; IF CONTROLLER OWNS
6057 051524  001154                      BNE     40$                    ;     COMMAND RING, EXIT
6058 051526  016504  000164              MOV     C.TBSF(R5),R4          ; GET FORWARD LINK TO TBS LIST
6059 051532  001551                      BEQ     40$                    ; IF LIST EMPTY, EXIT
6060 051534  032764  000070  000036      BIT     #70,P.OPCD+ME.CP(R4)   ; IF IMMEDIATE COMMAND,
6061 051542  001441                      BEQ     10$                    ;     USE SPECIAL CASE CODE
6062
6063                              ;          GET SEQUENTIAL COMMAND AND PUT AT END OF CMD LIST
6064                              ;
6065 051544  005765  000152              TST     C.CRED(R5)             ; IF CREDIT LIMIT NOT POSITIVE,
6066 051550  003542                      BLE     40$                    ;     EXIT
6067 051552  005365  000152              DEC     C.CRED(R5)             ; DECREMENT CREDITS
6068 051556  012764  177777  000012      MOV     #-1,ME.ST1(R4)         ; INITIALIZE
6069 051564  012764  177777  000014      MOV     #-1,ME.ST2(R4)         ;     COMMAND STATUS
6070 051572  016401  000000              MOV     ME.CFL(R4),R1          ; GET POINTER TO NEXT PACKET
6071 051576  001403                      BEQ     2$                     ; IF END OF LIST, SKIP
6072 051600  005061  000002              CLR     ME.CBL(R1)             ;     POINT TO HEADER
6073 051604  000402                      BR      3$                     ;     CONTINUE
6074 051606  005065  000166      2$:     CLR     C.TBSB(R5)             ; BLINK HEADER ZERO, TBS LIST IS EMPTY
6075 051612  010165  000164      3$:     MOV     R1,C.TBSF(R5)          ; FLINK HEADER POINTS TO NEXT PACKET
6076 051616  016501  000160              MOV     C.CMFL(R5),R1          ; GET FORWARD LINK TO CMD LIST
6077 051622  001451                      BEQ     20$                    ; CHECK FOR EMPTY LIST
6078 051624  010461  000002              MOV     R4,ME.CBL(R1)          ; FIRST PACKET POINTS TO NEW PACKET
6079 051630  010164  000000              MOV     R1,ME.CFL(R4)          ; NEW PACKET POINTS FORWARD TO LIST
6080 051634  005064  000002              CLR     ME.CBL(R4)             ; NEW PACKET POINTS BACK TO HEADER
6081 051640  010465  000160              MOV     R4,C.CMFL(R5)          ; HEADER POINTS TO NEW PACKET
6082 051644  000450                      BR      30$                    ; SEND PACKET
6083
6084 051646  005765  000152      10$:    TST     C.CRED(R5)             ; IF CREDIT LIMIT NEGATIVE,
6085 051652  002501                      BLT     40$                    ;     EXIT
6086 051654  005365  000152              DEC     C.CRED(R5)             ; DECREMENT CREDITS
6087 051660  012764  000001  000012      MOV     #1,ME.ST1(R4)          ; INITIALIZE
6088 051666  005064  000014              CLR     ME.ST2(R4)             ;     COMMAND STATUS
6089 051672  016401  000000              MOV     ME.CFL(R4),R1          ; GET POINTER TO NEXT PACKET
6090 051676  001403                      BEQ     12$                    ; IF END OF LIST, SKIP
6091 051700  005061  000002              CLR     ME.CBL(R1)             ;     POINT TO HEADER
6092 051704  000402                      BR      13$                    ;     CONTINUE
6093 051706  005065  000166      12$:    CLR     C.TBSB(R5)             ; BLINK HEADER ZERO, TBS LIST IS EMPTY
6094 051712  010165  000164      13$:    MOV     R1,C.TBSF(R5)          ; FLINK HEADER POINTS TO NEXT PACKET
```

```
6095 051716  016501  000162              MOV     C.CMBL(R5),R1            ; GET BACK LINK TO CMD LIST
6096 051722  001411                      BEQ     20$                     ; CHECK FOR EMPTY LIST
6097 051724  010461  000000              MOV     R4,ME.CFL(R1)           ; LAST PACKET POINTS TO NEW PACKET
6098 051730  010164  000002              MOV     R1,ME.CBL(R4)           ; NEW PACKET POINTS BACK TO LIST
6099 051734  005064  000000              CLR     ME.CFL(R4)              ; NEW PACKET POINTS FORWARD TO HEADER
6100 051740  010465  000162              MOV     R4,C.CMBL(R5)           ; HEADER POINTS TO NEW PACKET
6101 051744  000410                      BR      30$                     ; MAP BUFFER TO HI MEM
6102
6103 051746  005064  000000      20$:    CLR     ME.CFL(R4)              ; CMD LIST WAS
6104 051752  005064  000002              CLR     ME.CBL(R4)              ;   EMPTY
6105 051756  010465  000160              MOV     R4,C.CMFL(R5)           ; HEADER POINTS TO
6106 051762  010465  000162              MOV     R4,C.CMBL(R5)           ;   NEXT PACKET
6107 051766                      30$:
6109 051766  004737  023704              CALL    DBGCMD                  ; DUMP COMMAND PACKET
6111 051772  010413                      MOV     R4,(R3)                 ; GET PHYSICAL ADDRESSES
6112 051774  062713  000026              ADD     #ME.CP,(R3)             ;   OF COMMAND PACKET
6113 052000  012763  100000  000002      MOV     #RG.OWN,2(R3)           ; SET OWNERSHIP IN COMMAND RING
6114 052006  005775  000000              TST     @(R5)                   ; READ IP REG TO START POLLING
6115 052012  016501  000006              MOV     C.CTO(R5),R1            ; TIMEOUT = CONTROLLER TIMEOUT
6116 052016  010400                      MOV     R4,R0                   ; POINT TO TIME OUT COUNTER
6117 052020  062700  000006              ADD     #ME.CTO,R0              ; POINTER TO TIMER FIELD
6118 052024  004737  052070              CALL    SETTO                   ; START TIMER
6119 052030  062765  000004  000070      ADD     #4,C.CPTR(R5)           ; ADJUST COMMAND RING POINTER
6120 052036  026565  000070  000074      CMP     C.CPTR(R5),C.CEND(R5)   ; IF POINTER IS NOT AT END,
6121 052044  002622                      BLT     1$                      ;     SEND ANOTHER COMMAND.
6122 052046  016565  000072  000070      MOV     C.CBAS(R5),C.CPTR(R5)   ; ELSE,
6123 052054  000616                      BR      1$                      ;     RE-INIT POINTER & CONTINUE.
6124 052056                      40$:    POP     <R4,R3,R1>              ; RESTORE SAVED REGISTERS
6125 052064  000240              NOP                     ; [DBG] INSERT HALT FOR DEBUG
6126 052066  000207                      RETURN                          ; RETURN
```

```
6128
6129                                      ;++
6130                                      ;          SETTO - SET TIMEOUT COUNTER TO A GIVEN NUMBER OF SECONDS FROM CURRENT
6131                                      ;                  TIME.
6132                                      ;
6133                                      ;          INPUTS:
6134                                      ;                  R0 - ADDRESS OF STORAGE FOR TWO WORD TIMER
6135                                      ;                  R1 - NUMBER OF SECONDS FOR TIMEOUT
6136                                      ;          OUTPUTS:
6137                                      ;                  R0 - INCREMENTED BY 2
6138                                      ;                  R1 - CONTENTS DESTROYED
6139                                      ;--
6140
6141  052070  005737  002322     SETTO:  TST     KW.CSR                  ; IF NO CLOCK,
6142  052074  001421                     BEQ     1$                      ;    EXIT
6143  052076                             PUSH    <R2,R3,R4>              ; SAVE REGISTERS
6144
6145  052104  013702  002330             MOV     KW.HZ,R2                ; GET MULTIPLICAND
6146  052110  004737  047420             CALL    MULT                    ; PERFORM MULTIPLICATION
6147
6148  052114  063703  002332             ADD     KW.EL,R3                ; GET CURRENT TIME (LOW WORD)
6149  052120  005504                     ADC     R4                      ; INCREMENT HIGH WORD IF CARRY
6150  052122  063704  002334             ADD     KW.EL+2,R4              ; GET CURRENT TIME (HIGH WORD)
6151
6152  052126  010320                     MOV     R3,(R0)+                ; SAVE LOW WORD OF TIMEOUT
6153  052130  010410                     MOV     R4,(R0)                 ; SAVE HIGH WORD OF TIMEOUT
6154
6155  052132                             POP     <R4,R3,R2>
6156  052140                     1$:
6157  052140  000240             NOP                     ; [DBG] INSERT HALT FOR DEBUG
6158  052142  000207                     RETURN
6159
6160                                      ;++
6161                                      ;          SETMIN - SET TIMEOUT COUNTER TO A GIVEN NUMBER OF MINUTES FROM CURRENT
6162                                      ;                   TIME.
6163                                      ;
6164                                      ;          INPUTS:
6165                                      ;                  R0 - ADDRESS OF STORAGE FOR TWO WORD TIMER
6166                                      ;                  R1 - NUMBER OF SECONDS FOR TIMEOUT
6167                                      ;          OUTPUTS:
6168                                      ;                  R0 - INCREMENTED BY 2
6169                                      ;                  R1 - CONTENTS DESTROYED
6170                                      ;--
6171
6172  052144  005737  002322     SETMIN: TST     KW.CSR                  ; IF NO CLOCK,
6173  052150  001421                     BEQ     1$                      ;    EXIT
6174  052152                             PUSH    <R2,R3,R4>              ; SAVE REGISTERS
6175
6176  052160  013702  002336             MOV     KW.60,R2                ; GET MULTIPLICAND
6177  052164  004737  047420             CALL    MULT                    ; PERFORM MULTIPLICATION
6178
6179  052170  063703  002332             ADD     KW.EL,R3                ; GET CURRENT TIME (LOW WORD)
6180  052174  005504                     ADC     R4                      ; INCREMENT HIGH WORD IF CARRY
6181  052176  063704  002334             ADD     KW.EL+2,R4              ; GET CURRENT TIME (HIGH WORD)
6182
```

```
6183 052202  010320              MOV    R3,(R0)+              ; SAVE LOW WORD OF TIMEOUT
6184 052204  010410              MOV    R4,(R0)               ; SAVE HIGH WORD OF TIMEOUT
6185
6186 052206                      POP    <R4,R3,R2>
6187 052214            1$:
6188 052214  000240    NOP                        ; [DBG] INSERT HALT FOR DEBUG
6189 052216  000207              RETURN
6190
```

```
6192
6193                              ;++
6194                              ;          CNTINT - INITIALIZE A CONTROLLER AND BRING IT ON-LINE.  ALL STEPS
6195                              ;          ARE CHECKED. AN ERROR MESSAGE IS REPORTED IF ANY ERROR IS DETECTED.
6196                              ;
6197                              ;          INPUTS:
6198                              ;              R5 - ADDRESS OF CONTROLLER TABLE.
6199                              ;          OUTPUTS:
6200                              ;              R4 - ADDRESS OF IP REGISTER IN CONTROLLER
6201                              ;              R5 - UNCHANGED
6202                              ;              CARRY CLEAR IF NO ERROR, SET IF ANY ERROR REPORTED
6203                              ;--
6204
6205 052220                      CNTINT: PUSH    <R2,R3>                    ; SAVE REGISTERS ON STACK
6206                              ;
6207                              ;          SET UP INTERRUPT VECTOR & SET UP DATA TO SEND TO SA REGISTER
6208                              ;
6209 052224  016504  000004      1$:     MOV     C.VEC(R5),R4               ; GET VECTOR OF CONTROLLER
6210 052230  042704  177000              BIC     #^C<CT.VEC>,R4             ; MASK OTHER BITS
6211 052234  010501                      MOV     R5,R1                      ; GET INTERRUPT SERVICE LINK
6212 052236  062701  000010              ADD     #C.JSR,R1                  ;
6213 052242                              SETVEC  R4,R1,#PRI07               ; SET UP INTERRUPT VECTOR
     052242  012746  000340              MOV     #PRI07,-(SP)
     052246  010146                      MOV     R1,-(SP)
     052250  010446                      MOV     R4,-(SP)
     052252  012746  000003              MOV     #3,-(SP)
     052256  104437                      TRAP    C$SVEC
     052260  062706  000010              ADD     #10,SP
6214 052264  006204                      ASR     R4                         ; DIVIDE VECTOR
6215 052266  006204                      ASR     R4                         ;    BY 4
6216 052270  052704  011000              BIS     #RNGPWR,R4                 ; INSERT CMD/RSP QUE LENGTH
6217 052274  052704  100000              BIS     #SA.STP,R4                 ; SET BIT 15 IN DATA WORD
6218 052300  010437  002342              MOV     R4,SND.S1                  ; LOAD LEN, IE FLAG & VECTOR FOR STEP 1
6219 052304  016537  000100  002346      MOV     C.RBAS(R5),SND.S2          ; GET RING BASE
6220                              ;
6221                              ;          WRITE -1 TO ALL WORDS IN HOST COMMUNICATIONS AREA FOR STEP 3
6222                              ;
6223 052312  010502                      MOV     R5,R2                      ; GET FIRST ADDRESS
6224 052314  062702  000106              ADD     #C.CINT,R2                 ;    OF RING BUFFER
6225 052320  012700  000006              MOV     #6,R0                      ; GET SIZE OF RING BUFFER
6226 052324  012722  177777      5$:     MOV     #-1,(R2)+                  ; WRITE ONES TO EACH WORD IN BUFFER
6227 052330  005300                      DEC     R0                         ; IF NOT END OF BUFFER,
6228 052332  003374                      BGT     5$                         ;    LOOP
6229                              ;
6230                              ;          VERIFY THE ADDRESS OF THE SA AND IP REGISTERS ARE VALID AND
6231                              ;          START THE INITIALIZATION
6232                              ;
6233 052334  016504  000000              MOV     C.IPR(R5),R4               ; GET ADDRESS OF IP REGISTER
6234 052340  005037  002264              CLR     NXMAD                      ; CLEAR MEMORY ERROR FLAG
6235 052344                              SETVEC  #ERRVEC,#NXMI,#PRI07       ; SETUP TIMEOUT ERROR VECTOR
     052344  012746  000340              MOV     #PRI07,-(SP)
     052350  012746  054472              MOV     #NXMI,-(SP)
     052354  012746  000004              MOV     #ERRVEC,-(SP)
     052360  012746  000003              MOV     #3,-(SP)
     052364  104437                      TRAP    C$SVEC
```

```
        052366  062706  000010              ADD     #10,SP              ; ACCESS SA REGISTER
6236    052372  005764  000002              TST     2(R4)               ; WRITE TO IP
6237    052376  005014                      CLR     (R4)                ; RETURN TIMEOUT ERROR VECTOR
6238    052400                              CLRVEC  #ERRVEC
        052400  012700  000004              MOV     #ERRVEC,R0
        052404  104436                      TRAP    C$CVEC
6239    052406  005737  002264              TST     NXMAD               ; SEE IF A MEMORY ERROR OCCURRED
6240    052412  001406                      BEQ     11$                 ; IF NO ERROR, CONTINUE
6241    052414                              ERRDF   37,,ERR037          ; PRINT ERROR MESSAGE
        052414  104455                      TRAP    C$ERDF
        052416  000045                      .WORD   37
        052420  000000                      .WORD   0
        052422  020510                      .WORD   ERR037
6242    052424  000137  053066              JMP     41$                 ; EXIT
6243                                 ;
6244                                 ;      EXECUTE THE FIRST THREE STEPS OF INITIALIZATION AND CHECK
6245                                 ;      RESPONSE FROM CONTROLLER
6246                                 ;
6247    052430  012737  004000  002354 11$: MOV     #SA.S1,CNTRSD       ; STORE RESPONSE MASK
6248    052436  012703  002340       12$:   MOV     #INITBL,R3          ; GET INDEX TO SEND/REPOND INIT TABLE
6249    052442  004737  053252              CALL    CNTRSP              ; WAIT FOR STEP OR ERROR BITS
6250    052446  103002                      BCC     13$                 ;   EXIT IF ERROR
6251    052450  000137  053066              JMP     41$                 ;
6252    052454  004733                13$:  CALL    @(R3)+              ; CALL RESPONSE CHECKER FOR STEP
6253    052456  103002                      BCC     14$                 ;
6254    052460  000137  053066              JMP     41$                 ;   EXIT IF ERROR
6255    052464  006337  002354        14$:  ASL     CNTRSD              ; SHIFT TO NEXT STEP BIT
6256    052470  032737  040000  002354      BIT     #SA.S4,CNTRSD       ; IF NOW AT STEP 4,
6257    052476  001003                      BNE     20$                 ;   EXIT LOOP
6258    052500  012364  000002              MOV     (R3)+,2(R4)         ; WRITE DATA TO SA REGISTER
6259    052504  000756                      BR      12$                 ; STAY IN LOOP
6260                                 ;
6261                                 ;      PERFORM STEP 3. PERFORM PURGE/POLL TEST IF ENHANCED DIAGNOSTICS ARE
6262                                 ;      SUPPORTED BY THE CONTROLLER.
6263                                 ;
6264    052506  032765  004000  000014 20$: BIT     #CT.DI,C.FLG(R5)    ; IF ENHANCED DIAGS NOT SUPPORTED,
6265    052514  001423                      BEQ     25$                 ;   DON'T DO PURGE/POLL
6266    052516  012700  000310              MOV     #200.,R0            ; GET LOOP COUNTER
6267    052522  012764  100000  000002      MOV     #SA.TST,2(R4)       ; WRITE STEP 3 RESPONSE TO SA REGISTER
6268    052530  016402  000002        23$:  MOV     2(R4),R2            ; IF SA REGISTER IS ZERO,
6269    052534  001407                      BEQ     24$                 ;   EXIT LOOP
6270    052536  005300                      DEC     R0                  ; IF LOOP COUNTER NOT ZERO,
6271    052540  001373                      BNE     23$                 ;   KEEP LOOPING
6272    052542                              ERRDF   24,,ERR024          ; FATAL ERROR, SA REG NOT ZERO
        052542  104455                      TRAP    C$ERDF
        052544  000030                      .WORD   24
        052546  000000                      .WORD   0
        052550  017770                      .WORD   ERR024
6273    052552  000545                      BR      41$                 ;
6274
6275    052554  005064  000002        24$:  CLR     2(R4)               ; WRITE 0 TO SA REGISTER (PURGE)
6276    052560  005714                      TST     (R4)                ; READ FROM IP REGISTER (POLL)
6277    052562  000402                      BR      26$                 ;
6278
6279    052564  005064  000002        25$:  CLR     2(R4)               ; WRITE STEP 3 RESPONSE TO SA REG
```

```
6280 052570  004737  053252        26$:    CALL    CNTRSP                  ; WAIT FOR STEP OR ERROR BIT
6281 052574  103534                         BCS     41$                     ; EXIT IF MICROCODE REPORTED FAILURE
6282 052576  010237  002352                 MOV     R2,SSTEP4               ; SAVE STEP 4 RESPONSE VALUE.
6283                                 ;
6284                                 ;       CHECK HOST COMMUNICATION AREA FOR ALL ZEROS
6285                                 ;
6286 052602  010502                         MOV     R5,R2                   ; GET FIRST ADDRESS
6287 052604  062702  000106                 ADD     #C.CINT,R2              ;   OF RING BUFFER
6288 052610  012703  000006                 MOV     #6,R3                   ; GET SIZE OF RING BUFFER
6289 052614  005722                 28$:    TST     (R2)+                   ; CHECK WORD IN BUFFER
6290 052616  001003                         BNE     29$                     ; GO TO ERROR REPORTER IF NOT ZERO
6291 052620  005303                         DEC     R3                      ; COUNT THE WORDS IN BUFFER
6292 052622  003374                         BGT     28$                     ; LOOP UNTIL ALL WORDS CHECKED
6293 052624  000405                         BR      30$                     ; START CONTROLLER AND EXIT
6294
6295 052626                         29$:    ERRDF   23,,ERRO23              ; REPORT BUFFER NOT CLEARED
     052626  104455                         TRAP    C$ERDF
     052630  000027                         .WORD   23
     052632  000000                         .WORD   0
     052634  017606                         .WORD   ERRO23
6296 052636  000513                         BR      41$                     ; ERROR EXIT
6297                                 ;
6298                                 ;       INITIALIZE RESPONSE RING, CHECK IF THE CONTROLLER TYPE IS VALID,
6299                                 ;       SET CONTROLLER DEPENDENT PARAMETERS, AND START MSCP OPERATIONS.
6300
6301 052640  005065  000152        30$:    CLR     C.CRED(R5)              ; INIT CREDITS COUNT
6302 052644  042765  020000  000014         BIC     #CT.MSG,C.FLG(R5)       ; CLEAR MESSAGE RECEIVED FLAG
6303 052652  016565  000072  000070         MOV     C.CBAS(R5),C.CPTR(R5)   ; INITIALIZE COMMAND AND
6304 052660  016565  000100  000076         MOV     C.RBAS(R5),C.RPTR(R5)   ;   RESPONSE RING POINTERS
6305 052666  012703  000004                 MOV     #4,R3
6306 052672  004737  026132        32$:    CALL    CLRRSP                  ; INITIALIZE RESPONSE RING
6307 052676  005303                         DEC     R3                      ;   WITH 4 RESPONSE PACKETS
6308 052700  001374                         BNE     32$
6309 052702  013703  002352                 MOV     SSTEP4,R3               ; GET SAVED CONTROLLER STEP 4 RESPONSE
6310 052706  006003                         ROR     R3                      ; RIGHT
6311 052710  006003                         ROR     R3                      ;       JUSTIFY
6312 052712  006003                         ROR     R3                      ;               MODEL
6313 052714  006003                         ROR     R3                      ;                     NUMBER
6314 052716  042703  177760                 BIC     #^C<SA.CNT/16.>,R3      ; R3 = CONTROLLER MODEL NUMBER
6315 052722  010365  000026                 MOV     R3,C.TYPE(R5)           ; SAVE CONTROLLER TYPE
6316 052726  022703  000006                 CMP     #6.,R3                  ; CHECK IF UDA50-A
6317 052732  001006                         BNE     34$                     ; IF NOT, BRANCH
6318 052734  012765  036413  000034         MOV     #U50MSZ,C.MSIZ(R5)      ;   ELSE,GET MEMORY SIZE
6319 052742  012700  000375                 MOV     #<374+SA.GO>,R0         ;   SET THE BURST RATE AND GO BIT
6320 052746  000440                         BR      40$                     ;   EXIT
6321 052750  022703  000015        34$:    CMP     #13.,R3                 ; CHECK IF KDA50-Q
6322 052754  001006                         BNE     36$                     ; IF NOT, BRANCH
6323 052756  012765  032413  000034         MOV     #Q50MSZ,C.MSIZ(R5)      ;   ELSE,GET MEMORY SIZE
6324 052764  012700  000035                 MOV     #<34+SA.GO>,R0          ;   SET THE BURST RATE AND GO BIT
6325 052770  000427                         BR      40$                     ;   EXIT
6326 052772  022703  000007        36$:    CMP     #7.,R3                  ; CHECK IF RQDX1
6327 052776  001006                         BNE     37$                     ; IF NOT, BRANCH
6328 053000  012765  000000  000034         MOV     #QDXMSZ,C.MSIZ(R5)      ;   ELSE,GET MEMORY SIZE
6329 053006  012700  000001                 MOV     #<SA.GO>,R0             ;   SET THE BURST RATE AND GO BIT
6330 053012  000416                         BR      40$                     ;   EXIT
```

```
6331 053014  022703  000001          37$:   CMP    #1.,R3              ; CHECK IF RUC25
6332 053020  001006                         BNE    39$                ; IF NOT, BRANCH
6333 053022  012765  000000  000034         MOV    #C25MSZ,C.MSIZ(R5) ;   ELSE,GET MEMORY SIZE
6334 053030  012700  000001                 MOV    #<SA.GO>,R0        ;   SET THE BURST RATE AND GO BIT
6335 053034  000405                          BR     40$                ;   EXIT
6336 053036  000407                  39$:   ERRDF  14.,ERR014         ;      REPORT ERROR
     053036  104455                          TRAP   C$ERDF
     053040  000016                          .WORD  14
     053042  000000                          .WORD  0
     053044  017362                          .WORD  ERR014
6337 053046  000407                          BR     41$                ;      EXIT
6338
6339 053050  010064  000002          40$:   MOV    R0,2(R4)           ; WRITE TO SA REGISTER
6340 053054                                 POP    <R3,R2>            ; RESTORE REGISTERS FROM STACK
     053054  012603                          MOV    (SP)+,R3           ; POP STACK INTO R3
     053056  012602                          MOV    (SP)+,R2           ; POP STACK INTO R2
6341 053060  000241                          CLC                       ; CLEAR Z TO INDICATE NO ERROR
6342 053062  000240                  NOP                               ; [DBG] INSERT HALT FOR DEBUG
6343 053064  000207                          RETURN                    ;
6344                                  ;
6345                                  ;      ERROR RETURN
6346                                  ;
6347 053066  010502                  41$:   MOV    R5,R2              ; GET FIRST ADDRESS
6348 053070  062702  000106                 ADD    #C.CINT,R2         ;   OF RING BUFFER
6349 053074  012703  000006                 MOV    #6.,R3             ; GET SIZE OF RING BUFFER
6350 053100  005022                  42$:   CLR    (R2)+              ; CLEAR BUFFER
6351 053102  005303                          DEC    R3                 ; COUNT THE WORDS IN BUFFER
6352 053104  003375                          BGT    42$                ; LOOP UNTIL ENTIRE BUFFER CLEARED
6353 053106                                 POP    <R3,R2>            ; RESTORE REGISTERS FROM STACK
     053106  012603                          MOV    (SP)+,R3           ; POP STACK INTO R3
     053110  012602                          MOV    (SP)+,R2           ; POP STACK INTO R2
6354 053112  004737  027350                 CALL   DRPCNT             ; DROP CONTROLLER FROM TEST
6355 053116  000240                  NOP                               ; [DBG] INSERT HALT FOR DEBUG
6356 053120  000207                          RETURN                    ;
6357                                  ;
6358                                  ;      RESPONSE CHECK FOR FIRST WORD (STEP 1) FROM SA REGISTER
6359                                  ;      CHECK IF CONTROLLER SUPPORTS ENHANCED DIAGNOSTICS
6360                                  ;
6361 053122  012701  004000          RSP.S1: MOV   #SA.S1,R1          ; SET STEP ONE BIT
6362 053126  042765  004000  000014         BIC    #CT.DI,C.FLG(R5)   ; CLEAR DI FLAG
6363 053134  032702  000400                 BIT    #SA.DI,R2          ; IF DI BIT NOT SET,
6364 053140  001403                          BEQ    1$                 ;   SKIP
6365 053142  052765  004000  000014         BIS    #CT.DI,C.FLG(R5)   ; ELSE, SET DI BIT
6366 053150  032702  004000          1$:    BIT    #SA.S1,R2          ; IF STEP 1 BIT SET,
6367 053154  001033                          BNE    RSP.SU             ;   SUCCESS
6368 053156  000423                          BR     RSP.ER             ; ELSE, ERROR
6369                                  ;
6370                                  ;      RESPONSE CHECK FOR SECOND WORD (STEP 2) FROM SA REGISTER
6371                                  ;      CHECK FOR ECHO OF INTERRUPT ENABLE FLAG AND INTERRUPT VECTOR
6372                                  ;
6373 053160  013701  002342          RSP.S2: MOV   SND.S1,R1          ; GET WORD SENT TO SA REGISTER
6374 053164  000301                          SWAB   R1                 ; GET HIGH 8 BITS
6375 053166  042701  177400                 BIC    #177400,R1         ;
6376 053172  052701  010000                 BIS    #SA.S2,R1          ; SET STEP 2 BIT
6377 053176  020102                          CMP    R1,R2              ; IF DATA RECEIVED IS CORRECT,
```

```
6378 053200  001421                      BEQ    RSP.SU              ;   DO SUCCESSFUL EXIT.
6379 053202  000411                      BR     RSP.ER              ; ELSE, DO ERROR EXIT
6380                             ;
6381                             ;       RESPONSE CHECK FOR THIRD WORD (STEP 3) FROM SA REGISTER
6382                             ;       CHECK FOR ECHO OF MESSAGE AND COMMAND RING LENGTHS
6383                             ;
6384 053204  013701  002342     RSP.S3:  MOV    SND.S1,R1           ; GET WORD SENT TO SA REGISTER
6385 053210  042701  177400              BIC    #177400,R1          ; JUST LOW 8 BITS
6386 053214  052701  020000              BIS    #SA.S3,R1           ; SET STEP 3 BIT
6387 053220  020102                      CMP    R1,R2               ; IF DATA RECEIVED IS CORRECT,
6388 053222  001410                      BEQ    RSP.SU              ;   DO SUCCESSFUL EXIT.
6389 053224  000400                      BR     RSP.ER              ; ELSE, DO ERROR EXIT
6390                             ;
6391                             ;       EXIT RESPONSE CHECK ROUTINES
6392                             ;
6393 053226              RSP.ER: ERRDF   25,,ERRO25                 ; ERROR - WRONG DATA IN SA REGISTER
     053226  104455              TRAP    C$ERDF
     053230  000031              .WORD   25
     053232  000000              .WORD   0
     053234  020046              .WORD   ERRO25
6394 053236  000261                      SEC                       ; SET CARRY TO INDICATE ERROR
6395 053240  000240              NOP                        ; [DBG] INSERT HALT FOR DEBUG
6396 053242  000207                      RETURN                     ;
6397
6398 053244  000241      RSP.SU: CLC                        ; CLEAR CARRY TO INDICATE SUCCESS
6399 053246  000240              NOP                        ; [DBG] INSERT HALT FOR DEBUG
6400 053250  000207                      RETURN                     ;
```

```
6402
6403                                    ;++
6404                                    ;      CNTRSP - WAIT FOR CONTROLLER TO RESPOND WITH DATA IN SA REGISTER.
6405                                    ;      EITHER STEP BIT FROM MASK IN LOCATION CNTRSD OR ERROR BIT
6406                                    ;      WILL CAUSE A TERMINATION.
6407                                    ;      AN ERROR MESSAGE WILL BE PRINTED IF THE CONTROLLER DOES NOT RESPOND
6408                                    ;      IN 10 SECONDS OR IF ERROR SETS.
6409                                    ;
6410                                    ;      INPUTS:
6411                                    ;               CNTRSD - MASK OF STEP BIT TO LOOK FOR
6412                                    ;               R4 - ADDRESS OF IP REGISTER
6413                                    ;               R5 - ADDRESS OF CONTROLLER TABLE
6414                                    ;      OUTPUTS:
6415                                    ;               ERROR MESSAGE IF TIME OUT ON RESPONSE OR ERROR BIT SETS
6416                                    ;               R2 - DATA FROM SA REGISTER
6417                                    ;               CARRY SET IF ERROR BIT SETS OR TIME OUT
6418                                    ;--
6419
6420 053252                            CNTRSP: PUSH    R1
     053252  010146                            MOV     R1,-(SP)              ; PUSH R1 ON STACK
6421 053254  052737  100000  002354            BIS     #SA.ERR,CNTRSD        ; SET ERROR BIT IN MASK WORD
6422 053262  012701  000012                    MOV     #10.,R1               ; SET UP FOR 10 SECOND TIMEOUT
6423 053266  010500                            MOV     R5,R0                 ; POINT TO COUNTER IN CONTROLLER TABLE
6424 053270  062700  000154                    ADD     #C.TO,R0              ;
6425 053274  004737  052070                    CALL    SETTO                 ;
6426 053300                                    POP     R1
     053300  012601                            MOV     (SP)+,R1              ; POP STACK INTO R1
6427 053302  033764  002354  000002    1$:     BIT     CNTRSD,2(R4)          ; LOOK AT ERROR AND STEP BIT
6428 053310  001026                            BNE     3$                    ; BRANCH IF EITHER SET
6429 053312  005737  002322                    TST     KW.CSR                ; SEE IF CLOCK ON SYSTEM
6430 053316  001771                            BEQ     1$
6431 053320  023765  002334  000156            CMP     KW.EL+2,C.TOH(R5)     ; CHECK IF TIME OUT OCCURRED
6432 053326  101005                            BHI     2$                    ;
6433 053330  001364                            BNE     1$                    ;
6434 053332  023765  002332  000154            CMP     KW.EL,C.TO(R5)        ;
6435 053340  103760                            BLO     1$
6436 053342  016402  000002            2$:     MOV     2(R4),R2              ; GET REGISTER CONTENTS
6437 053346  042737  100000  002354            BIC     #SA.ERR,CNTRSD        ; CLEAR ERROR BIT IN MASK WORD
6438 053354                                    ERRDF   22,,ERR022            ; REPORT TIME OUT ERROR
     053354  104455                            TRAP    C$ERDF
     053356  000026                            .WORD   22
     053360  000000                            .WORD   0
     053362  017524                            .WORD   ERR022
6439 053364  000407                            BR      4$                    ;
6440
6441                                    ;      CHECK IF ERROR BIT SET
6442                                    ;
6443 053366  016402  000002            3$:     MOV     2(R4),R2              ; GET REGISTER CONTENTS
6444 053372  100007                            BPL     5$                    ; EXIT IF ERROR NOT SET
6445 053374                                    ERRDF   21,,ERR021            ; REPORT ERROR INFO
     053374  104455                            TRAP    C$ERDF
     053376  000025                            .WORD   21
     053400  000000                            .WORD   0
     053402  017452                            .WORD   ERR021
6446 053404  000261                    4$:     SEC                           ; SET CARRY TO INDICATE ERROR
```

```
6447 053406  000240            NOP                  ; [DBG] INSERT HALT FOR DEBUG
6448 053410  000207                  RETURN                      ;
6449                           ;      NORMAL EXIT
6450                           ;
6451                           ;
6452 053412  000241            5$:    CLC                  ; CLEAR CARRY AS NO ERROR INDICATION
6453 053414  000240            NOP                  ; [DBG] INSERT HALT FOR DEBUG
6454 053416  000207                  RETURN                      ;
```

```
6456
6457                                    ;++
6458                                    ;          REINIT - RE-INITIALIZE ALL CONTROLLERS
6459                                    ;
6460                                    ;          INPUTS:
6461                                    ;                  CTABS - ADDRESS OF FIRST CONTROLLER TABLE
6462                                    ;          OUTPUTS:
6463                                    ;                  NONE
6464                                    ;--
6465
6466 053420  013705  002200            REINIT: MOV     CTABS,R5                ; GET CONTROLLER TABLE ADDRESS
6467 053424  004737  053452            1$:     CALL    RESET                   ; RE-INIT THE CONTROLLER
6468 053430  042765  050000  000014            BIC     #CT.IOC!CT.DUN,C.FLG(R5); CLEAR CONTROLLER FLAGS
6469 053436  062705  000170                    ADD     #C.SIZE,R5              ; POINT TO NEXT TABLE
6470 053442  005715                            TST     (R5)                    ; IF NOT ZERO,
6471 053444  001367                            BNE     1$                      ;     LOOK AT NEXT CONTROLLER
6472 053446  000240                    NOP                                     ; [DBG] INSERT HALT FOR DEBUG
6473 053450  000207                            RETURN                          ;
```

```
6475
6476                             ;++
6477                             ;        RESET - RESET SPECIFIED CONTROLLER
6478                             ;
6479                             ;        INPUTS:
6480                             ;            R5 - ADDRESS OF CONTROLLER TABLE
6481                             ;        OUTPUTS:
6482                             ;            NONE
6483                             ;--
6484
6485 053452  005037  002264     RESET:  CLR     NXMAD                      ; CLEAR NON-EXISTANT MEMORY ADDRESS
6486 053456                             SETVEC  #ERRVEC,#NXMI,#PRI07        ; SETUP TIMEOUT ERROR VECTOR
     053456  012746  000340             MOV     #PRI07,-(SP)
     053462  012746  054472             MOV     #NXMI,-(SP)
     053466  012746  000004             MOV     #ERRVEC,-(SP)
     053472  012746  000003             MOV     #3,-(SP)
     053476  104437                     TRAP    C$SVEC
     053500  062706  000010             ADD     #10,SP
6487 053504  005075  000000             CLR     @C.IPR(R5)                 ; RESET CONTROLLER (CLEAR IP)
6488 053510                             CLRVEC  #ERRVEC                     ; RETURN TIMEOUT ERROR VECTOR
     053510  012700  000004             MOV     #ERRVEC,R0
     053514  104436                     TRAP    C$CVEC
6489 053516  005037  002264             CLR     NXMAD                      ; CLEAR MEMORY ERROR FLAG
6490 053522  005065  000032             CLR     C.UCNT(R5)                 ; RESET UNIT UNDER TEST COUNT
6491 053526  005065  000030             CLR     C.STEP(R5)                 ; RESET TEST STEP COUNTER
6492 053532  005065  000042             CLR     C.NEXT(R5)                 ; RESET NEXT DRIVE POINTER
6493 053536  042765  024400  000014     BIC     #<CT.MSG!CT.DI!CT.MRW>,C.FLG(R5); CLEAR FLAGS
6494 053544  000240             NOP                                        ; (DBG) INSERT HALT FOR DEBUG
6495 053546  000207             RETURN                                     ;
```

```
6497
6498                                    ;++
6499                                    ;        RNTIME - PRINT RUN TIME
6500                                    ;
6501                                    ;        INPUTS:
6502                                    ;                KW.EL - CONTAINS ELAPSED TIME
6503                                    ;                KW.HZ - HERTZ OF CLOCK
6504                                    ;        OUTPUTS:
6505                                    ;                RNTHH - ELAPSED HOURS
6506                                    ;                RNTMM - ELAPSED MINUTES
6507                                    ;                RNTSS - ELAPSED SECONDS
6508                                    ;
6509                                    ;--
6510  053550                  RNTIME:  PUSH     <R1,R2,R3,R4>         ; SAVE REGISTERS
      053550   010146                  MOV      R1,-(SP)             ; PUSH R1 ON STACK
      053552   010246                  MOV      R2,-(SP)             ; PUSH R2 ON STACK
      053554   010346                  MOV      R3,-(SP)             ; PUSH R3 ON STACK
      053556   010446                  MOV      R4,-(SP)             ; PUSH R4 ON STACK
6511  053560   013701   002332         MOV      KW.EL,R1             ; GET ELAPSED TIME
6512  053564   013702   002334         MOV      KW.EL+2,R2           ;
6513  053570   005003                  CLR      R3                   ;
6514  053572   013704   002330         MOV      KW.HZ,R4             ; GET SPEED OF CLOCK
6515  053576   004737   047256         CALL     DIVIDE               ; CONVERT FROM TICKS TO SECONDS
6516  053602   012704   000074         MOV      #60.,R4              ; NOW DIVIDE BY 60
6517  053606   004737   047256         CALL     DIVIDE               ;   TO CONVERT TO MINUTES
6518  053612   010437   002272         MOV      R4,RNTSS             ; SAVE REMAINDER AS SECONDS
6519  053616   012704   000074         MOV      #60.,R4              ; NOW DIVIDE BY 60
6520  053622   004737   047256         CALL     DIVIDE               ;   TO CONVERT TO  HOURS
6521  053626   010137   002266         MOV      R1,RNTHH             ; SAVE QUOTIENT AS HOURS
6522  053632   010437   002270         MOV      R4,RNTMM             ; SAVE REMAINDER AS MINUTES
6523  053636                           POP      <R4,R3,R2,R1>        ; RESTORE SAVED REGISTERS
      053636   012604                  MOV      (SP)+,R4             ; POP STACK INTO R4
      053640   012603                  MOV      (SP)+,R3             ; POP STACK INTO R3
      053642   012602                  MOV      (SP)+,R2             ; POP STACK INTO R2
      053644   012601                  MOV      (SP)+,R1             ; POP STACK INTO R1
6524  053646   000240          NOP                                  ; [DBG] INSERT HALT FOR DEBUG
6525  053650   000207          RETURN                               ;
```

```
6527
6528                                         ;++
6529                                         ;          WCHNG - WAIT UNTIL WRAPPED DATA APPEARS IN SA REGISTER
6530                                         ;
6531                                         ;          INPUTS:
6532                                         ;                  R4 - IP REGISTER
6533                                         ;                  R5 - ADDRESS OF CONTROLLER TABLE
6534                                         ;--
6535
6536  053652                                 WCHNG:  PUSH    <R1>              ; SAVE REGISTERS
      053652   010146                                MOV     R1,-(SP)          ; PUSH R1 ON STACK
6537  053654   012701   000012                       MOV     #10.,R1           ; SET TIMEOUT FOR 10 SECONDS
6538  053660   010500                                MOV     R5,R0             ; POINT TO CONTROLLER TABLE
6539  053662   062700   000154                       ADD     #C.TO,R0          ;
6540  053666   004737   052070                       CALL    SETTO             ;
6541  053672   016402   000002              1$:      MOV     2(R4),R2          ; GET SA CONTENTS
6542  053676   020237   002356                       CMP     R2,WCHNGD         ; IS LOOPED DATA IN SA REGISTER?
6543  053702   001423                                BEQ     3$                ;
6544  053704   005737   002322                       TST     KW.CSR            ; SEE IF CLOCK ON SYSTEM
6545  053710   001770                                BEQ     1$                ;
6546  053712   023765   002333   000156              CMP     KW.EL+2,C.TOH(R5) ; CHECK IF TIME OUT OCCURRED
6547  053720   101005                                BHI     2$                ;
6548  053722   001363                                BNE     1$                ;
6549  053724   023765   002332   000154              CMP     KW.EL,C.TO(R5)    ;
6550  053732   103757                                BLO     1$                ;
6551  053734                              2$:      ERRDF   27,,ERR027        ; REPORT ERROR
      053734   104455                                TRAP    C$ERDF
      053736   000033                                .WORD   27
      053740   000000                                .WORD   0
      053742   020210                                .WORD   ERR027
6552  053744   004737   027350                       CALL    DRPCNT            ; DROP CONTROLLER FROM TESTING
6553  053750   000401                                BR      4$
6554  053752   000241              3$:      CLC                       ; SUCCESSFUL EXIT
6555  053754                              4$:      POP     <R1>              ; RESTORE SAVED REGISTER
      053754   012601                                MOV     (SP)+,R1          ; POP STACK INTO R1
6556  053756   000240              NOP                       ; [DBG] INSERT HALT FOR DEBUG
6557  053760   000207                       RETURN                    ; RETURN TO CALLING PROGRAM
6558
```

```
6560                                    .SBTTL  PRE-PROGRAMMED MESSAGE SUBROUTINES
6561
6562                                    ;++
6563                                    ;       CALR3 - PRE-PROGRAMMED PRINT ROUTINE 3
6564                                    ;
6565                                    ;       PRINT "FATAL CONTROLLER ERROR"
6566                                    ;--
6567
6568 053762                    CALR3:   PRINTB   #XFCE                        ; PRINT MESSAGE
     053762  012746  016543             MOV      #XFCE,-(SP)
     053766  012746  000001             MOV      #1,-(SP)
     053772  010600                      MOV      SP,R0
     053774  104414                      TRAP     C$PNTB
     053776  062706  000004             ADD      #4,SP
6569 054002  000240             NOP                                   ; [DBG] INSERT HALT FOR DEBUG
6570 054004  000207                      RETURN
6571
6572                                    ;++
6573                                    ;       CALR4 - PRE-PROGRAMMED PRINT ROUTINE 4
6574                                    ;
6575                                    ;       PRINT BASIC LINE FOR HOST PROGRAM ERROR WITHOUT CONTROLLER ADDRESS
6576                                    ;       OR DRIVE NUMBER.
6577                                    ;--
6578
6579 054006                    CALR4:   PUSH     <R3>                         ; SAVE REGISTERS
6580 054010  013703  002206             MOV      TNUM,R3                      ; GET TEST NUMBER
6581 054014  006303                      ASL      R3                           ; CONVERT TO WORD OFFSET
6582 054016                              PRINTB   #BASLO,TNAMES(R3)            ; PRINT MESSAGE
     054016  016346  007244             MOV      TNAMES(R3),-(SP)
     054022  012746  007403             MOV      #BASLO,-(SP)
     054026  012746  000002             MOV      #2,-(SP)
     054032  010600                      MOV      SP,R0
     054034  104414                      TRAP     C$PNTB
     054036  062706  000006             ADD      #6,SP
6583 054042  005737  002322             TST      KW.CSR                       ; CHECK FOR CLOCK ON SYSTE
6584 054046  001420                      BEQ      1$                           ; IF ZERO, SKIP
6585 054050  004737  053550             CALL     RNTIME                       ; GET RUN TIME
6586 054054                              PRINTB   #RNTIM,RNTHH,RNTMM,RNTSS; PRINT RUN TIME
     054054  013746  002272             MOV      RNTSS,-(SP)
     054060  013746  002270             MOV      RNTMM,-(SP)
     054064  013746  002266             MOV      RNTHH,-(SP)
     054070  012746  010303             MOV      #RNTIM,-(SP)
     054074  012746  000004             MOV      #4,-(SP)
     054100  010600                      MOV      SP,R0
     054102  104414                      TRAP     C$PNTB
     054104  062706  000012             ADD      #12,SP
6587 054110                    1$:      PRINTB   #CRLF                        ; TERMINATE LINE
     054110  012746  010300             MOV      #CRLF,-(SP)
     054114  012746  000001             MOV      #1,-(SP)
     054120  010600                      MOV      SP,R0
     054122  104414                      TRAP     C$PNTB
     054124  062706  000004             ADD      #4,SP
6588 054130                              POP      <R3>                         ; RESTORE SAVED REGISTER
6589 054132  000240             NOP                                   ; [DBG] INSERT HALT FOR DEBUG
6590 054134  000207                      RETURN
```

```
6591
6592                                          ;++          CALR5 - PRE-PROGRAMMED PRINT ROUTINE 5
6593                                          ;
6594                                          ;
6595                                          ;            PRINT BASIC LINE FOR HOST PROGRAM ERROR WITH CONTROLLER ADDRESS
6596                                          ;            AND NO DRIVE NUMBER
6597                                          ;
6598                                          ;            INPUTS:
6599                                          ;                  R5 - CONTROLLER CSR
6600                                          ;--
6601
6602 054136                                   CALR5:  PUSH    <R3>                    ; SAVE REGISTERS
6603 054140   013703   002206                         MOV     TNUM,R3                 ; GET TEST NUMBER
6604 054144   006303                                  ASL     R3                      ; CONVERT TO WORD OFFSET
6605 054146                                           PRINTB  #BASL1,TNAMES(R3),(R5)  ; PRINT MESSAGE
     054146   011546                                  MOV     (R5),-(SP)
     054150   016346   007244                         MOV     TNAMES(R3),-(SP)
     054154   012746   007406                         MOV     #BASL1,-(SP)
     054160   012746   000003                         MOV     #3,-(SP)
     054164   010600                                  MOV     SP,R0
     054166   104414                                  TRAP    C$PNTB
     054170   062706   000010                         ADD     #10,SP
6606 054174   005737   002322                         TST     KW.CSR                  ; CHECK FOR CLOCK ON SYSTE
6607 054200   001420                                  BEQ     1$                      ; IF ZERO, SKIP
6608 054202   004737   053550                         CALL    RNTIME                  ; GET RUN TIME
6609 054206                                           PRINTB  #RNTIM,RNTHH,RNTMM,RNTSS; PRINT RUN TIME
     054206   013746   002272                         MOV     RNTSS,-(SP)
     054212   013746   002270                         MOV     RNTMM,-(SP)
     054216   013746   002266                         MOV     RNTHH,-(SP)
     054222   012746   010303                         MOV     #RNTIM,-(SP)
     054226   012746   000004                         MOV     #4,-(SP)
     054232   010600                                  MOV     SP,R0
     054234   104414                                  TRAP    C$PNTB
     054236   062706   000012                         ADD     #12,SP
6610 054242                                   1$:     PRINTB  #CRLF                   ; TERMINATE LINE
     054242   012746   010300                         MOV     #CRLF,-(SP)
     054246   012746   000001                         MOV     #1,-(SP)
     054252   010600                                  MOV     SP,R0
     054254   104414                                  TRAP    C$PNTB
     054256   062706   0C0004                         ADD     #4,SP
6611 054262                                           POP     <R3>                    ; RESTORE SAVED REGISTER
6612 054264   000240                           NOP                                    ; [DBG] INSERT HALT FOR DEBUG
6613 054266   000207                           RETURN
6614
6615
6616                                          ;++          CALR6 - PRE-PROGRAMMED PRINT ROUTINE 6
6617                                          ;
6618                                          ;            PRINT BASIC LINE FOR HOST PROGRAM ERROR WITH CONTROLLER ADDRESS
6619                                          ;            AND DRIVE NUMBER
6620                                          ;
6621                                          ;            INPUTS:
6622                                          ;                  (R1) - DRIVE NUMBER
6623                                          ;                  (R5) - CONTROLLER CSR
6624                                          ;--
6625
```

CZUDJAO UDA50-A/KDA50-Q SUBSY E MACRO V05.03 Wednesday 02-Oct-85 16:03 Page 128-2
PRE-PROGRAMMED MESSAGE SUBROUTINES

```
6626 054270                        CALR6:  PUSH    <R3>                    ; SAVE REGISTERS
6627 054272  013703  002206                MOV     TNUM,R3                 ; GET TEST NUMBER
6628 054276  006303                         ASL     R3                      ; CONVERT TO WORD OFFSET
6629 054300                                 PRINTB  #BASL2,TNAMES(R3),(R5),(R1); PRINT MESSAGE
     054300  011146                         MOV     (R1),-(SP)
     054302  011546                         MOV     (R5),-(SP)
     054304  016346  007244                 MOV     TNAMES(R3),-(SP)
     054310  012746  007437                 MOV     #BASL2,-(SP)
     054314  012746  000004                 MOV     #4,-(SP)
     054320  010600                         MOV     SP,R0
     054322  104414                         TRAP    C$PNTB
     054324  062706  000012                 ADD     #12,SP
6630 054330  005737  002322                 TST     KW.CSR                  ; CHECK FOR CLOCK ON SYSTE
6631 054334  001420                         BEQ     1$                      ; IF ZERO, SKIP
6632 054336  004737  053550                 CALL    RNTIME                  ; GET RUN TIME
6633 054342                                 PRINTB  #RNTIM,RNTHH,RNTMM,RNTSS; PRINT RUN TIME
     054342  013746  002272                 MOV     RNTSS,-(SP)
     054346  013746  002270                 MOV     RNTMM,-(SP)
     054352  013746  002266                 MOV     RNTHH,-(SP)
     054356  012746  010303                 MOV     #RNTIM,-(SP)
     054362  012746  000004                 MOV     #4,-(SP)
     054366  010600                         MOV     SP,R0
     054370  104414                         TRAP    C$PNTB
     054372  062706  000012                 ADD     #12,SP
6634 054376                        1$:      PRINTB  #CRLF                   ; TERMINATE LINE
     054376  012746  010300                 MOV     #CRLF,-(SP)
     054402  012746  000001                 MOV     #1,-(SP)
     054406  010600                         MOV     SP,R0
     054410  104414                         TRAP    C$PNTB
     054412  062706  000004                 ADD     #4,SP
6635 054416                                 POP     <R3>                    ; RESTORE SAVED REGISTER
6636 054420  000240               NOP                                       ; [DBG] INSERT HALT FOR DEBUG
6637 054422  000207                         RETURN
6638
6639                               ;++
6640                               ;       CALR7 - PRE-PROGRAMMED PRINT ROUTINE 7
6641                               ;
6642                               ;       PRINT "REPLACE CONTROLLER PROCESSOR MODULE"
6643                               ;--
6644
6645 054424                        CALR7:  PRINTX  #XFRU                    ; PRINT MESSAGE
     054424  012746  016743                 MOV     #XFRU,-(SP)
     054430  012746  000001                 MOV     #1,-(SP)
     054434  010600                         MOV     SP,R0
     054436  104415                         TRAP    C$PNTX
     054440  062706  000004                 ADD     #4,SP
6646 054444  000207                         RETURN                          ;
6647
6648                               ;++
6649                               ;       CALR8 - PRE-PROGRAMMED PRINT ROUTINE 8
6650                               ;
6651                               ;       PRINT  "SA REGISTER CONTAINS  XXXXXX"
6652                               ;--
6653
6654 054446                        CALR8:  PRINTX  #XSA,R2                  ; PRINT MESSAGE
```

```
        054446  010246              MOV     R2,-(SP)
        054450  012746  016705      MOV     #XSA,-(SP)
        054454  012746  000002      MOV     #2,-(SP)
        054460  010600              MOV     SP,R0
        054462  104415              TRAP    C$PNTX
        054464  062706  000006      ADD     #6,SP
6655    054470  000207              RETURN                      ;
6656
```

```
 6658                                              .SBTTL INTERRUPT SERVICE ROUTINES
 6659
 6660                                      ;++
 6661                                      ;          NXMI - NON-EXISTANT MEMORY SERVICE ROUTINE
 6662
 6663                                      ;      INPUTS:
 6664                                      ;             NXMAD SET TO ZERO
 6665                                      ;      OUTPUTS:
 6666                                      ;             NXMAD SET TO ONES IF NON-EXISTANT TRAP OCCURED
 6667                                      ;--
 6668
 6669 054472                              BGNSRV  NXMI
      054472                              NXMI::
 6670 054472  012737  177777  002264      MOV      #-1,NXMAD
 6671 054500                              ENDSRV
      054500                              L10043:
      054500  000002                      RTI
 6672
 6673                                      ;++
 6674                                      ;          CNTSRV - CONTROLLER INTERRUPT SERVICE ROUTINE. MARKS CONTROLLER TABLE
 6675                                      ;          THAT AN INTERRUPT HAS BEEN RECEIVED.
 6676
 6677                                      ;          THIS ROUTINE IS CALLED BY A [JSR R0,CNTSRV] INSTRUCTION FROM WITHIN
 6678                                      ;          THE CONTROLLER TABLE. THE PC STORED IN R0 IS THE ADDRESS OF THE C.FLG
 6679                                      ;          WORD IN THE CONTROLLER TABLE. THE STACK CONTAINS THE SAVED CONTENTS
 6680                                      ;          OF R0 FOLLOWED BY THE INTERRUPTED PC AND PS.
 6681
 6682                                      ;      INPUTS:
 6683                                      ;             R0 - ADDRESS OF C.FLG WORD IN CONTROLLER TABLE
 6684                                      ;             STACK - SAVED CONTENTS OF R0
 6685                                      ;      OUTPUTS:
 6686                                      ;             CT.MSG SET IN C.FLG WORD OF CONTROLLER TABLE
 6687                                      ;             R0 - RESTORED FROM STACK
 6688                                      ;--
 6689
 6690 054502                              BGNSRV  CNTSRV
      054502                              CNTSRV::
 6691 054502  052710  020000              BIS      #CT.MSG,(R0)           ; SET CT.MSG
 6692 054506                              POP      R0                     ; RESTORE R0
      054506  012600                      MOV      (SP)+,R0               ; POP STACK INTO R0
 6693 054510                              ENDSRV
      054510                              L10044:
      054510  000002                      RTI
 6694
 6695                                      ;++
 6696                                      ;          KW11I - CLOCK INTERRUPT SERVICE ROUTINE
 6697                                      ;--
 6698 054512                              BGNSRV  KW11I
      054512                              KW11I::
 6699 054512  062737  000001  002332      ADD      #1,KW.EL               ; COUNT THE INTERRUPT
 6700 054520  005537  002334              ADC      KW.EL+2                ; PUT CARRY IN HIGH WORD
 6701 054524  012777  000105  125570      MOV      #KW.OUT,@KW.CSR        ; RESTART THE CLOCK
 6702 054532                              ENDSRV
      054532                              L10045:
      054532  000002                      RTI
```

6703

```
6705                                    .SBTTL   REPORT CODING SECTION
6706
6707                            ;++
6708                            ;        THE REPORT CODING SECTION CONTAINS THE CODE FOR PRINTING
6709                            ;        STATISTICAL INFORMATION GATHERED BY THE DIAGNOSTIC.  IT IS
6710                            ;        EXECUTED BY THE OPERATOR COMMAND "PRINT" OR BY THE MACRO CALL
6711                            ;        "DORPT".
6712                            ;--
6713  054534                            BGNRPT
      054534                    L$RPT::

6714
6715  054534                            PUSH     <R1,R2,R3,R4,R5>
      054534   010146                   MOV      R1,-(SP)               ; PUSH R1 ON STACK
      054536   010246                   MOV      R2,-(SP)               ; PUSH R2 ON STACK
      054540   010346                   MOV      R3,-(SP)               ; PUSH R3 ON STACK
      054542   010446                   MOV      R4,-(SP)               ; PUSH R4 ON STACK
      054544   010546                   MOV      R5,-(SP)               ; PUSH R5 ON STACK
6716  054546                            PRINTS   #RPTHDR,TNUM           ; PRINT TEST NUMBER
      054546   013746   002206          MOV      TNUM,-(SP)
      054552   012746   057300          MOV      #RPTHDR,-(SP)
      054556   012746   000002          MOV      #2,-(SP)
      054562   010600                   MOV      SP,R0
      054564   104416                   TRAP     C$PNTS
      054566   062706   000006          ADD      #6,SP
6717  054572   005737   002322          TST      KW.CSR                 ; CHECK FOR CLOCK ON SYSTE
6718  054576   001420                   BEQ      10$                    ; IF ZERO, SKIP
6719  054600   004737   053550          CALL     RNTIME                 ; GET RUN TIME
6720  054604                            PRINTS   #RNTIM,RNTHH,RNTMM,RNTSS; PRINT RUN TIME
      054604   013746   002272          MOV      RNTSS,-(SP)
      054610   013746   002270          MOV      RNTMM,-(SP)
      054614   013746   002266          MOV      RNTHH,-(SP)
      054620   012746   010303          MOV      #RNTIM,-(SP)
      054624   012746   000004          MOV      #4,-(SP)
      054630   010600                   MOV      SP,R0
      054632   104416                   TRAP     C$PNTS
      054634   062706   000012          ADD      #12,SP
6721  054640                    10$:    PRINTS   #CRLF                  ; TERMINATE LINE
      054640   012746   010300          MOV      #CRLF,-(SP)
      054644   012746   000001          MOV      #1,-(SP)
      054650   010600                   MOV      SP,R0
      054652   104416                   TRAP     C$PNTS
      054654   062706   000004          ADD      #4,SP
6722  054660   012700   002250          MOV      #STIME,R0              ; GET REPORT TIMER STORAGE
6723  054664   013701   002150          MOV      SFPTBL+SO.SRI,R1       ; GET REPORT INTERVAL
6724  054670   004737   052144          CALL     SETMIN                 ; SET TIME FOR NEXT REPORT
6725
6726  054674                            PRINTS   #RPTIO1                ; PRINT
      054674   012746   057333          MOV      #RPTIO1,-(SP)
      054700   012746   000001          MOV      #1,-(SP)
      054704   010600                   MOV      SP,R0
      054706   104416                   TRAP     C$PNTS
      054710   062706   000004          ADD      #4,SP
6727  054714                            PRINTS   #RPTIO2                ;     SUBSYSTEM I/O
      054714   012746   057370          MOV      #RPTIO2,-(SP)
      054720   012746   000001          MOV      #1,-(SP)
```

```
              054724  010600                       MOV     SP,R0
              054726  104416                       TRAP    C$PNTS
              054730  062706   000004              ADD     #4,SP
       6728   054734                               PRINTS  #RPTIO3                  ;      SUMMARY HEADING
              054734  012746   057505              MOV     #RPTIO3,-(SP)
              054740  012746   000001              MOV     #1,-(SP)
              054744  010600                       MOV     SP,R0
              054746  104416                       TRAP    C$PNTS
              054750  062706   000004              ADD     #4,SP
       6729   054754  013705   002200              MOV     CTABS,R5                 ; GET ADDRESS OF 1ST CONTROLLER TABLE
       6730
       6731   054760  005765   000056      20$:    TST     C.XFHR(R5)               ; IF NON ZERO,
       6732   054764  001011                       BNE     21$                      ;    CALCULATE MEGA BYTES
       6733   054766  022765   000017 000054       CMP     #17,C.XFMR(R5)           ; IF BYTE COUNT < 1 MB,
       6734   054774  101033                       BHI     22$                      ;    CALCULATE KILOBYTES
       6735   054776  103404                       BLO     21$
       6736   055000  022765   041100 000052       CMP     #41100,C.XFLR(R5)        ;
       6737   055006  101026                       BHI     22$                      ;
       6738   055010                       21$:    PUSH    <R1,R5>                  ; SAVE REGISTERS
       6739   055014  016503   000056              MOV     C.XFHR(R5),R3            ; CALCULATE
       6740   055020  016502   000054              MOV     C.XFMR(R5),R2            ;    MEGA
       6741   055024  016501   000052              MOV     C.XFLR(R5),R1            ;       BYTE
       6742   055030  012704   041100              MOV     #41100,R4                ;          COUNT
       6743   055034  012705   000017              MOV     #17,R5                   ;
       6744   055040  004737   047326              CALL    DIV48                    ;
       6745   055044  012737   060005 002302       MOV     #RPTMB,RPTRF             ; SET M BYTE FLAG
       6746   055052  010137   002300              MOV     R1,RPTRV                 ; SAVE REMAINING
       6747   055056                               POP     <R5,R1>                  ; RESTORE REGISTERS
       6748   055062  000442                       BR      25$                      ;
       6749   055064  005765   000054      22$:    TST     C.XFMR(R5)               ; IF NON ZERO,
       6750   055070  001004                       BNE     23$                      ;    CALCULATE KILO BYTES
       6751   055072  022765   001750 000052       CMP     #1000.,C.XFLR(R5)        ; IF BYTE COUNT < 1 KB,
       6752   055100  101025                       BHI     24$                      ;    CALCULATE BYTES
       6753   055102                       23$:    PUSH    <R1,R5>                  ; SAVE REGISTERS
       6754   055106  016503   000056              MOV     C.XFHR(R5),R3            ;
       6755   055112  016502   000054              MOV     C.XFMR(R5),R2            ; CALCULATE
       6756   055116  016501   000052              MOV     C.XFLR(R5),R1            ;    KILO
       6757   055122  012704   001750              MOV     #1000.,R4                ;       BYTE
       6758   055126  005005                       CLR     R5                       ;          COUNT
       6759   055130  004737   047326              CALL    DIV48                    ;
       6760   055134  012737   060003 002302       MOV     #RPTKB,RPTRF             ; SET M BYTE FLAG
       6761   055142  010137   002300              MOV     R1,RPTRV                 ; SAVE REMAINING
       6762   055146                               POP     <R5,R1>                  ; RESTORE REGISTERS
       6763   055152  000406                       BR      25$                      ;
       6764   055154  012737   060001 002302 24$:  MOV     #RPTB,RPTRF              ; SET BYTE FLAG
       6765   055162  016537   000052 002300       MOV     C.XFLR(R5),RPTRV         ;
       6766   055170  005765   000050      25$:    TST     C.XFHW(R5)               ; IF NON ZERO,
       6767   055174  001011                       BNE     26$                      ;    CALCULATE MEGA BYTES
       6768   055176  022765   000017 000046       CMP     #17,C.XFMW(R5)           ; IF BYTE COUNT < 1 MB,
       6769   055204  101033                       BHI     27$                      ;    CALCULATE KILOBYTES
       6770   055206  103404                       BLO     26$                      ;
       6771   055210  022765   041100 000044       CMP     #41100,C.XFLW(R5)        ;
       6772   055216  101026                       BHI     27$                      ;
       6773   055220                       26$:    PUSH    <R1,R5>                  ; SAVE REGISTERS
       6774   055224  016503   000050              MOV     C.XFHW(R5),R3            ;
```

```
6775 055230 016502 000046             MOV    C.XFMW(R5),R2           ; CALCULATE
6776 055234 016501 000044             MOV    C.XFLW(R5),R1           ;   MEGA
6777 055240 012704 041100             MOV    #41100,R4              ;     BYTE
6778 055244 012705 000017             MOV    #17,R5                 ;       COUNT
6779 055250 004737 047326             CALL   DIV48
6780 055254 012737 060005 002276      MOV    #RPTMB,RPTWF           ; SET M BYTE FLAG
6781 055262 010137 002274             MOV    R1,RPTWV              ; SAVE REMAINING
6782 055266                           POP    <R5,R1>               ; RESTORE REGISTERS
6783 055272 000442                    BR     29$                   ;
6784 055274 005765 000046      27$:   TST    C.XFMW(R5)            ; IF NON ZERO,
6785 055300 001004                    BNE    271$                  ;   CALCULATE KILO BYTES
6786 055302 022765 001750 000044      CMP    #1000.,C.XFLW(R5)     ; IF BYTE COUNT < 1 KB,
6787 055310 101025                    BHI    28$                   ;   CALCULATE BYTES
6788 055312                    271$:  PUSH   <R1,R5>               ; SAVE REGISTERS
6789 055316 016503 000050             MOV    C.XFHW(R5),R3         ;
6790 055322 016502 000046             MOV    C.XFMW(R5),R2         ; CALCULATE
6791 055326 016501 000044             MOV    C.XFLW(R5),R1         ;   KILO
6792 055332 012704 001750             MOV    #1000.,R4            ;     BYTE
6793 055336 005005                    CLR    R5                   ;       COUNT
6794 055340 004737 047326             CALL   DIV48
6795 055344 012737 060003 002276      MOV    #RPTKB,RPTWF         ; SET M BYTE FLAG
6796 055352 010137 002274             MOV    R1,RPTWV            ; SAVE REMAINING
6797 055356                           POP    <R5,R1>             ; RESTORE REGISTERS
6798 055362 000406                    BR     29$                 ;
6799 055364 012737 060001 002276 28$: MOV    #RPTB,RPTWF
6800 055372 016537 000052 002274      MOV    C.XFLR(R5),RPTWV    ;

6801
6802 055400 016502 000062      29$:   MOV    C.UID+2(R5),R2       ; GET
6803 055404 016503 000064             MOV    C.UID+4(R5),R3       ;   UNIT
6804 055410 016501 000060             MOV    C.UID(R5),R1         ;     ID
6805 055414 012704 000017             MOV    #15.,R4             ; GET CHARACTER COUNT FOR STRING
6806 055420 004737 047472             CALL   ITOA                ; CONVERT BINARY TO ASCII
6807 055424 116503 000066             MOVB   C.UID+6(R5),R3       ; GET MODEL TYPE BYTE
6808 055430 116504 000067             MOVB   C.UID+7.(R5),R4     ; GET CLASS TYPE BYTE

6809
6810 055434                           PRINTS #RPTIOC.(R5),R3,R4,#TEMP,RPTWV,RPTWF,RPTRV,RPTRF
     055434 013746 002302             MOV    RPTRF,-(SP)
     055440 013746 002300             MOV    RPTRV,-(SP)
     055444 013746 002276             MOV    RPTWF,-(SP)
     055450 013746 002274             MOV    RPTWV,-(SP)
     055454 012746 002212             MOV    #TEMP,-(SP)
     055460 010446                    MOV    R4,-(SP)
     055462 010346                    MOV    R3,-(SP)
     055464 011546                    MOV    (R5),-(SP)
     055466 012746 057623             MOV    #RPTIOC,-(SP)
     055472 012746 000011             MOV    #11,-(SP)
     055476 010600                    MOV    SP,R0
     055500 104416                    TRAP   C$PNTS
     055502 062706 000024             ADD    #24,SP
6811
6812 055506 010504                    MOV    R5,R4               ; GET ADDRESS OF CONTROLLER TABLE
6813 055510 062704 000016             ADD    #C.DRO,R4           ; POINT TO DRIVE TABLE POINTERS
6814 055514 012703 000004             MOV    #4.,R3              ; GET COUNT OF DRIVES
6815 055520 012401             30$:   MOV    (R4)+,R1            ; LOOK AT POINTER
6816 055522 001002                    BNE    40$                 ;
```

```
6817  055524  000137  056734                JMP     90$                 ; GO TO NEXT CONTROLLER IF NO TABLE
6818  055530                      40$:       PUSH    <R3,R4,R5>          ; SAVE REGISTERS
      055530  010346                         MOV     R3,-(SP)            ; PUSH R3 ON STACK
      055532  010446                         MOV     R4,-(SP)            ; PUSH R4 ON STACK
      055534  010546                         MOV     R5,-(SP)            ; PUSH R5 ON STACK
6819  055536  005761  000074                 TST     D.XFHR(R1)          ; IF NON ZERO,
6820  055542  001011                         BNE     42$                 ;    CALCULATE MEGA BYTES
6821  055544  022761  000017  000072         CMP     #17,D.XFMR(R1)      ; IF BYTE COUNT < 1 MB,
6822  055552  101031                         BHI     43$                 ;    CALCULATE KILOBYTES
6823  055554  103404                         BLO     42$
6824  055556  022761  041100  000070         CMP     #41100,D.XFLR(R1)
6825  055564  101024                         BHI     43$                 ;
6826  055566                      42$:       PUSH    <R1>                ; SAVE REGISTERS
      055566  010146                         MOV     R1,-(SP)            ; PUSH R1 ON STACK
6827  055570  016103  000074                 MOV     D.XFHR(R1),R3       ;
6828  055574  016102  000072                 MOV     D.XFMR(R1),R2       ; CALCULATE
6829  055600  016101  000070                 MOV     D.XFLR(R1),R1       ;    MEGA
6830  055604  012704  041100                 MOV     #41100,R4           ;       BYTE
6831  055610  012705  000017                 MOV     #17,R5              ;          COUNT
6832  055614  004737  047326                 CALL    DIV48               ;
6833  055620  012737  060005  002302         MOV     #RPTMB,RPTRF        ; SET M BYTE FLAG
6834  055626  010137  002300                 MOV     R1,RPTRV            ; SAVE REMAINING
6835  055632                      POP     <R1>                ; RESTORE REGISTERS
      055632  012601                          MOV     (SP)+,R1            ; POP STACK INTO R1
6836  055634  000440                          BR      50$                 ;
6837  055636  005761  000072      43$:       TST     D.XFMR(R1)          ; IF NON ZERO,
6838  055642  001004                          BNE     44$                 ;    CALCULATE KILO BYTES
6839  055644  022761  001750  000070         CMP     #1000.,D.XFLR(R1)   ; IF BYTE COUNT < 1 KB,
6840  055652  101023                          BHI     45$                 ;    CALCULATE BYTES
6841  055654                      44$:       PUSH    <R1>                ; SAVE REGISTERS
      055654  010146                          MOV     R1,-(SP)            ; PUSH R1 ON STACK
6842  055656  016103  000074                 MOV     D.XFHR(R1),R3       ;
6843  055662  016102  000072                 MOV     D.XFMR(R1),R2       ; CALCULATE
6844  055666  016101  000070                 MOV     D.XFLR(R1),R1       ;    KILO
6845  055672  012704  001750                 MOV     #1000.,R4           ;       BYTE
6846  055676  005005                         CLR     R5                  ;          COUNT
6847  055700  004737  047326                 CALL    DIV48               ;
6848  055704  012737  060003  002302         MOV     #RPTKB,RPTRF        ; SET M BYTE FLAG
6849  055712  010137  002300                 MOV     R1,RPTRV            ; SAVE REMAINING
6850  055716                      POP     <R1>                ; RESTORE REGISTERS
      055716  012601                          MOV     (SP)+,R1            ; POP STACK INTO R1
6851  055720  000406                          BR      50$                 ;
6852  055722  012737  060001  002302  45$:   MOV     #RPTB,RPTRF         ;
6853  055730  016137  000070  002300         MOV     D.XFLR(R1),RPTRV    ;
6854  055736  005761  000066      50$:       TST     D.XFHW(R1)          ; IF NON ZERO,
6855  055742  001011                          BNE     52$                 ;    CALCULATE MEGA BYTES
6856  055744  022761  000017  000064         CMP     #17,D.XFMW(R1)      ; IF BYTE COUNT < 1 MB,
6857  055752  101031                          BHI     53$                 ;    CALCULATE KILOBYTES
6858  055754  103404                          BLO     52$
6859  055756  022761  041100  000062         CMP     #41100,D.XFLW(R1)
6860  055764  101024                          BHI     53$                 ;
6861  055766                      52$:       PUSH    <R1>                ; SAVE REGISTERS
      055766  010146                          MOV     R1,-(SP)            ; PUSH R1 ON STACK
6862  055770  016103  000066                 MOV     D.XFHW(R1),R3       ;
6863  055774  016102  000064                 MOV     D.XFMW(R1),R2       ; CALCULATE
```

```
6864 056000  016101  000062                   MOV    D.XFLW(R1),R1         ;    MEGA
6865 056004  012704  041100                   MOV    #41100,R4             ;      BYTE
6866 056010  012705  000017                   MOV    #17,R5                ;        COUNT
6867 056014  004737  047326                   CALL   DIV48                 ; SET M BYTE FLAG
6868 056020  012737  060005  002276           MOV    #RPTMB,RPTWF          ; SAVE REMAINING
6869 056026  010137  002274                   MOV    R1,RPTWV              ; RESTORE REGISTERS
6870 056032                                   POP    <R1>                  ; POP STACK INTO R1
     056032  012601                           MOV    (SP)+,R1
6871 056034  000440                           BR     60$                   ;
6872 056036  005761  000064           53$:    TST    D.XFMW(R1)            ; IF NON ZERO,
6873 056042  001004                           BNE    54$                   ;    CALCULATE KILO BYTES
6874 056044  022761  001750  000062           CMP    #1000.,D.XFLW(R1)     ; IF BYTE COUNT < 1 KB,
6875 056052  101023                           BHI    55$                   ;    CALCULATE BYTES
6876 056054                           54$:    PUSH   <R1>                  ; SAVE REGISTERS
     056054  010146                           MOV    R1,-(SP)              ; PUSH R1 ON STACK
6877 056056  016103  000066                   MOV    D.XFHW(R1),R3         ;
6878 056062  016102  000064                   MOV    D.XFMW(R1),R2         ; CALCULATE
6879 056066  016101  000062                   MOV    D.XFLW(R1),R1         ;   KILO
6880 056072  012704  001750                   MOV    #1000.,R4             ;     BYTE
6881 056076  005005                           CLR    R5                    ;       COUNT
6882 056100  004737  047326                   CALL   DIV48                 ;
6883 056104  012737  060003  002276           MOV    #RPTKB,RPTWF          ; SET M BYTE FLAG
6884 056112  010137  002274                   MOV    R1,RPTWV              ; SAVE REMAINING
6885 056116                                   POP    <R1>                  ; RESTORE REGISTERS
     056116  012601                           MOV    (SP)+,R1              ; POP STACK INTO R1
6886 056120  000406                           BR     60$                   ;
6887 056122  012737  060001  002276   55$:    MOV    #RPTB,RPTWF           ;
6888 056130  016137  000062  002274           MOV    D.XFLW(R1),RPTWV      ;
6889 056136  005761  000102           60$:    TST    D.XFMA(R1)            ; IF NON ZERO,
6890 056142  001011                           BNE    62$                   ;    CALCULATE MEGA BYTES
6891 056144  022761  000017  000100           CMP    #17,D.XFMA(R1)        ; IF BYTE COUNT < 1 MB,
6892 056152  101031                           BHI    63$                   ;    CALCULATE KILOBYTES
6893 056154  103404                           BLO    62$                   ;
6894 056156  022761  041100  000076           CMP    #41100,D.XFLA(R1)     ;
6895 056164  101024                           BHI    63$                   ;
6896 056166                           62$:    PUSH   <R1>                  ; SAVE REGISTERS
     056166  010146                           MOV    R1,-(SP)              ; PUSH R1 ON STACK
6897 056170  016103  000102                   MOV    D.XFHA(R1),R3         ;
6898 056174  016102  000100                   MOV    D.XFMA(R1),R2         ; CALCULATE
6899 056200  016101  000076                   MOV    D.XFLA(R1),R1         ;   MEGA
6900 056204  012704  041100                   MOV    #41100,R4             ;     BYTE
6901 056210  012705  000017                   MOV    #17,R5                ;       COUNT
6902 056214  004737  047326                   CALL   DIV48                 ;
6903 056220  012737  060005  002306           MOV    #RPTMB,RPTAF          ; SET M BYTE FLAG
6904 056226  010137  002304                   MOV    R1,RPTAV              ; SAVE REMAINING
6905 056232                                   POP    <R1>                  ; RESTORE REGISTERS
     056232  012601                           MOV    (SP)+,R1              ; POP STACK INTO R1
6906 056234  000440                           BR     70$                   ;
6907 056236  005761  000100           63$:    TST    D.XFMA(R1)            ; IF NON ZERO,
6908 056242  001004                           BNE    64$                   ;    CALCULATE KILO BYTES
6909 056244  022761  001750  000076           CMP    #1000.,D.XFLA(R1)     ; IF BYTE COUNT < 1 KB,
6910 056252  101023                           BHI    65$                   ;    CALCULATE BYTES
6911 056254                           64$:    PUSH   <R1>                  ; SAVE REGISTERS
     056254  010146                           MOV    R1,-(SP)              ; PUSH R1 ON STACK
6912 056256  016103  000102                   MOV    D.XFHA(R1),R3         ;
```

```
6913 056262  016102  000100              MOV    D.XFMA(R1),R2      ; CALCULATE
6914 056266  016101  000076              MOV    D.XFLA(R1),R1      ;   KILO
6915 056272  012704  001750              MOV    #1000.,R4          ;     BYTE
6916 056276  005005                      CLR    R5                 ;       COUNT
6917 056300  004737  047326              CALL   DIV48              ;
6918 056304  012737  060003  002306      MOV    #RPTKB,RPTAF       ; SET M BYTE FLAG
6919 056312  010137  002304              MOV    R1,RPTAV           ; SAVE REMAINING
6920 056316                              POP    <R1>               ; RESTORE REGISTERS
     056316  012601                      MOV    (SP)+,R1           ; POP STACK INTO R1
6921 056320  000406                      BR     70$                ;
6922 056322  012737  060001  002306  65$: MOV   #RPTB,RPTAF        ;
6923 056330  016137  000076  002304      MOV    D.XFLA(R1),RPTAV   ;
6924 056336  005761  000110          70$: TST   D.XFHC(R1)         ; IF NON ZERO,
6925 056342  001011                      BNE    72$                ;   CALCULATE MEGA BYTES
6926 056344  022761  000017  000106      CMP    #17,D.XFMC(R1)     ; IF BYTE COUNT < 1 MB,
6927 056352  101031                      BHI    73$                ;   CALCULATE KILOBYTES
6928 056354  103404                      BLO    72$                ;
6929 056356  022761  041100  000104      CMP    #41100,D.XFLC(R1)  ;
6930 056364  101024                      BHI    73$                ;
6931 056366                          72$: PUSH  <R1>               ; SAVE REGISTERS
     056366  010146                      MOV    R1,-(SP)           ; PUSH R1 ON STACK
6932 056370  016103  000110              MOV    D.XFHC(R1),R3      ;
6933 056374  016102  000106              MOV    D.XFMC(R1),R2      ; CALCULATE
6934 056400  016101  000104              MOV    D.XFLC(R1),R1      ;   MEGA
6935 056404  012704  041100              MOV    #41100,R4          ;     BYTE
6936 056410  012705  000017              MOV    #17,R5             ;       COUNT
6937 056414  004737  047326              CALL   DIV48              ;
6938 056420  012737  060005  002312      MOV    #RPTMB,RPTCF       ; SET M BYTE FLAG
6939 056426  010137  002310              MOV    R1,RPTCV           ; SAVE REMAINING
6940 056432                              POP    <R1>               ; RESTORE REGISTERS
     056432  012601                      MOV    (SP)+,R1           ; POP STACK INTO R1
6941 056434  000440                      BR     80$                ;
6942 056436  005761  000106          73$: TST   D.XFMC(R1)         ; IF NON ZERO,
6943 056442  001004                      BNE    74$                ;   CALCULATE KILO BYTES
6944 056444  022761  001750  000104      CMP    #1000.,D.XFLC(R1)  ; IF BYTE COUNT < 1 KB,
6945 056452  101023                      BHI    75$                ;   CALCULATE BYTES
6946 056454                          74$: PUSH  <R1>               ; SAVE REGISTERS
     056454  010146                      MOV    R1,-(SP)           ; PUSH R1 ON STACK
6947 056456  016103  000110              MOV    D.XFHC(R1),R3      ;
6948 056462  016102  000106              MOV    D.XFMC(R1),R2      ; CALCULATE
6949 056466  016101  000104              MOV    D.XFLC(R1),R1      ;   KILO
6950 056472  012704  001750              MOV    #1000.,R4          ;     BYTE
6951 056476  005005                      CLR    R5                 ;       COUNT
6952 056500  004737  047326              CALL   DIV48              ;
6953 056504  012737  060003  002312      MOV    #RPTKB,RPTCF       ; SET M BYTE FLAG
6954 056512  010137  002310              MOV    R1,RPTCV           ; SAVE REMAINING
6955 056516                              POP    <R1>               ; RESTORE REGISTERS
     056516  012601                      MOV    (SP)+,R1           ; POP STACK INTO R1
6956 056520  000406                      BR     80$                ;
6957 056522  012737  060001  002312  75$: MOV   #RPTB,RPTCF        ;
6958 056530  016137  000104  002310      MOV    D.XFLC(R1),RPTCV   ;
6959
6960 056536                          80$: PUSH  <R1>               ;
     056536  010146                      MOV    R1,-(SP)           ; PUSH R1 ON STACK
6961 056540  016102  000124              MOV    D.UID+2(R1),R2     ; GET
```

```
6962 056544 016103 000126            MOV     D.UID+4(R1),R3      ;    UNIT
6963 056550 016101 000122            MOV     D.UID(R1),R1        ;      ID
6964 056554 012704 000017            MOV     #15.,R4             ; GET CHARACTER COUNT FOR STRING
6965 056560 004737 047472            CALL    ITOA                ; CONVERT BINARY TO ASCII
6966 056564                          POP     <R1>                ; RESTORE SAVED REGISTERS
     056564 012601                    MOV     (SP)+,R1            ; POP STACK INTO R1
6967 056566 116103 000130            MOVB    D.UID+6(R1),R3      ; GET MODEL TYPE BYTE
6968 056572 116104 000131            MOVB    D.UID+7.(R1),R4     ; GET CLASS TYPE BYTE
6969
6970 056576                          PRINTS  #RPTIOD,D.DRV(R1),R3,R4,#TEMP,RPTWV,RPTWF
     056576 013746 002276            MOV     RPTWF,-(SP)
     056602 013746 002274            MOV     RPTWV,-(SP)
     056606 012746 002212            MOV     #TEMP,-(SP)
     056612 010446                    MOV     R4,-(SP)
     056614 010346                    MOV     R3,-(SP)
     056616 016146 000000            MOV     D.DRV(R1),-(SP)
     056622 012746 057674            MOV     #RPTIOD,-(SP)
     056626 012746 000007            MOV     #7.,-(SP)
     056632 010600                    MOV     SP,R0
     056634 104416                    TRAP    C$PNTS
     056636 062706 000020            ADD     #20,SP
6971 056642                          PRINTS  #RPTIOE,RPTRV,RPTRF,RPTAV,RPTAF,RPTCV,RPTCF,D.ECCC(R1)
     056642 016146 000120            MOV     D.ECCC(R1),-(SP)
     056646 013746 002312            MOV     RPTCF,-(SP)
     056652 013746 002310            MOV     RPTCV,-(SP)
     056656 013746 002306            MOV     RPTAF,-(SP)
     056662 013746 002304            MOV     RPTAV,-(SP)
     056666 013746 002302            MOV     RPTRF,-(SP)
     056672 013746 002300            MOV     RPTRV,-(SP)
     056676 012746 057743            MOV     #RPTIOE,-(SP)
     056702 012746 000010            MOV     #10,-(SP)
     056706 010600                    MOV     SP,R0
     056710 104416                    TRAP    C$PNTS
     056712 062706 000022            ADD     #22,SP
6972 056716                          POP     <R5,R4,R3>          ; RESTORE SAVED REGISTERS
6973 056724 005303                   DEC     R3                  ; COUNT THE DRIVE TABLES
6974 056726 003402                   BLE     90$                 ;
6975 056730 000137 055520            JMP     30$                 ; REPEAT FOR ALL DRIVE TABLES
6976 056734 062705 000170  90$:      ADD     #C.SIZE,R5          ; GO TO
6977 056740 005715                   TST     (R5)                ;   NEXT
6978 056742 001402                   BEQ     RPTDES              ;     CONTROLLER
6979 056744 000137 054760            JMP     20$                 ;       TABLE
6980
6981 056750           RPTDES: PRINTS  #RPTDE1                    ; PRINT
     056750 012746 060007            MOV     #RPTDE1,-(SP)
     056754 012746 000001            MOV     #1,-(SP)
     056760 010600                    MOV     SP,R0
     056762 104416                    TRAP    C$PNTS
     056764 062706 000004            ADD     #4,SP
6982 056770                          PRINTS  #RPTDE2             ;       DRIVE ERROR
     056770 012746 060042            MOV     #RPTDE2,-(SP)
     056774 012746 000001            MOV     #1,-(SP)
     057000 010600                    MOV     SP,R0
     057002 104416                    TRAP    C$PNTS
     057004 062706 000004            ADD     #4,SP
```

```
6983 057010                              PRINTS  #RPTDE3                  ;        SUMMARY HEADINGS
     057010  012746  C60145              MOV     #RPTDE3,-(SP)
     057014  012746  000001              MOV     #1,-(SP)
     057020  010600                      MOV     SP,R0
     057022  104416                      TRAP    C$PNTS
     057024  062706  000004              ADD     #4,SP
6984 057030  013705  002200              MOV     CTABS,R5                 ; GET ADDRESS OF 1ST CONTROLLER TABLE
6985
6986 057034  010504           1$:       MOV     R5,R4                    ; GET ADDRESS OF CONTROLLER TABLE
6987
6988 057036  062704  000016              ADD     #C.DR0,R4                ; POINT TO DRIVE TABLE POINTERS
6989 057042  012703  000004              MOV     #4,,R3                   ; GET COUNT OF DRIVES
6990 057046  012401           2$:       MOV     (R4)+,R1                 ; LOOK AT POINTER
6991 057050  001477                      BEQ     7$                       ; GO TO NEXT IF NO TABLE
6992 057052                              PUSH    <R4>                     ; SAVE REGISTERS
6993 057054  010104                      MOV     R1,R4                    ; GET POINTER TO VOLUME
6994 057056  062704  000132              ADD     #D.VSN,R4                ;   SERIAL NUMBER
6995 057062  004737  047552              CALL    BLDSTR                   ; CONVERT INTEGER TO DECIMAL ASCII
6996 057066                              POP     <R4>                     ; RESTORE SAVED REGISTERS
6997 057070  016137  000002  002274      MOV     D.UNIT(R1),RPTLUN        ; GET LOGICAL
6998 057076  042737  177700  002274      BIC     #^C<DT.UNT>,RPTLUN       ;   UNIT NUMBER OF DEVICE
6999 057104  005761  000002              TST     D.UNIT(R1)               ; SEE IF DRIVE AVAILABLE
7000 057110  100027                      BPL     5$                       ; IF SO, SKIP (BIT 15 = 0)
7001 057112                              ASSUME DT.AVL EQ BIT15
7002 057112                              PRINTS  #RPTDED,RPTLUN,D.DRV(R1),#TEMP,#RPTSTA,D.HERR(R1),D.CERR(R1),D.SERR(R1)
     057112  016146  000114              MOV     D.SERR(R1),-(SP)
     057116  016146  000116              MOV     D.CERR(R1),-(SP)
     057122  016146  000112              MOV     D.HERR(R1),-(SP)
     057126  012746  060327              MOV     #RPTSTA,-(SP)
     057132  012746  002212              MOV     #TEMP,-(SP)
     057136  016146  000000              MOV     D.DRV(R1),-(SP)
     057142  013746  002274              MOV     RPTLUN,-(SP)
     057146  012746  060253              MOV     #RPTDED,-(SP)
     057152  012746  000010              MOV     #10,-(SP)
     057156  010600                      MOV     SP,R0
     057160  104416                      TRAP    C$PNTS
     057162  062706  000022              ADD     #22,SP
7003 057166  000426                      BR      6$
7004 057170                   5$:       PRINTS  #RPTDED,RPTLUN,D.DRV(R1),#TEMP,#RPTSTB,D.HERR(R1),D.CERR(R1),D.SERR(R1)
     057170  016146  000114              MOV     D.SERR(R1),-(SP)
     057174  016146  000116              MOV     D.CERR(R1),-(SP)
     057200  016146  000112              MOV     D.HERR(R1),-(SP)
     057204  012746  060341              MOV     #RPTSTB,-(SP)
     057210  012746  002212              MOV     #TEMP,-(SP)
     057214  016146  000000              MOV     D.DRV(R1),-(SP)
     057220  013746  002274              MOV     RPTLUN,-(SP)
     057224  012746  060253              MOV     #RPTDED,-(SP)
     057230  012746  000010              MOV     #10,-(SP)
     057234  010600                      MOV     SP,R0
     057236  104416                      TRAP    C$PNTS
     057240  062706  000022              ADD     #22,SP
7005
7006 057244  005303           6$:       DEC     R3                       ; COUNT THE DRIVE TABLES
7007 057246  003277                      BGT     2$                       ; REPEAT FOR ALL DRIVE TABLES
7008 057250  062705  000170           7$:  ADD   #C.SIZE,R5               ; GO TO
```

```
 7009 057254  005715                       TST     (R5)                    ;            NEXT
 7010 057256  001266                       BNE     1$                      ;            CONTROLLER TABLE
 7011 057260                               POP     <R5,R4,R3,R2,R1>        ;  RESTORE SAVED REGISTERS
 7012 057272  000240             NOP                                       ; [DBG] INSERT HALT FOR DEBUG
 7013
 7014 057274                               EXIT    RPT
      057274  000167                       .WORD   J$JMP
      057276  001054                       .WORD   L10046-2-.
 7015
 7016 057300     045    116    045 RPTHDR: .ASCIZ  \%N%ATEST %D1%A IN PROGRESS\
 7017 057333     045    116    045 RPTIO1: .ASCIZ  \%N%ASUBSYSTEM I/O SUMMARY:%N\
 7018 057370     045    101    103 RPTIO2: .ASCIZ  \%ACTRLR  DRIVE        UNIQUE          BYTES   BYTES   BYTES   BYTES   ECC%N
 7019 057505     045    101    040 RPTIO3: .ASCIZ  \%A CSR    NAME     IDENTIFIER      WRITTEN READ   ACCESSED COMPARED DATA%
                                                                                                                          N
 7020 057623     045    116    045 RPTIOC: .ASCIZ  \%N%06%S6%Z2%S1%Z2%S1%T%S2%D5%T%S2%D5%T%N\
 7021 057674     045    123    066 RPTIOD: .ASCIZ  \%S6%ADU%Z3%S1%Z2%S1%Z2%S1%T%S2%D5%T%S2\
 7022 057743     045    104    065 RPTIOE: .ASCIZ  \%D5%T%S2%D5%T%S2%D5%T%S2%D5%N\
 7023
 7024 060001     040    000        RPTB:   .ASCIZ  \ \
 7025 060003     113    000        RPTKB:  .ASCIZ  \K\
 7026 060005     115    000        RPTMB:  .ASCIZ  \M\
 7027
 7028 060007     045    116    045 RPTDE1: .ASCIZ  \%N%ADRIVE ERROR SUMMARY:%N\
 7029 060042     045    101    125 RPTDE2: .ASCIZ  \%AUNIT DRIVE      VOLUME          HARD/FATAL DATA CHECK    SOFT%N\
 7030 060145     045    101    040 RPTDE3: .ASCIZ  \%A #   NAME     SERIAL #        ERRORS      ERRORS      ERRORS%N%N\
 7031 060253     045    104    063 RPTDED: .ASCIZ  \%D3%S2%ADU%Z3%S2%T%S2%T%S4%D5%S6%D5%S6%D5%N\
 7032 060327     050    104    122 RPTSTA: .ASCIZ  \(DROPPED)\
 7033 060341     040    040    040 RPTSTB: .ASCIZ  \              \
 7034                                       .EVEN
 7035
 7036 060354                                ENDRPT
      060354                       L10046:
      060354  104425                        TRAP    C$RPT
```

```
7038                              .SBTTL   PROTECTION TABLE
7039
7040                          ;++
7041                          ;        THIS TABLE IS USED BY THE RUN TIME SERVICES
7042                          ;        TO PROTECT THE LOAD MEDIA.
7043                          ;--
7044
7045 060356                      BGNPROT
     060356                  L$PROT::
7046
7047 060356  000000              .WORD   0            ; P-TABLE OFFSET FOR CSR ADDRESS
7048 060360  177777              .WORD   -1           ; P-TABLE OFFSET FOR MASSBUS ADDRESS
7049 060362  000002              .WCRD   2            ; P-TABLE OFFSET FOR DRIVE NUMBER
7050
7051 060364                      ENDPROT
7052
```

```
7054                                    .SBTTL   INITIALIZE SECTION
7055
7056                                    ;++
7057                                    ;        THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
7058                                    ;        AT THE BEGINNING OF EACH PASS. THIS CODE IS EXECUTED UNDER FIVE
7059                                    ;        CONDITIONS.  THERE
7060                                    ;        ARE SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE
7061                                    ;        DIAGNOSTIC KNOW UNDER WHICH CONDITION THE EXECUTION IS TAKING
7062                                    ;        PLACE.   THE EVENT FLAGS ARE READ USING THE "READEF" MACRO.
7063                                    ;        THE CONDITIONS UNDER WHICH THE INIT CODE IS EXECUTED AND THE
7064                                    ;        CORRRESPONDING EVENT FLAGS ARE:
7065                                    ;                    START COMMAND              EF.START
7066                                    ;                    RESTART COMMAND            EF.RESTART
7067                                    ;                    CONTINUE COMMAND           EF.CONTINUE
7068                                    ;                    POWERDOWN/POWERUP          EF.PWR
7069                                    ;                    NEW PASS                   EF.NEW
7070                                    ;
7071                                    ;        IF HERE FROM START COMMAND OR POWER FAILURE THEN
7072                                    ;                    SET ISTRT BIT & CLEAR OTHER BITS IN FLAG
7073                                    ;
7074                                    ;        IF HERE FROM RESTART COMMAND THEN
7075                                    ;                    SET IREST BIT & CLEAR OTHER BITS IN IFLAGS
7076                                    ;
7077                                    ;        IF HERE FROM START OR RESTART COMMAND THEN
7078                                    ;                    RESET ALL UNITS
7079                                    ;                    ESTABLISH FREE MEMORY
7080                                    ;                    CLEAR TNUM
7081                                    ;                    INITIALIZE CLOCK
7082                                    ;                    BUILD CONTROLLER & DRIVES TABLES IN MEMORY
7083                                    ;                    EXIT INIT SECTION
7084                                    ;
7085                                    ;        IF HERE FROM CONTINUE COMMAND THEN
7086                                    ;                    SET ICONT BIT IN IFLAGS
7087                                    ;                    RE-INIT DATA STRUCTURES
7088                                    ;                    EXIT INIT SECTION
7089                                    ;
7090                                    ;        IF HERE FROM NEW PASS OR SUB-PASS THEN
7091                                    ;                    LOOK FOR ANY ADDED OR DROPPED UNITS
7092                                    ;                    EXIT INIT SECTION
7093                                    ;--
7094
7095 060364                                     BGNINIT
     060364                            L$INIT::
7096
7097 060364  005037  002206                     CLR      TNUM                    ; INITIALIZE TEST NUMBER
7098
7099 060370                                     READEF   #EF.STA                 ; HERE FROM START COMMAND?
     060370  012700  000040                     MOV      #EF.STA,R0
     060374  104447                             TRAP     C$REFG
7100 060376                                     BNCOMPLETE        1$              ; BRANCH TO 1$ IF NOT, ELSE
     060376  103004                             BCC      1$
7101 060400  012737  000010  002204             MOV      #ISTRT,IFLAGS           ; SET START BIT IN FLAG.
7102 060406  000521                             BR       INIT1
7103
7104 060410                            1$:      READEF   #EF.RES                 ; HERE FROM RESTART COMMAND?
```

```
        060410   012700  000037              MOV     #EF.RES,R0
        060414   104447                      TRAP    C$REFG
7105    060416                               BNCOMPLETE        2$          ; BRANCH TO 2$ IF NOT, ELSE
        060416   103004                      BCC     2$
7106    060420   012737  000004  002204      MOV     #IREST,IFLAGS         ; SET RESTART BIT IN FLAG.
7107    060426   000511                      BR      INIT1
7108
7109    060430                       2$:     READEF  #EF.CON              ; HERE FROM CONTINUE COMMAND?
        060430   012700  000036              MOV     #EF.CON,R0
        060434   104447                      TRAP    C$REFG
7110    060436                               BNCOMPLETE        4$          ; IF NOT, DO END OF PASS CODE
        060436   103003                      BCC     4$
7111    060440   052737  000002  002204      BIS     #ICONT,IFLAGS         ; SET CONTINUE BIT IN FLAG.
7112                                 ;
7113                                 ;        PERFORM END OF PASS CODE.
7114                                 ;        RE-INITIALIZE ALL DATA STRUCTURES
7115                                 ;        MAKE ALL CONTROLLER/DRIVE TABLES NOT AVAILABLE FOR TESTING
7116                                 ;
7117    060446   013705  002200      4$:     MOV     CTABS,R5             ; GET ADDRESS OF 1ST CONTROLLER TABLE
7118    060452   052765  100000  000014 5$:  BIS     #CT.AVL,C.FLG(R5)    ; SET CONTROLLER TABLE NOT AVAILABLE
7119    060460   005065  000160              CLR     C.CMFL(R5)           ; RESET COMMAND
7120    060464   005065  000162              CLR     C.CMBL(R5)           ;   LIST POINTERS
7121    060470   005065  000164              CLR     C.TBSF(R5)           ; RESET TBS LIST
7122    060474   005065  000166              CLR     C.TBSB(R5)           ;   POINTERS
7123    060500   010502                      MOV     R5,R2                ; GET POINTER TO DRIVE TABLES
7124    060502   062702  000016              ADD     #C.DR0,R2            ;
7125    060506   012703  000004              MOV     #4.,R3               ; GET NUMBER OF DRIVES PER CONTROLLER
7126    060512   012200              6$:     MOV     (R2)+,R0             ; SEE IF THIS DRIVE HAS A TABLE,
7127    060514   001403                      BEQ     7$                   ; BRANCH IF NOT, ELSE
7128    060516   052760  100000  000002      BIS     #DT.AVL,D.UNIT(R0)   ; SET DRIVE TABLE NOT AVAILABLE.
7129    060524   005303              7$:     DEC     R3                   ; GET NEXT DRIVE IN CONTROLLER TABLE,
7130    060526   001371                      BNE     6$                   ; BRANCH IF NO DRIVES, ELSE
7131    060530   062705  000170              ADD     #C.SIZE,R5           ; MOVE TO NEXT CONTROLLER TABLE
7132    060534   005715                      TST     (R5)                 ; IS THERE A NEXT ONE?
7133    060536   001345                      BNE     5$                   ; IF SO, CLEAR THE BITS THERE
7134                                 ;
7135                                 ;        NOW GET EACH P-TABLE AND MAKE THE APPROPRIATE CONTROLLER/DRIVE
7136                                 ;        TABLES AVAILABLE FOR TESTING.
7137                                 ;
7138    060540   005003                      CLR     R3                   ; START WITH LOGICAL UNIT 0
7139    060542                       8$:     GPHARD  R3,R0                ; GET POINTER TO IT'S P-TABLE
        060542   010300                      MOV     R3,R0
        060544   104442                      TRAP    C$GPHRD
7140    060546                               BNCOMPLETE        12$         ; BRANCH TO 12$ IF NOT AVAILABLE
        060546   103033                      BCC     12$
7141    060550   013705  002200              MOV     CTABS,R5             ; GET ADDRESS OF 1ST CONTROLLER TABLE
7142    060554   021015              9$:     CMP     (R0),(R5)            ; SEE IF CSR ADDRESSES ARE THE SAME,
7143    060556   001410                      BEQ     11$                  ; BRANCH IF SO, ELSE
7144    060560   062705  000170              ADD     #C.SIZE,R5           ; LOOK AT NEXT CONTROLLER TABLE.
7145    060564   005715                      TST     (R5)                 ; ANY MORE CONTROLLER TABLES?
7146    060566   001372                      BNE     9$                   ; BRANCH IF SO, ELSE
7147    060570   012701  060566      10$:    MOV     #.-2,R1              ; SAVE CURRENT LOCATION
7148    060574   000137  024310              JMP     BADMEM               ; PRINT SYSTEM FATAL ERROR
7149
7150    060600   016001  000002      11$:    MOV     H.DRV(R0),R1         ; GET DRIVE NUMBER FROM P-TABLE
```

```
7151 060604  004737  047164              CALL     GTDRVT                  ; FIND THE DRIVE TABLE ADDRESS
7152 060610  103767                       BCS      10$                     ; BRANCH IF NOT FOUND, ELSE
7153 060612  042765  100000  000014       BIC      #CT.AVL,C.FLG(R5)       ; CLEAR NOT AVAILABLE FLAG
7154 060620  042761  147000  000002       BIC      #DT.CLR,D.UNIT(R1)      ; CLEAR TEMPORARY DRIVE FLAGS
7155 060626  005061  000060               CLR      D.CCNT(R1)              ; ZERO "COMPARE HOST DATA" COUNT
7156 060632  005061  000056               CLR      D.WCNT(R1)              ; ZERO OUTSTANDING WRITE COUNT
7157 060636  005203                 12$:  INC      R3                      ; INCREMENT TO NEXT UNIT IN P-TABLE
7158 060640  020337  002012               CMP      R3,L$UNIT               ; SEE IF ALL P-TABLES CHECKED,
7159 060644  002736                        BLT      8$                      ; BRANCH IF NOT, ELSE
7160 060646  000137  062062          13$:  JMP      INIT7                   ; RE-INIT DATA STRUCTURES
7161                                  ;
7162                                  ;       INITIALIZE KW11 CLOCK, FREE MEMORY AND IP ADDRESS TABLE
7163                                  ;       DURING START OR RESTART COMMAND ONLY
7164
7165 060652  005037  002332        INIT1:  CLR      KW.EL                   ; CLEAR ELAPSED TIME
7166 060656  005037  002334               CLR      KW.EL+2                 ;
7167 060662                               CLOCK    L,R0                    ; SEE IF L-CLOCK PRESENT
     060662  012700  000114               MOV      #'L,R0
     060666  104462                        TRAP     C$CLCK
7168 060670                               BCOMPLETE          2$            ;
     060670  103417                        BCS      2$
7169 060672                               CLOCK    P,R0                    ; SEE IF P-CLOCK PRESENT
     060672  012700  000120               MOV      #'P,R0
     060676  104462                        TRAP     C$CLCK
7170 060700                               BCOMPLETE          2$            ;
     060700  103413                        BCS      2$
7171 060702  005037  002322               CLR      KW.CSR                  ; IF NEITHER, CLEAR CSR STORAGE WORD
7172 060706                               PRINTF   #NOCLOCK                ; PRINT "NO CLOCK" MESSAGE
     060706  012746  011071               MOV      #NOCLOCK,-(SP)
     060712  012746  000001               MOV      #1,-(SP)
     060716  010600                        MOV      SP,R0
     060720  104417                        TRAP     C$PNTF
     060722  062706  000004               ADD      #4,SP
7173 060726  000435                        BR       3$                      ;
7174
7175 060730  012037  002322          2$:  MOV      (R0)+,KW.CSR            ; STORE DATA RETURNED
7176 060734  012037  002324               MOV      (R0)+,KW.BRL            ;
7177 060740  012037  002326               MOV      (R0)+,KW.VEC            ;
7178 060744  011037  002330               MOV      (R0),KW.HZ              ;
7179 060750  011001                        MOV      (R0),R1                 ; CALCULATE
7180 060752  012702  000074               MOV      #60.,R2                 ;    TICKS PER
7181 060756  004737  047420               CALL     MULT                    ;      MINUTE
7182 060762  010337  002336               MOV      R3,KW.60                ;
7183 060766                               SETVEC   KW.VEC,#KW11I,#PRIO7    ; SETUP KW11 VECTOR ADDRESS
     060766  012746  000340               MOV      #PRIO7,-(SP)
     060772  012746  054512               MOV      #KW11I,-(SP)
     060776  013746  002326               MOV      KW.VEC,-(SP)
     061002  012746  000003               MOV      #3,-(SP)
     061006  104437                        TRAP     C$SVEC
     061010  062706  000010               ADD      #10,SP
7184 061014  012777  000105  121300       MOV      #KW.OUT,@KW.CSR         ; START THE CLOCK
7185 061022                          3$:  MEMORY   FFREE                   ; RESET START OF FREE MEMORY
     061022  104431                        TRAP     C$MEM
     061024  010037  002166               MOV      R0,FFREE
7186 061030  017737  121132  002170       MOV      @FFREE,FSIZE            ; RESET SIZE OF FREE MEMORY
```

```
7187 061036  005037  002260            CLR     TS.PAT              ; RESET TEST PATTERN SELECTION
7188 061042  012701  006250            MOV     #PAT00+2,R1         ; RE-INIT
7189 061046  012721  000001            MOV     #1,(R1)+            ;     DEFAULT
7190 061052  012700  000020            MOV     #16.,R0             ;        PATTERN
7191 061056  005021             4$:    CLR     (R1)+               ;          16
7192 061060  005300                    DEC     R0                  ;
7193 061062  001375                    BNE     4$                  ;
7194                            ;
7195                            ;       ALLOCATE DRIVE TABLES TO MEMORY
7196
7197 061064  013737  002166  002176 INIT2:  MOV  FFREE,DTABS       ; STORE START OF DRIVE TABLES AND
7198 061072  005077  121100            CLR     @DTABS              ; MARK ZERO END.
7199 061076  013700  002012            MOV     L$UNIT,R0           ; GET NUMBER OF LOGICAL UNITS TO RUN,
7200 061102  012701  000001            MOV     #1,R1               ; GET INITIAL SIZE OF DRIVE TABLE AND
7201 061106  062701  000057     1$:    ADD     #<D.SIZE>/2,R1      ; ACCUMULATE DRIVE TABLE SIZE.
7202 061112  005300                    DEC     R0                  ; SEE IF ANY MORE LOGICAL UNITS,
7203 061114  001374                    BNE     1$                  ; BRANCH IF NOT, ELSE
7204 061116  004737  024322            CALL    ALOCM               ; ALLOCATE ALL DRIVE TABLES TO MEMORY.
7205                            ;                                  ; R1 POINTS TO 1ST WORD IN DRIVE TABLE
7206                            ;
7207                            ;       INITIALIZE CONTROLLER TABLE STORAGE WITH A WORD OF ZEROS
7208
7209 061122  013737  002166  002200 INIT3:  MOV  FFREE,CTABS       ; STORE START OF CONTROLLER TABLES AND
7210 061130  005077  121044            CLR     @CTABS              ; MARK ZEROS END.
7211 061134  005037  002202            CLR     CTRLRS              ; CLEAR CONTROLLER COUNT
7212                            ;
7213                            ;       BUILD CONTROLLER TABLES
7214
7215 061140  005002             INIT4:  CLR    R2                  ; START WITH LOGICAL UNIT 0
7216 061142  005003                    CLR     R3                  ; CLEAR FLAG FOR WARNING MESSAGE
7217 061144  012737  000160  002262    MOV     #160,VECSAV         ; SAVE DEFAULT FOR VECTOR
7218        061152             1$:    GPHARD   R2,R0               ; GET POINTER TO IT'S P-TABLE
     061152  010200                    MOV     R2,R0
     061154  104442                    TRAP    C$GPHRD
7219 061156                            BNCOMPLETE       26$        ; BRANCH TO 26$ IF NOT AVAILABLE
     061156  103142                    BCC     26$
7220 061160  013705  002200            MOV     CTABS,R5            ; GET ADDRESS OF 1ST CONTROLLER TABLE
7221 061164  005715             2$:    TST     (R5)                ; CHECK IF ANY MORE TABLES
7222 061166  001405                    BEQ     10$                 ; BUILD NEW TABLE IF FOUND ZERO WORD
7223 061170  021015                    CMP     (R0),(R5)           ; CHECK IF SAME CSR ADDRESS.
7224 061172                            ASSUME  C.IPR EQ 0
7225 061172                            ASSUME  H.IPR EQ 0
7226 061172  001446                    BEQ     21$                 ; BRANCH IF SO
7227
7228 061174  062705  000170     5$:    ADD     #C.SIZE,R5          ; POINT TO BEGINNING OF NEXT CONTROLLER
7229 061200  000771                    BR      2$                  ; TABLE IN MEMORY.
7230                            ;
7231                            ;       BUILD NEW CONTROLLER TABLE
7232
7233 061202  012701  000074    10$:    MOV     #<C.SIZE>/2,R1      ; GET # OF ENTRIES IN CONTROLLER TABLE
7234 061206  004737  024322            CALL    ALOCM               ; AND ALLOCATE A TABLE TO MEMORY.
7235                            ;                                  ; R0 => 1ST WORD P-TABLE
7236                            ;                                  ; R1 => 1ST WORD IN CONTROLLER TABLE
7237 061212                            PUSH    R1                  ; SAVE ADDRESS OF TABLE
     061212  010146                    MOV     R1,-(SP)            ; PUSH R1 ON STACK
```

```
7238 061214 011021              MOV     (R0),(R1)+              ; STORE CSR ADDRESS AND
7239 061216 010221              MOV     R2,(R1)+                ; UNIT NUMBER IN THE CONTROLLER TABLE.
7240 061220 013704 002262       MOV     VECSAV,R4               ; GET DEFAULT VECTOR & BR LEVEL
7241 061224 162704 000004       SUB     #4,R4                   ; GET NEXT VECTOR
7242 061230 010437 002262       MOV     R4,VECSAV               ; SAVE NEXT VECTOR
7243 061234 010421              MOV     R4,(R1)+                ; STORE IT IN THE CONTROLLER TABLE.
7244 061236 012721 000074       MOV     #60.,(R1)+              ; SET DEFAULT CONTROLLER TIMEOUT
7245 061242 012721 004037       MOV     #4037,(R1)+             ; THE 'JSR R0' INSTRUCION AND
7246 061246 012721 054502       MOV     #CNTSRV,(R1)+           ; THE ADDRESS OF THE INTERRUPT SERVICE
7247                                                            ; ROUTINE IN THE CONTROLLER TABLE.
7248 061252 032737 000400 002154  BIT   #SM.DET.SO.BIT+SFPTBL  ; DO
7249 061260 001403              BEQ     12$                     ;    DETERMINISTIC PHASE?
7250 061262 012721 002000       MOV     #CT.DET,(R1)+           ; SET DETERMINISTIC PHASE FLAG
7251 061266 000401              BR      13$                     ;
7252 061270 005021          12$: CLR    (R1)+                   ; CLEAR FLAG WORD
7253 061272 012704 000065   13$: MOV    #<C.SIZE-C.DR0>/2,R4    ; GET # OF ENTRIES TO END OF TABLE,
7254 061276 005021          14$: CLR    (R1)+                   ; CLEAR REST OF TABLE AND
7255 061300 005304              DEC     R4                      ; ADD ZERO WORD AT END.
7256 061302 002375              BGE     14$                     ; LOOP TIL ALL CLEARED
7257 061304 005237 002202       INC     CTRLRS                  ; KEEP TRACK OF CONTROLLER COUNT
7258                         ;
7259                         ;    BUILD DRIVE TABLES
7260                         ;
7261 061310              21$: PUSH      <R5>                     ; SAVE CONTROLLER TABLE POINTER
     061310 010546            MOV       R5,-(SP)                ; PUSH R5 ON STACK
7262 061312 013701 002176     MOV       DTABS,R1                ; GET ADDRESS OF CURRENT DRIVE TABLE
7263 061316 062705 000016     ADD       #C.DR0,R5               ; INDEX TO 1ST DRIVE IN TABLE
7264 061322 012704 000004     MOV       #4,R4                   ; GET # OF DRIVES PER CONTROLLER
7265 061326 005715      22$: TST        (R5)                    ; ANY ENTRY TO DRIVE TABLE.
7266 061330 001422            BEQ       24$                     ; BRANCH IF NOT, ELSE
7267 061332 026035 000002     CMP       H.DRV(R0),@(R5)+        ; COMPARE DRIVE NUMBER IN DRIVE TABLE,
7268 061336 001007            BNE       23$                     ; IF DIFFERENT, BUILD NEW TABLE
7269 061340 014501            MOV       -(R5),R1                ; ELSE, GET ADDRESS OF DRIVE TABLE
7270 061342                   POP       <R5>                    ;    GET CONTROLLER ADDRESS
     061342 012605            MOV       (SP)+,R5                ; POP STACK INTO R5
7271 061344                   ERRSF 2., ERR002                  ;    PRINT 'MULTIPLE P-TABLE' ERROR
     061344 104454            TRAP      C$ERSF
     061346 000002            .WORD     2
     061350 000000            .WORD     0
     061352 017056            .WORD     ERR002
7272 061354                   DOCLN                             ;    DO CLEAN-UP TRAP
     061354 104444            TRAP      C$DCLN
7273                          ;
7274 061356 005304      23$: DEC        R4                      ; COUNT DRIVES
7275 061360 001362            BNE       22$                     ; IF LESS THAN 4 DRIVES, BUILD NEW TABLE
7276 061362                   POP       <R5>                    ; ELSE, GET CONTROLLER ADDRESS
     061362 012605            MOV       (SP)+,R5                ; POP STACK INTO R5
7277 061364                   ERRSF 3., ERR003                  ;    PRINT 'TOO MANY DRIVES' ERROR
     061364 104454            TRAP      C$ERSF
     061366 000003            .WORD     3
     061370 000000            .WORD     0
     061372 017124            .WORD     ERR003
7278 061374                   DOCLN                             ;    DO CLEAN-UP TRAP
     061374 104444            TRAP      C$DCLN
7279
```

```
7280 061376 010115              24$:    MOV     R1,(R5)                 ; STORE ADDRESS OF DRIVE TABLE IN
7281                                                                     ; CONTROLLER TABLE.
7282 061400 016021 000002               MOV     H.DRV(R0),(R1)+         ; STORE DRIVE NUMBER AND
7283 061404 010211                       MOV     R2,(R1)                ; LOGICAL UNIT NUMBER IN DRIVE TABLE.
7284 061406 056011 000004               BIS     H.PRM(R0),(R1)         ; SET HARDWARE PARAMETER BITS
7285 061412 052711 020000               BIS     #DT.PRM,(R1)           ; SET DEFAULT TEST PARAMETERS
7286 061416 052103                       BIS     (R1)+,R3               ; SAVE FOR WARNING MESSAGE
7287
7288 061420 012721 000001               MOV     #1,(R1)+               ; LOAD 1 INTO B/E SET COUNT
7289 061424 005021                       CLR     (R1)+                  ; INIT
7290 061426 005021                       CLR     (R1)+                  ;    BEGIN/END
7291 061430 012721 177777               MOV     #-1.,(R1)+             ;       SET
7292 061434 012721 177777               MOV     #-1.,(R1)+             ;          LIMITS
7293
7294 061440 012704 000050               MOV     #<D.SIZE-D.BGN2>/2,R4  ; GET # OF ENTRIES TO END OF TABLE.
7295 061444 005021              25$:    CLR     (R1)+                  ; CLEAR REST OF TABLE
7296 061446 005304                       DEC     R4                     ;
7297 061450 002375                       BGE     25$                    ; LOOP TIL ALL CLEARED
7298
7299 061452 062737 000136 002176        ADD     #D.SIZE,DTABS          ; NEXT DRIVE TABLE ADDRESS AND
7300 061460 005077 120512               CLR     @DTABS                 ; MARK ZERO END.
                                26$:    POP     <R5>                   ; RESTORE STACK
     061464 012605                       MOV     (SP)+,R5               ; POP STACK INTO R5
7301 061464                             INC     R2                     ; INCREMENT LOGICAL UNIT NUMBER
7302 061466 005202                       CMP     R2,L$UNIT              ; CHECK IF GOT ALL TABLES
7303 061470 020237 002012               BLT     1$                     ; IF NOT, GO BACK FOR NEXT. ELSE
7304 061474 002626                       MOV     #1,R1                  ; GET 1 WORD TO TERMINATE ALL CONTROLLER
7305 061476 012701 000001               CALL    ALOCM                  ; TABLES AND ALLOCATE IT TO MEMORY.
7306 061502 004737 024322
7307                            ;
7308                            ;       CHECK FOR CUSTOMER WARNING MESSAGE
7309                            ;
7310 061506 032703 010000      INIT5:   BIT     #HM.WRT,R3             ; CHECK IF BIT EVER SET
7311 061512 001475                       BEQ     INIT6                  ; BYPASS IF NOT
7312 061514                              PRINTF  #INITWA                ; PRINT WARNING
     061514 012746 010725               MOV     #INITWA,-(SP)
     061520 012746 000001               MOV     #1,-(SP)
     061524 010600                       MOV     SP,R0
     061526 104417                       TRAP    C$PNTF
     061530 062706 000004               ADD     #4,SP
7313 061534                              PRINTF  #INITWB                ;    HEADER
     061534 012746 010777               MOV     #INITWB,-(SP)
     061540 012746 000001               MOV     #1,-(SP)
     061544 010600                       MOV     SP,R0
     061546 104417                       TRAP    C$PNTF
     061550 062706 000004               ADD     #4,SP
7314 061554 013705 002200               MOV     CTABS,R5               ; GET ADDRESS 1ST CONTROLLER TABLE
7315 061560 010504              1$:     MOV     R5,R4                  ; GET ADDRESS OF POINTER TO DRIVE TABLE
7316 061562 062704 000016               ADD     #C.DR0,R4
7317 061566 012701 000004               MOV     #4,R1                  ; GET COUNT OF DRIVE TABLES
7318 061572 012403              2$:     MOV     (R4)+,R3               ; GET ADDRESS OF DRIVE TABLE
7319 061574 001423                       BEQ     4$                     ;
7320 061576 032763 010000 000002        BIT     #DT.WRT,D.UNIT(R3)     ; CHECK IF CUSTOMER DATA SELECTED
7321 061604 001415                       BEQ     3$                     ;
7322 061606 116300 000002               MOVB    D.UNIT(R3),R0          ; GET LOGICAL UNIT NUMBER
7323 061612                              PRINTF  #INITWC,R0,(R5),(R3)   ; PRINT NUMBERS
```

```
           061612  011346                        MOV      (R3),-(SP)
           061614  011546                        MOV      (R5),-(SP)
           061616  010046                        MOV      R0,-(SP)
           061620  012746  011044                MOV      #INITWC,-(SP)
           061624  012746  000004                MOV      #4,-(SP)
           061630  010600                        MOV      SP,R0
           061632  104417                        TRAP     C$PNTF
           061634  062706  000012                ADD      #12,SP
7324       061640  005301          3$:           DEC      R1                      ; COUNT THE DRIVE TABLES
7325       061642  001353                        BNE      2$                      ; LOOK AT ALL OF THEM
7326       061644  062705  000170  4$:           ADD      #C.SIZE,R5              ; MOVE TO NEXT CONTROLLER TABLE
7327       061650  005715                        TST      (R5)                    ; SEE IF ANOTHER TABLE AND
7328       061652  001342                        BNE      1$                      ; LOOK AT IT
7329                                  ;
7330                                  ;          GET CONFIRMATION TO PROCEED
7331                                  ;
7332       061654                     MANUAL                                      ; CHECK IF MANUAL INTERVENTION ALLOWED
           061654  104450                        TRAP     C$MANI
7333       061656                     BNCOMPLETE INIT6                            ; BRANCH IF NOT ALLOWED
           061656  103013                        BCC      INIT6
7334       061660                     GMANIL     INITWD,TEMP,1,NO                 ; ASK OPERATOR
           061660  104443                        TRAP     C$GMAN
           061662  000404                        BR       10000$
           061664  002212                        .WORD    TEMP
           061666  000120                        .WORD    T$CODE
           061670  010224                        .WORD    INITWD
           061672  000001                        .WORD    1
           061674                     10000$:
7335       061674  032737  000001  002212        BIT      #1,TEMP                 ; LOOK AT RESPONSE
7336       061702  001001                        BNE      INIT6                   ; BRANCH IF YES WAS ANSWER
7337       061704                     DOCLN                                       ; DO CLEAN-UP TRAP
           061704  104444                        TRAP     C$DCLN
7338                                  ;
7339                                  ;          ALLOCATE RESPONSE ENVELOPES
7340                                  ;
7341       061706  013705  002200     INIT6:     MOV      CTABS,R5                ; GET POINTER TO FIRST TABLE
7342       061712  012701  000517     11$:       MOV      #RSPSIZ*5,R1            ; ALLOCATE
7343       061716  004737  024322                CALL     ALOCM                   ;   RESPONSE ENVELOPES
7344       061722  010165  000104                MOV      R1,C.RHDR(R5)           ; R.E. LIST HEADER POINTS TO TOP
7345       061726  012700  000004                MOV      #RNGLEN,R0              ; SET UP LINKS
7346       061732  010161  000000     18$:       MOV      R1,RE.RFL(R1)           ;   TO ALL
7347       061736  062761  000206  000000        ADD      #RSPSIZ*2,RE.RFL(R1)    ;       OF THE
7348       061744  016101  000000                MOV      RE.RFL(R1),R1           ;           R.E. BUFFERS
7349       061750  005300                        DEC      R0                      ; IF NOT ZERO,
7350       061752  001367                        BNE      18$                     ;    GET NEXT R.E. BUFFER
7351       061754  016561  000104  000000        MOV      C.RHDR(R5),RE.RFL(R1)   ; MAKE IT A RING
7352       061762  010565  000072                MOV      R5,C.CBAS(R5)           ; CALCULATE
7353       061766  062765  000132  000072        ADD      #C.CRNG,C.CBAS(R5)      ;    COMMAND RING
7354       061774  016565  000072  000074        MOV      C.CBAS(R5),C.CEND(R5)   ;        ADDRESSES
7355       062002  062765  000020  000074        ADD      #<RNGLEN*4>,C.CEND(R5)  ;
7356       062010  010565  000100                MOV      R5,C.RBAS(R5)           ; CALCULATE
7357       062014  062765  000112  000100        ADD      #C.RRNG,C.RBAS(R5)      ;    RESPONSE RING
7358       062022  016565  000100  000102        MOV      C.RBAS(R5),C.REND(R5)   ;        ADDRESSES
7359       062030  062765  000020  000102        ADD      #<RNGLEN*4>,C.REND(R5)  ;
7360       062036  062705  000170                ADD      #C.SIZE,R5              ; MOVE TO NEXT CONTROLLER TABLE
```

```
7361 062042  005715                          TST    (R5)                ; IF ANOTHER TABLE,
7362 062044  001322                          BNE    11$                 ;   SET UP POINTERS.
7363 062046  013737  002166  002172          MOV    FFREE,FMADR         ; SAVE FFREE &
7364 062054  013737  002170  002174          MOV    FSIZE,FMSIZ         ;   FSIZE FOR NEXT PASS
7365                                    ;
7366                                    ;           INITIALIZE FREE MEMORY BUFFERS
7367                                    ;
7368 062062  013737  002172  002166   INIT7:  MOV    FMADR,FFREE         ; RESTORE FFREE &
7369 062070  013737  002174  002170          MOV    FMSIZ,FSIZE         ;   FSIZE
7370
7371 062076  013701  002202                  MOV    CTRLRS,R1           ; NEED A MINIMUM
7372 062102  006301                          ASL    R1                  ;   OF 4 MESSAGE ENVELOPES
7373 062104  006301                          ASL    R1                  ;     PER CONTROLLER
7374 062106  010137  006234                  MOV    R1,TBSMIN           ; CALCULATE
7375 062112  012702  000051                  MOV    #CMDSIZ,R2          ;   MINIMUM MEMORY (WORDS)
7376 062116  004737  047420                  CALL   MULT                ;     NEEDED FOR
7377 062122  010337  006232                  MOV    R3,MEMIN            ;       MESSAGE ENVELOPES
7378 062126                                  MMU    ON                  ; TURN ON MEMORY MANAGEMENT UNIT
     062126  012700  000001                  MOV    #ON,R0
     062132  104503                          TRAP   C$MMU
7379 062134                                  BNCOMPLETE 3$              ; IF NOT SUCCESSFUL, USE FREE MEMORY
     062134  103023                          BCC    3$
7380 062136  042737  100000  002204          BIC    #IMMU,IFLAGS        ; FLAG MMU IS AVAILABLE
7381 062144  005037  004370                  CLR    BUFBA               ; GET BASE ADDRESS OF BUFFER
7382 062150  012737  000001  004372          MOV    #1,BUFEA            ; GET EXTENDED ADDRESS FOR BUFFER
7383 062156  012737  000010  004376          MOV    #8.,BUFSIZ          ; GET BUFFER SIZE IN 4KW
7384 062164                                  BUFREQ BUFSIZ,BUFBA,BUFEA  ; ALLOCATE BUFFER
     062164  000403                          BR     10001$
     062166  004376                          .WORD  BUFSIZ
     062170  004372                          .WORD  BUFEA
     062172  004370                          .WORD  BUFBA
                                     10001$:
     062174  012700  062166                  MOV    #.-6,R0
     062200  104501                          TRAP   C$FREQ
7385 062202                                  BCOMPLETE 10$             ; IF SUCCESSFUL, MAP THE BUFFER
     062202  103433                          BCS    10$
7386 062204  013737  006234  006236   3$:    MOV    TBSMIN,TBSSIZ       ; ALLOCATE MINIMUM
7387 062212  013701  006232                  MOV    MEMIN,R1            ;   SIZE
7388 062216  004737  024322                  CALL   ALOCM               ;     MESSAGE ENVELOPE
7389 062222  010137  006240                  MOV    R1,TBSTRT           ;       LINKED LIST
7390 062226  013701  002170                  MOV    FSIZE,R1            ; CONVERT
7391 062232  005002                          CLR    R2                  ;   BUFFER
7392 062234  005003                          CLR    R3                  ;     SIZE TO
7393 062236  012704  000400                  MOV    #256.,R4            ;       512 BYTE
7394 062242  004737  047256                  CALL   DIVIDE              ;         BLOCKS
7395 062246  010137  004374                  MOV    R1,BUFBLK           ; SAVE NUMBER OF 512 BYTE BLOCKS
7396 062252  001002                          BNE    4$                  ; IF ZERO,
7397 062254  000137  062574                  JMP    NOMEM               ;   FATAL SYSTEM ERROR
7398 062260  010103                  4$:     MOV    R1,R3               ; GET BLOCK COUNT
7399 062262  052737  100000  002204          BIS    #IMMU,IFLAGS        ; MMU IS NOT AVAILABLE
7400 062270  000440                          BR     20$                 ; BUILD LINKED LIST
7401
7402 062272  005737  004376           10$:   TST    BUFSIZ              ; IF ZERO,
7403 062276  001742                          BEQ    3$                  ;   USE FREE MEMORY
7404 062300  013701  002170                  MOV    FSIZE,R1            ; GET REMAINING FREE MEM
```

```
7405 062304 023701 006232              CMP    MEMIN,R1              ; IS MEMORY BIG ENOUGH FOR M.E.?
7406 062310 003131                     BGT    NOMEM                ; IF TOO SMALL, EXIT WITH ERROR
7407 062312 005002                     CLR    R2                   ; ALLOCATE
7408 062314 005003                     CLR    R3                   ;    TBS
7409 062316 012704 000051              MOV    #CMDSIZ,R4           ;       PACKET
7410 062322 004737 047256              CALL   DIVIDE               ;          QUEUE
7411 062326 022701 000377              CMP    #255.,R1             ; DON'T USE
7412 062332 002002                     BGE    11$                  ;    MORE THAN
7413 062334 012701 000377              MOV    #255.,R1             ;       255 PACKETS
7414 062340 013737 002166 006240 11$:  MOV    FREE,TBSTRT          ; GET TBS LIST START
7415 062346 010137 006236              MOV    R1,TBSSIZ            ;    AND LENGTH (PACKETS)
7416 062352 012701 000020              MOV    #16.,R1              ; CONVERT
7417 062356 013702 004376              MOV    BUFSIZ,R2           ;    BUFFER SIZE
7418 062362 004737 047420              CALL   MULT                 ;       FROM PAGES
7419 062366 010337 004374              MOV    R3,BUFBLK           ;          BLOCKS
7420
7421 062372 012701 004626         20$: MOV    #MMPKTS,R1           ; POINT TO MAPPED MEMORY PACKET AREA
7422 062376 010137 004622              MOV    R1,M.HDR            ; INITIALIZE AVAILABLE
7423 062402 005021                     CLR    (R1)+               ; TERMINATE AVAILABLE MEMORY LIST
7424 062404 013721 004374              MOV    BUFBLK,(R1)+        ; SET LENGTH TO ENTIRE AREA
7425 062410 005021                     CLR    (R1)+               ; MAPPED MEMORY OFFSET = 0
7426 062412 010137 004624              MOV    R1,F.HDR            ; INITIALIZE FREE MEMORY
7427 062416 005303              21$:    DEC    R3                   ; SET BLOCK COUNT TO REMAINING PACKETS
7428 062420 001407                     BEQ    30$                  ; EXIT WHEN DONE
7429 062422 010102                     MOV    R1,R2                ; PUT CURRENT ADDRESS IN PACKET
7430 062424 062702 000006              ADD    #PKTSIZ,R2          ; POINT TO NEXT PACKET
7431 062430 010221                     MOV    R2,(R1)+            ; STORE POINTER TO NEXT PACKET
7432 062432 005021                     CLR    (R1)+               ; CLEAR REMAINDER
7433 062434 005021                     CLR    (R1)+               ;    OF PACKET
7434 062436 000767                     BR     21$                  ; DO NEXT PACKET
7435
7436 062440 162701 000006         30$: SUB    #PKTSIZ,R1          ; MOVE BACK TO PREVIOUS PACKET
7437 062444 005011                     CLR    (R1)                 ; CLEAR POINTER TO END LIST
7438
7439 062446 013704 006240         50$: MOV    TBSTRT,R4            ; INIT TBS FREE PACKET LIST
7440 062452 013702 006236              MOV    TBSSIZ,R2           ; GET SIZE IN PACKETS
7441 062456 020237 002202              CMP    R2,CTRLRS           ; TBS LIST MUST CONTAIN MORE
7442 062462 003444                     BLE    NOMEM                ;    PACKETS THAN CONTROLLERS UNDER TEST
7443 062464 012703 006242              MOV    #T.HDR,R3           ; POINT TO HEADER FORWARD LINK
7444 062470 005001                     CLR    R1                   ; 0 INDICATES HEADER
7445 062472 010164 000002         51$: MOV    R1,ME.CBL(R4)       ; INIT PACKET BACK LINK
7446 062476 010463 000000              MOV    R4,ME.CFL(R3)       ; INIT PREVIOUS FORWARD LINK
7447 062502 010264 000004              MOV    R2,ME.CRF(R4)       ; INIT COMMAND REFERENCE NUMBER
7448 062506 010403                     MOV    R4,R3                ; POINT TO LAST PACKET
7449 062510 010401                     MOV    R4,R1                ; GET ADDRESS OF LAST PACKET
7450 062512 062704 000122              ADD    #<2*CMDSIZ>,R4      ; POINT TO NEXT PACKET
7451 062516 005302                     DEC    R2                   ; IF MORE PACKETS,
7452 062520 001364                     BNE    51$                  ;    LOOP
7453 062522 005063 000000              CLR    ME.CFL(R3)          ; 0 INDICATES HEADER
7454
7455 062526 012700 000042         60$: MOV    #34.,R0              ; CLEAR
7456 062532 012703 004516              MOV    #MUTSIZ,R3          ;    OUTPUT
7457 062536 012701 004412              MOV    #MUTOFF,R1          ;       BUFFER
7458 062542 005023              61$:    CLR    (R3)+               ;          MULTI-USE
7459 062544 012721 177777              MOV    #-1,(R1)+           ;             TABLE
```

```
7460 062550  005300                       DEC     RO               ;
7461 062552  001373                       BNE     61$              ;
7462                          ;
7463                          ;           EXIT INITIALIZE SECTION
7464                          ;
7465 062554  042737  070000  002204 INITXX: BIC    #IDIE!IERL!IDUN,IFLAGS  ; CLEAR LEFTOVER FLAGS
7466 062562                          SETPRI  #PRIOO           ; SET RUNNING PRIORITY TO ZERO
     062562  012700  000000          MOV     #PRIOO,RO
     062566  104441                  TRAP    C$SPRI
7467
7468 062570                          EXIT    INIT
     062570  104432                  TRAP    C$EXIT
     062572  000014                  .WORD   L10050-.
7469
7470                          ;
7471                          ;           INSUFFICIENT MEMORY ERROR
7472                          ;
7473
7474 062574                   NOMEM:  ERRSF 4,,ERRO04
     062574  104454                  TRAP    C$ERSF
     062576  000004                  .WORD   4
     062600  000000                  .WORD   0
     062602  017172                  .WORD   ERRO04
7475 062604                          DOCLN                    ; DO CLEAN-UP TRAP
     062604  104444                  TRAP    C$DCLN
7476
7477 062606                          ENDINIT
     062606                   L10050:
     062606  104411                  TRAP    C$INIT
7478
```

```
7480                                    .SBTTL   AUTODROP SECTION
7481
7482                                    ;++
7483                                    ;        THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
7484                                    ;        THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
7485                                    ;        SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
7486                                    ;        DROPPED FROM TESTING.
7487                                    ;--
7488
7489  062610                                    BGNAUTO
      062610                            L$AUTO::
7490
7491  062610                                    ENDAUTO
      062610                            L10051:
      062610   104461                           TRAP     C$AUTO
```

```
7493                                      .SBTTL  CLEANUP CODING SECTION
7494
7495                                 ;++
7496                                 ;         THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
7497                                 ;         AFTER EACH PASS AND AFTER THE PROGRAM IS INTERRUPTED BY "+C".
7498                                 ;--
7499
7500  062612                              BGNCLN
      062612                         L$CLEAN::
7501
7502  062612  004737  053420             CALL    REINIT              ; RESET ALL CONTROLLERS
7503  062616  032737  100000  002204      BIT     #IMMU,IFLAGS        ; IF IMMU SET,
7504  062624  001001                      BNE     10$                 ;   MEMORY MAPPING NOT ON
7505  062626                              BUFREL                      ; RELEASE MAPPED HIGH MEMORY
      062626  104477                      TRAP    C$REL
7506  062630                         10$:
7507  062630                              ENDCLN
      062630                         L10052:
      062630  104412                      TRAP    C$CLEAN
7508
```

```
7510                             .SBTTL  DROP UNIT SECTION
7511
7512                             ;++
7513                             ;       THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
7514                             ;       TO NO LONGER BE TESTED.
7515                             ;--
7516
7517 062632                             BGNDU
     062632              L$DU::
7518
7519 062632                             ENDDU
     062632              L10053:
     062632   104453             TRAP    C$DU
7520
```

```
7522                              .SBTTL  ADD    UNIT SECTION
7523
7524                              ;++        THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
7525                              ;          TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
7526                              ;          TO THE TEST CYCLE.
7527                              ;
7528                              ;--
7529
7530 062634                              BGNAU
     062634              L$AU::

7531
7532 062634                              ENDAU
     062634              L10054:
     062634  104452              TRAP    C$AU
7533
```

```
7535                                    .SBTTL HARDWARE TESTS
7536
7537                                    .SBTTL  TEST 1 - CONTROLLER VERIFICATION TEST
7538
7539                                    ;++
7540                                    ;       TEST 1 - CONTROLLER VERIFICATION TEST
7541
7542                                    ;       THIS TEST WILL VERIFY THAT ALL OF THE CONTROLLERS SPECIFIED IN
7543                                    ;       THE HARDWARE QUESTIONS ARE REACHABLE, WILL PASS THEIR SELF
7544                                    ;       DIAGNOSTICS, AND CAN SUCCESSFULLY COMMUNICATE WITH THE HOST.
7545                                    ;       EACH CONTROLLER IS TESTED SEPERATELY. NO DRIVES ARE ACCESSED IN
7546                                    ;       THIS TEST.
7547                                    ;--
7548
7549  062636                            BGNTST
      062636                    T1::
7550
7551  062636  012737  000001  002206            MOV     #1,TNUM              ; SAVE TEST NUMBER
7552  062644  012737  040734  002246            MOV     #T1PGEN,TSPGEN       ; GET ADDRESS OF PKT GEN ROUTINE
7553
7554                                    ;++
7555                                    ;       TEST 1, SUBTEST 1
7556                                    ;       HOST MEMORY I/O PAGE ADDRESSING AND CONTROLLER DIAGNOSTIC WRAP
7557                                    ;       MODE TEST
7558                                    ;--
7559
7560  062652                            BGNSUB; 1
      062652                    T1.1:
      062652  104402                    TRAP    C$BSUB
7561
7562  062654  013705  002200            MOV     CTABS,R5             ; GET CONTROLLER TABLE ADDRESS
7563  062660  005037  002210            CLR     UTEST                ; CLEAR DRIVE TO TEST COUNT
7564  062664  116537  000002  002074 T1NEXT: MOVB  C.UNIT(R5),L$LUN  ; CHECK IF UNIT AVAILABLE FOR TESTING
7565  062672  032765  100000  000014    BIT     #CT.AVL,C.FLG(R5)    ;
7566  062700  001106                    BNE     T1SKIP               ; SKIP IF NOT AVAILABLE
7567  062702  005237  002210            INC     UTEST                ; COUNT NUMBER OF CONTROLLERS TO TEST
7568                                    ;
7569                                    ;       VERIFY THE ADDRESS OF THE SA AND IP REGISTERS ARE VALID AND
7570                                    ;       START THE INITIALIZATION BY WRITING ZEROS TO IP REGISTER
7571                                    ;
7572  062706  016504  000000            MOV     C.IPR(R5),R4         ; GET ADDRESS OF IP REGISTER
7573  062712  005037  002264            CLR     NXMAD                ; CLEAR MEMORY ERROR FLAG
7574  062716                            SETVEC  #ERRVEC,#NXMI,#PRIO7 ; SETUP TIMEOUT ERROR VECTOR
      062716  012746  000340            MOV     #PRIO7,-(SP)
      062722  012746  054472            MOV     #NXMI,-(SP)
      062726  012746  000004            MOV     #ERRVEC,-(SP)
      062732  012746  000003            MOV     #3,-(SP)
      062736  104437                    TRAP    C$SVEC
      062740  062706  000010            ADD     #10,SP
7575  062744  005764  000002            TST     2(R4)                ; ACCESS SA REGISTER
7576  062750  005014                    CLR     (R4)                 ; WRITE TO IP
7577  062752                            CLRVEC  #ERRVEC              ; RETURN TIMEOUT ERROR VECTOR
      062752  012700  000004            MOV     #ERRVEC,R0
      062756  104436                    TRAP    C$CVEC
7578  062760  005737  002264            TST     NXMAD                ; SEE IF A MEMORY ERROR OCCURRED
```

```
7579 062764  001407                          BEQ    1$                          ; IF NO ERROR, CONTINUE
7580 062766                                  ERRDF  37,,ERRO37                   ; PRINT ERROR MESSAGE
     062766  104455                          TRAP   C$ERDF
     062770  000045                          .WORD  37
     062772  000000                          .WORD  0
     062774  020510                          .WORD  ERRO37
7581 062776  004737  027350                  CALL   DRPCNT                      ; DROP CONTROLLER FROM TEST
7582 063002  000445                          BR     T1SKIP                      ; END TEST NOW
7583
7584                                  ;
7585                                  ;      PERFORM PORT WRAP TEST USING A 1 SHIFTED THROUGH A FIELD OF
7586                                  ;      ZEROS THEN AGAIN WITH A 0 SHIFTED THROUGH A FIELD OF ONES.
7587                                  ;
7588
7589 063004  012737  004000  002354  1$:     MOV    #SA.S1,CNTRSC               ; SET STEP 1 BIT
7590 063012  004737  053252                  CALL   CNTRSP                      ; WAIT FOR RESPONSE FROM STEP 1
7591 063016  103437                          BCS    T1SKIP                      ; IF ERROR, BRANCH
7592 063020  032702  000400                  BIT    #SA.DI,R2                   ; IF CTRLR DOESN'T HAVE DIAGNOSTICS,
7593 063024  001434                          BEQ    T1SKIP                      ;     TRY NEXT CONTROLLER
7594 063026  012737  140000  002356          MOV    #<SA.STP+SA.WRP>,WCHNGD     ; SAVE NEW PORT CONTENTS
7595 063034  012703  000104                  MOV    #68.,R3                     ; INITIALIZE PATTERN POINTER
7596 063040  000403                          BR     5$                          ;
7597 063042  016337  007036  002356  4$:     MOV    MNTPAT(R3),WCHNGD           ; SAVE NEW PORT CONTENTS
7598 063050  013764  002356  000002  5$:     MOV    WCHNGD,2(R4)                ; INITIALIZE FOR PORT WRAP
7599 063056  004737  053652                  CALL   WCHNG                       ; WAIT FOR THE PORT TO CHANGE
7600 063062  103415                          BCS    T1SKIP                      ; IF ERROR, BRANCH
7601 063064  023702  002356                  CMP    WCHNGD,R2                   ; COMPARE WITH DATA WRITTEN
7602 063070  001407                          BEQ    7$                          ; IF MATCH, BRANCH
7603 063072                          6$:     ERRDF  26,,ERRO26                  ; REPORT ERROR
     063072  104455                          TRAP   C$ERDF
     063074  000032                          .WORD  26
     063076  000000                          .WORD  0
     063100  020126                          .WORD  ERRO26
7604 063102  004737  027350                  CALL   DRPCNT                      ; DROP CONTROLLER FROM TEST
7605 063106  000403                          BR     T1SKIP                      ; BRANCH
7606 063110  162703  000002          7$:     SUB    #2,R3                       ; POINT TO NEXT DATA PATTERN
7607 063114  002352                          BGE    4$                          ; IF NOT NEGATIVE, KEEP LOOPING
7608
7609 063116  062705  000170          T1SKIP: ADD    #C.SIZE,R5                  ; MOVE TO NEXT CONTROLLER TABLE
7610 063122  005715                          TST    (R5)                        ; SEE IF ANOTHER TABLE AND
7611 063124  001257                          BNE    T1NEXT                      ; IF NOT ZERO, LOOK AT NEXT CONTROLLER
7612
7613 063126                                  ENDSUB
     063126                          L10056: 
     063126  104403                          TRAP   C$ESUB
7614
```

```
7616
7617                              ;++
7618                              ;           TEST 1, SUBTEST 2
7619                              ;           INITIALIZE CONTROLLER THEN PERFORM LARGE BUFFER MAINTENANCE
7620                              ;           READ AND WRITE OPERATIONS TO ALL LOCATIONS IN CONTROLLER MEMORY
7621                              ;           TO TEST CONTROLLER ADDRESSING ABILITY AND MEMORY INTEGRITY
7622                              ;--
7623
7624  063130                             BGNSUB; 2
       063130                      T1.2:
       063130   104402                   TRAP      C$BSUB
7625
7626  063132   004737  053420            CALL      REINIT                      ; RESET ALL CONTROLLERS
7627
7628  063136   013705  002200            MOV       CTABS,R5                    ; GET CONTROLLER TABLE ADDRESS
7629  063142   005037  002210            CLR       UTEST                       ; CLEAR DRIVE TO TEST COUNT
7630  063146   032765  100000  000014 41$: BIT     #CT.AVL.C.FLG(R5)           ; IF NOT AVAILABLE OR DONE,
7631  063154   001002                    BNE       42$                         ;    DON'T COUNT CONTROLLER
7632  063156   005237  002210            INC       UTEST                       ; COUNT NUMBER OF CONTROLLERS TO TEST
7633  063162   062705  000170     42$:   ADD       #C.SIZE,R5                  ; MOVE TO NEXT CONTROLLER TABLE
7634  063166   005715                    TST       (R5)                        ; SEE IF ANOTHER TABLE AND
7635  063170   001366                    BNE       41$                         ; IF NOT ZERO, LOOK AT NEXT CONTROLLER
7636  063172   012737  177777  002242    MOV       #-1,TSTOFF                  ; -1 INDICATES NO BUFFER IN USE
7637  063200   004737  026426            CALL      DRIVER                      ; PERFORM SUBTEST
7638
7639  063204                             ENDSUB
       063204                      L10057:
       063204   104403                   TRAP      C$ESUB
7640
7641  063206                             DORPT                                 ; PRINT A STATISTICAL REPORT
       063206   104424                   TRAP      C$DRPT
7642
7643  063210                             ENDTST
       063210                      L10055:
       063210   104401                   TRAP      C$ETST
```

```
7645                                    .SBTTL  TEST 2 - SUBSYSTEM VERIFICATION TEST
7646
7647                             ;++
7648                             ;        TEST 2 - SUBSYSTEM VERIFICATION TEST
7649                             ;        THIS TEST VERIFIES THE ABILITY OF THE SUBSYSTEM TO BRING THE
7650                             ;        DRIVES ONLINE AND TO PERFORM I/O OPERATIONS TO THE DRIVES.
7651                             ;--
7652
7653  063212                             BGNTST
       063212                     T2::
7654
7655  063212  012737  000002  002206     MOV    #2,TNUM            ; SAVE TEST NUMBER
7656  063220  012737  041616  002246     MOV    #T2PGEN,TSPGEN     ; GET ADDRESS OF PKT GEN ROUTINE
7657
7658  063226  004737  053420             CALL   REINIT            ; RESET ALL CONTROLLERS
7659
7660  063232  013705  002200             MOV    CTABS,R5          ; GET CONTROLLER TABLE ADDRESS
7661  063236  005037  002210             CLR    UTEST             ; CLEAR DRIVE TO TEST COUNT
7662  063242  032765  100000  000014 41$: BIT    #CT.AVL,C.FLG(R5) ; IF NOT AVAILABLE OR DONE,
7663  063250  001002                      BNE    42$               ;    DON'T COUNT CONTROLLER
7664  063252  005237  002210             INC    UTEST             ; COUNT NUMBER OF CONTROLLERS TO TEST
7665  063256  062705  000170      42$:   ADD    #C.SIZE,R5        ; MOVE TO NEXT CONTROLLER TABLE
7666  063262  005715                     TST    (R5)              ; SEE IF ANOTHER TABLE AND
7667  063264  001366                     BNE    41$               ; IF NOT ZERO, LOOK AT NEXT CONTROLLER
7668  063266  012737  177777  002242     MOV    #-1,TSTOFF        ; -1 INDICATES NO BUFFER USED
7669  063274  004737  026426             CALL   DRIVER            ; PERFORM THE TEST
7670  063300                             DORPT                    ; PRINT A STATISTICAL REPORT
       063300  104424                     TRAP   C$DRPT
7671
7672  063302                             ENDTST
       063302                     L10060:
       063302  104401                     TRAP   C$ETST
7673
```

```
7675                                        .SBTTL  TEST 3 - SUBSYSTEM EXERCISER
7676
7677                                ;++
7678                                ;       TEST 3 - SUBSYSTEM EXERCISER
7679                                ;       THIS TEST EXERCISES THE COMPLETE SUBSYSTEM. THE TEST WILL MAINTAIN
7680                                ;       AS HEAVY AN I/O LOAD AS POSSIBLE ON THE SUBSYSTEM
7681                                ;--
7682
7683 063304                                 BGNTST
     063304                        T3::
7684
7685 063304 012737 000003 002206           MOV     #3,TNUM                 ; SAVE TEST NUMBER
7686 063312 012737 042736 002246           MOV     #T3PGEN,TSPGEN          ; GET ADDRESS OF PKT GEN ROUTINE
7687
7688 063320 032737 000002 002204           BIT     #ICONT,IFLAGS           ; IF HERE FROM CONTINUE COMMAND,
7689 063326 001014                          BNE     10$                     ;   DON'T RE-INIT TIMERS
7690 063330 012700 002250                   MOV     #STIME,R0               ; GET REPORT TIMER
7691 063334 013701 002150                   MOV     SFPTBL+SO.SRI,R1        ; GET REPORT INTERVAL
7692 063340 004737 052144                   CALL    SETMIN                  ; SET TIME FOR NEXT REPORT
7693 063344 012700 002254                   MOV     #TSTIM,R0               ; GET TEST TIMER
7694 063350 013701 002146                   MOV     SFPTBL+SO.TL,R1         ; GET TIME OUT INTERVAL
7695 063354 004737 052144                   CALL    SETMIN                  ; SET TIME FOR NEXT REPORT
7696                                ;
7697                                ;       ASK MANUAL INTERVENTION QUESTIONS
7698                                ;
7699 063360 032737 000016 002204   10$:      BIT     #ICONT!ISTRT!IREST,IFLAGS; IF NOT HERE FROM OPERATOR COMMAND,
7700 063366 001424                           BEQ     30$                     ;   USE PREVIOUS PARAMETERS
7701 063370 042737 000016 002204            BIC     #ICONT!ISTRT!IREST,IFLAGS; CLEAR FLAGS FOR NEXT TIME HERE
7702 063376 032737 000001 002154            BIT     #SM.MAN,SFPTBL+SO.BIT   ; IF MANUAL INTERVENTION NOT SELECTED,
7703 063404 001415                          BEQ     30$                     ;   USE DEFAULT PARAMETERS
7704 063406                                 MANUAL                          ; IF MANUAL INTERVENTION NOT ALLOWED,
     063406 104450                           TRAP    C$MANI
7705 063410                                 BNCOMPLETE 24$                  ;   GIVE WARNING & USE DEFAULTS
     063410 103003                           BCC     24$
7706
7707 063412 004737 063520                   CALL    T3QUES                  ; ASK MANUAL INTERVENTION QUESTIONS
7708 063416 000410                          BR      30$                     ;   AND START TEST
7709
7710 063420                         24$:     PRINTF  #T3WARN                 ; MANUAL INTERVENTION NOT ALLOWED
     063420 012746 010625                    MOV     #T3WARN,-(SP)
     063424 012746 000001                    MOV     #1,-(SP)
     063430 010600                           MOV     SP,R0
     063432 104417                           TRAP    C$PNTF
     063434 062706 000004                    ADD     #4,SP
7711                                ;
7712                                ;       START TEST
7713                                ;
7714 063440 004737 053420          30$:      CALL    REINIT                  ; RESET ALL CONTROLLERS
7715 063444 013705 002200                    MOV     CTABS,R5                ; GET CONTROLLER TABLE ADDRESS
7716 063450 005037 002210                    CLR     UTEST                   ; CLEAR DRIVE TO TEST COUNT
7717 063454 032765 100000 000014  41$:      BIT     #CT.AVL,C.FLG(R5)       ; IF NOT AVAILABLE,
7718 063462 001002                           BNE     42$                     ;   DON'T COUNT CONTROLLER
7719 063464 005237 002210                    INC     UTEST                   ; COUNT NUMBER OF CONTROLLERS TO TEST
7720 063470 062705 000170          42$:      ADD     #C.SIZE,R5              ; MOVE TO NEXT CONTROLLER TABLE
7721 063474 005715                           TST     (R5)                    ; SEE IF ANOTHER TABLE AND
```

```
     7722 063476  001366                        BNE    41$               ; IF NOT ZERO, LOOK AT NEXT CONTROLLER
     7723 063500  012737  177777  002242        MOV    #-1,TSTOFF        ; -1 INDICATES NO BUFFER IN USE
     7724
     7725 063506  004737  026426                CALL   DRIVER            ; PERFORM TEST
     7726 063512                                 DORPT                    ; PRINT A STATISTICAL REPORT
          063512  104424                         TRAP   C$DRPT
     7727
     7728 063514                                 EXIT   TST
          063514  104432                         TRAP   C$EXIT
          063516  001472                         .WORD  L10061-.
     7729
```

```
7731                                        .SBTTL   TEST 3 MANUAL INTERVENTION QUESTIONS
7732
7733                                        ;++
7734                                        ;        T3QUES - ASK TEST 3 MANUAL INTERVENTION QUESTIONS
7735                                        ;
7736                                        ;        INPUTS:
7737                                        ;            NONE
7738                                        ;        OUTPUTS:
7739                                        ;            DRIVE TABLES CONTAIN NEW PARAMETERS
7740                                        ;            DATA PATTERN 0 CHANGED, IF REQUESTED
7741                                        ;
7742                                        ;--
7743  063520                               T3QUES: PRINTF   #T3QHDR                      ; PRINT HEADER
       063520  012746  064625                      MOV      #T3QHDR,-(SP)
       063524  012746  000001                      MOV      #1,-(SP)
       063530  010600                              MOV      SP,R0
       063532  104417                              TRAP     C$PNTF
       063534  062706  000004                      ADD      #4,SP
7744  063540                                       GMANID   T3PAT,TS.PAT,D,-1,0.,16..YES; DATA PATTERN TO USE?
       063540  104443                              TRAP     C$GMAN
       063542  000406                              BR       10000$
       063544  002260                              .WORD    TS.PAT
       063546  000052                              .WORD    T$CODE
       063550  064502                              .WORD    T3PAT
       063552  177777                              .WORD    -1
       063554  000000                              .WORD    T$LOLIM
       063556  000020                              .WORD    T$HILIM
       063560                               10000$:
7745  063560  005737  002260                      TST      TS.PAT                       ; IF ZERO,
7746  063564  001404                              BEQ      10$                          ;   ASK ABOUT PATTERN 16
7747  063566  022737  000020  002260              CMP      #16.,TS.PAT                  ; IF NOT PATTERN 16,
7748  063574  001050                              BNE      20$                          ;   SKIP PATTERN 16 QUESTIONS
7749
7750  063576                               10$:     GMANIL   T3P16,IFLAGS,IPAT16,YES     ; CHANGE PATTERN 16?
       063576  104443                              TRAP     C$GMAN
       063600  000404                              BR       10001$
       063602  002204                              .WORD    IFLAGS
       063604  000130                              .WORD    T$CODE
       063606  064526                              .WORD    T3P16
       063610  000100                              .WORD    IPAT16
       063612                               10001$:
7751  063612  032737  000100  002204              BIT      #IPAT16,IFLAGS               ; IF NOT SET,
7752  063620  001436                              BEQ      20$                          ;   USE DEFAULT 16
7753
7754  063622  012703  006250                      MOV      #PAT00+2,R3                  ; GET POINTER TO PATTERN 16 SIZE
7755  063626  011337  002212                      MOV      (R3),TEMP                    ; GET PATTERN SIZE
7756  063632                                       GMANID   T3NWD,TEMP,D,-1,1..16..YES; NUMBER OF DATA WORDS?
       063632  104443                              TRAP     C$GMAN
       063634  000406                              BR       10002$
       063636  002212                              .WORD    TEMP
       063640  000052                              .WORD    T$CODE
       063642  064555                              .WORD    T3NWD
       063644  177777                              .WORD    -1
       063646  000001                              .WORD    T$LOLIM
       063650  000020                              .WORD    T$HILIM
```

```
         063652                         10002$:
7757 063652   013723  002212              MOV     TEMP,(R3)+              ; SAVE PATTERN SIZE
7758 063656   013702  002212              MOV     TEMP,R2                ; GET COUNT OF DATA WORDS
7759 063662   011337  002212              MOV     (R3),TEMP              ; GET LAST PATTERN
7760 063666                       12$:    GMANID  T3WRD,TEMP,0,-1,0,,-1,,YES; DATA WORD?
     063666   104443                       TRAP    C$GMAN
     063670   000406                       BR      10003$
     063672   002212                       .WORD   TEMP
     063674   000032                       .WORD   T$CODE
     063676   064612                       .WORD   T3WRD
     063700   177777                       .WORD   -1
     063702   000000                       .WORD   T$LOLIM
     063704   177777                       .WORD   T$HILIM
     063706                         10003$:
7761 063706   013723  002212              MOV     TEMP,(R3)+             ; SAVE DATA WORD
7762 063712   005302                       DEC     R2                    ; DECREMENT WORD COUNTER
7763 063714   001364                       BNE     12$                   ; IF NOT ZERO, GET NEXT WORD
7764
7765 063716   013705  002200      20$:    MOV     CTABS,R5               ; GET ADDRESS OF 1ST CONTROLLER TABLE
7766 063722   012702  000004      21$:    MOV     #4,R2                  ; GET COUNT OF DRIVE TABLES
7767 063726   010504                       MOV     R5,R4                 ; GET FIRST DRIVE TABLE POINTER
7768 063730   062704  000016              ADD     #C.DR0,R4              ;
7769 063734   012403              22$:    MOV     (R4)+,R3               ; IF NO MORE DRIVES,
7770 063736   001002                       BNE     23$                   ;    GET NEXT
7771 063740   000137  064340              JMP     41$                   ;       CONTROLLER TABLE
7772 063744   032763  100000  000002  23$: BIT    #DT.AVL,D.UNIT(R3)    ; IF DRIVE IS TO BE TESTED,
7773 063752   001402                       BEQ     30$                   ;    ASK DRIVE QUESTIONS
7774 063754   000137  064330              JMP     40$                   ;
7775
7776 063760                       30$:    PUSH    <R4>                   ; SAVE REGISTERS
     063760   010446                       MOV     R4,-(SP)              ; PUSH R4 ON STACK
7777 063762   016337  000002  002212      MOV     D.UNIT(R3),TEMP       ; GET DRIVE
7778 063770   042737  177700  002212      BIC     #↑CDT.UNT,TEMP        ;    UNIT NUMBER
7779 063776                       PRINTF  #T3QHED,TEMP,(R5),(R3)         ; PRINT HEADER
     063776   011346                       MOV     (R3),-(SP)
     064000   011546                       MOV     (R5),-(SP)
     064002   013746  002212              MOV     TEMP,-(SP)
     064006   012746  064701              MOV     #T3QHED,-(SP)
     064012   012746  000004              MOV     #4,-(SP)
     064016   010600                       MOV     SP,R0
     064020   104417                       TRAP    C$PNTF
     064022   062706  000012              ADD     #12,SP
7780 064026   016337  000002  002212      MOV     D.UNIT(R3),TEMP       ; GET DRIVE FLAGS
7781 064034                       GMANIL  T3TED,TEMP,DT.BES,YES          ; TEST OVER ENTIRE DISK?
     064034   104443                       TRAP    C$GMAN
     064036   000404                       BR      10004$
     064040   002212                       .WORD   TEMP
     064042   000130                       .WORD   T$CODE
     064044   064360                       .WORD   T3TED
     064046   020000                       .WORD   DT.BES
     064050                         10004$:
7782 064050   013763  003212  000002      MOV     TEMP,D.UNIT(R3)       ; CHECK FOR BIT 15 SET
7783 064056   032763  020000  000002      BIT     #DT.BES,D.UNIT(R3)    ; IF USING ENTIRE DISK,
7784 064064   001103                       BNE     38$                   ;    SKIP B/E SET QUESTIONS
7785 064066   016337  000004  002212      MOV     D.BEC(R3),TEMP        ; GET PREVIOUS B/E SET COUNT
```

```
7786 064074                              GMANID  T3BES,TEMP,D,-1,1,4,YES ; NUMBER OF B/E SETS?
     064074  104443                      TRAP    C$GMAN
     064076  000406                      BR      10005$
     064100  002212                      .WORD   TEMP
     064102  000052                      .WORD   T$CODE
     064104  064412                      .WORD   T3BES
     064106  177777                      .WORD   -1
     064110  000001                      .WORD   T$LOLIM
     064112  000004                      .WORD   T$HILIM
     064114                      10005$:
7787 064114  013701  002212              MOV     TEMP,R1                 ; INIT LOOP COUNTER
7788 064120  010163  000004              MOV     R1,D.BEC(R3)            ; SAVE NEW B/E SET COUNT
7789 064124  010304                      MOV     R3,R4                   ; GET POINTER TO STORAGE AREA
7790 064126  062704  000006              ADD     #D.BGN1,R4              ; POINT TO START OF 1ST B/E SET
7791 064132  004737  047552      33$:    CALL    BLDSTR                  ; BUILD DEFAULT VALUE
7792 064136                              GMANID  T3BEG,TEMP,A,-1,0,9.,YES; BEGIN BLOCK?
     064136  104443                      TRAP    C$GMAN
     064140  000406                      BR      10006$
     064142  002212                      .WORD   TEMP
     064144  000152                      .WORD   T$CODE
     064146  064452                      .WORD   T3BEG
     064150  177777                      .WORD   -1
     064152  000000                      .WORD   T$LOLIM
     064154  000011                      .WORD   T$HILIM
     064156                      10006$:
7793 064156  004737  047616              CALL    BLDLWD                  ; CHECK RESPONSE & SAVE IN TABLE
7794 064162  103763                      BCS     33$                     ; BRANCH ON ERROR
7795 064164  004737  047552      34$:    CALL    BLDSTR                  ; BUILD DEFAULT VALUE
7796 064170                              GMANID  T3END,TEMP,A,-1,0,9.,YES; END BLOCK?
     064170  104443                      TRAP    C$GMAN
     064172  000406                      BR      10007$
     064174  002212                      .WORD   TEMP
     064176  000152                      .WORD   T$CODE
     064200  064467                      .WORD   T3END
     064202  177777                      .WORD   -1
     064204  000000                      .WORD   T$LOLIM
     064206  000011                      .WORD   T$HILIM
     064210                      10007$:
7797 064210  004737  047616              CALL    BLDLWD                  ; CHECK RESPONSE & SAVE IN TABLE
7798 064214  103763                      BCS     34$                     ; BRANCH ON ERROR
7799 064216  026464  177776  177772      CMP     -2(R4),-6(R4)           ; IF ENDING LBN
7800 064224  101020                      BHI     37$                     ;   LESS THAN
7801 064226  103404                      BLO     35$                     ;     STARTING LBN,
7802 064230  026464  177774  177770      CMP     -4(R4),-8.(R4)          ;       PRINT ERROR
7803 064236  103013                      BHIS    37$                     ;         AND RE-PROMPT
7804 064240                      35$:    PRINTF  #INP28C                 ;
     064240  012746  065123              MOV     #INP28C,-(SP)
     064244  012746  000001              MOV     #1,-(SP)
     064250  010600                      MOV     SP,R0
     064252  104417                      TRAP    C$PNTF
     064254  062706  000004              ADD     #4,SP
7805 064260  162704  000010              SUB     #8.,R4
7806 064264  000722                      BR      33$                     ;
7807 064266  005301      37$:    DEC     R1                      ; DECREMENT LOOP COUNT
7808 064270  001320                      BNE     33$                     ; IF NOT DONE, LOOP
```

```
7809 064272  000415                              BR      39$                    ; USE
7810 064274  012763  000001  000004  38$:  MOV   #1,D.BEC(R3)             ;   DEFAULT
7811 064302  005063  000006              CLR   D.BGN1(R3)                ;     BEGIN/END
7812 064306  005063  000010              CLR   D.BGN1+2(R3)              ;       SET
7813 064312  012763  177777  000012      MOV   #-1,D.END1(R3)            ;         (ENTIRE DISK)
7814 064320  012763  177777  000014      MOV   #-1,D.END1+2(R3)          ;
7815 064326                        39$:  POP   <R4>                     ; RESTORE DRIVE TABLE POINTER
7816
7817 064330  005302              40$:  DEC   R2                        ; IF MORE DRIVE TABLES,
7818 064332  001402                    BEQ   41$                       ;   ASK QUESTIONS FOR
7819 064334  000137  063734            JMP   22$                       ;     NEXT DRIVE
7820 064340  062705  000170      41$:  ADD   #C.SIZE,R5                ; POINT TO NEXT CONTROLLER TABLE
7821 064344  005715                    TST   (R5)                      ; IF MORE CONTROLLER TABLES,
7822 064346  001402                    BEQ   42$                       ;   ASK QUESTIONS FOR
7823 064350  000137  063722            JMP   21$                       ;     NEXT CONTROLLER
7824 064354                      42$:
7825 064354  000240              NOP                     ; [DBG] INSERT HALT FOR DEBUG
7826 064356  000207              RETURN
7827
7828                             ;
7829                             ;      UNFORMATTED QUESTIONS
7830                             ;
7831 064360  124   105   123    T3TED:  .ASCIZ  \TEST OVER THE ENTIRE DISK\
7832 064412  116   125   115    T3BES:  .ASCIZ  \NUMBER OF BEGIN/END SETS TO USE\
7833 064452  123   124   101    T3BEG:  .ASCIZ  \STARTING LBN\
7834 064467  105   116   104    T3END:  .ASCIZ  \ENDING LBN\
7835 064502  104   101   124    T3PAT:  .ASCIZ  \DATA PATTERN TO USE\
7836 064526  115   117   104    T3P16:  .ASCIZ  \MODIFY DATA PATTERN 16\
7837 064555  110   117   127    T3NWD:  .ASCIZ  \HOW MANY WORDS IN PATTERN 16\
7838 064612  104   101   124    T3WRD:  .ASCIZ  \DATA WORD \
7839                             ;
7840                             ;      FORMATTED QUESTIONS
7841                             ;
7842 064625  045   116   045    T3QHDR: .ASCIZ  \%N%ATEST 3 MANUAL INTERVENTION QUESTIONS:%N\
7843 064701  045   116   045    T3QHED: .ASCIZ  \%N%ATHESE QUESTIONS REFER TO UNIT %D2%A CONTROLLER AT %O6%A DRIVE %D3%N\
7844 065011  045   101   114    INP28A: .ASCIZ  \%ALIMITS - LO= 0, HI= 4294967295%N\
7845 065054  045   101   111    INP28B: .ASCIZ  \%AINVALID INPUT, TYPE DECIMAL NUMBER%N\
7846 065123  045   101   111    INP28C: .ASCIZ  \%AINVALID INPUT, ENDING LBN LESS THAN STARTING LBN%N\
                                        .EVEN
7847
7848                             ENDTST
7849 065210
     065210                     L10061:
     065210  104401             TRAP    C$ETST
7850
```

```
 7852                                    .SBTTL   HARDWARE PARAMETER CODING SECTION
 7853
 7854                             ;++
 7855                             ;        THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
 7856                             ;        THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
 7857                             ;        MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
 7858                             ;        INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
 7859                             ;        MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
 7860                             ;        WITH THE OPERATOR.
 7861                             ;--
 7862 065212                              BGNHRD
      065212   000014                     .WORD    L10062-L$HARD/2
      065214                      L$HARD::
 7863
 7864 065214                      TABLE                                    ;START A TABLE DEFINITION
 7865 065214                      ITEM H.IPR         2        <CSR ADDRESS>
 7866 065214                      ITEM H.DRV         2        <DRIVE NUMBER>
 7867 065214                      ITEM H.PRM         2        <PROGRAM PARAMETERS>
 7868
 7869          010000            HM.WRT   == BIT12                         ; WRITE ON CUSTOMER DATA AREA
 7870 065214                      END
 7871
 7872 065214                              GPRMA    MSGIPR,H.IPR,0,160000,177774,YES; CSR ADDRESS OF CONTROLLER?
      065214   000031                     .WORD    T$CODE
      065216   065244                     .WORD    MSGIPR
      065220   160000                     .WORD    T$LOLIM
      065222   177774                     .WORD    T$HILIM
 7873
 7874 065224                              GPRMD    MSGLDR,H.DRV,D,-1,0.,254.,YES; DRIVE #?
      065224   001052                     .WORD    T$CODE
      065226   065276                     .WORD    MSGLDR
      065230   177777                     .WORD    -1
      065232   000000                     .WORD    T$LOLIM
      065234   000376                     .WORD    T$HILIM
 7875
 7876 065236                              GPRML    MSGCST,H.PRM,HM.WRT,YES ; WRITE ON CUSTOMER DATA AREA?
      065236   002130                     .WORD    T$CODE
      065240   065306                     .WORD    MSGCST
      065242   010000                     .WORD    HM.WRT
 7877
 7878 065244                              ENDHRD
                                          .EVEN
      065244                      L10062:
 7879
 7880 065244   103   123   122   MSGIPR: .ASCIZ   \CSR ADDRESS OF CONTROLLER\
 7881 065276   104   122   111   MSGLDR: .ASCIZ   \DRIVE #\
 7882 065306   127   122   111   MSGCST: .ASCIZ   \WRITE ON CUSTOMER DATA AREA\
 7883                                     .EVEN
 7884
```

```
7886                                        .SBTTL   SOFTWARE PARAMETER CODING SECTION
7887
7888                                  ;++
7889                                  ;       THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
7890                                  ;       THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
7891                                  ;       MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7892                                  ;       INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
7893                                  ;       MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7894                                  ;       WITH THE OPERATOR.
7895                                  ;--
7896
7897 065342                                  BGNSFT
     065342  000061                           .WORD  L10063-L$SOFT/2
     065344                          L$SOFT::
7898 065344                          TABLE                                 ; START A TABLE DEFINITION
7899 065344                          ITEM SO.EL      2       <HARD ERROR LIMIT>
7900 065344                          ITEM SO.TL      2       <TIME LIMIT (MINUTES)>
7901 065344                          ITEM SO.SRI     2       <STATISTICAL REPORT INTERVAL>
7902 065344                          ITEM SO.MTS     2       <MAXIMUM TRANSFER SIZE>
7903 065344                          ITEM SO.BIT     2       <SINGLE BIT ANSWERS>
7904
7905         000001                  SM.MAN == BIT00                       ; MANUAL INTERVENTION MODE
7906         000002                  SM.PSE == BIT01                       ; PRINT SOFT ERRORS
7907         000004                  SM.PCK == BIT02                       ; PATTERN VERIFICATION ON READS
7908         000010                  SM.CMP == BIT03                       ; PATTERN VERIFICATION ON WRITES
7909         000020                  SM.VLT == BIT04                       ; VARIABLE LENGTH TRANSFERS
7910         000040                  SM.EER == BIT05                       ; ENABLE ERROR RETRIES
7911         000100                  SM.ECC == BIT06                       ; ENABLE ECC
7912         000200                  SM.RAN == BIT07                       ; RANDOMLY ACCESS DRIVE
7913         000400                  SM.DET == BIT08                       ; DO DETERMINISTIC PHASE
7914 065344                          END
7915
7916 065344                          DISPLAY S.HDR                         ; PRINT HEADER
     065344  000003                           .WORD  T$CODE
     065346  065506                           .WORD  S.HDR
7917
7918 065350                          GPRML S.MAN,SO.BIT,SM.MAN,YES   ; ENTER MANUAL INTERVENTION MODE?
     065350  004130                           .WORD  T$CODE
     065352  065567                           .WORD  S.MAN
     065354  000001                           .WORD  SM.MAN
7919
7920 065356                          GPRMD S.EL,SO.EL,D,-1,1.,-1..YES; ERROR LIMIT?
     065356  000052                           .WORD  T$CODE
     065360  065626                           .WORD  S.EL
     065362  177777                           .WORD  -1
     065364  000001                           .WORD  T$LOLIM
     065366  177777                           .WORD  T$HILIM
7921
7922 065370                          GPRMD S.TL,SO.TL,D,-1,0.,65535..YES; EXERCISER TIME LIMIT?
     065370  001052                           .WORD  T$CODE
     065372  065647                           .WORD  S.TL
     065374  177777                           .WORD  -1
     065376  000000                           .WORD  T$LOLIM
     065400  177777                           .WORD  T$HILIM
7923
```

```
7924 065402                              GPRMD  S.SRI,SO.SRI,D,-1,0.,65535.,YES; STATISTICAL REPORT INTERVAL?
     065402  002052                      .WORD  T$CODE
     065404  065707                      .WORD  S.SRI
     065406  177777                      .WORD  -1
     065410  000000                      .WORD  T$LOLIM
     065412  177777                      .WORD  T$HILIM
7925
7926 065414                              GPRML  S.SSF,SO.BIT,SM.PSE,YES   ; PRINT SOFT ERRORS?
     065414  004130                      .WORD  T$CODE
     065416  065753                      .WORD  S.SSF
     065420  000002                      .WORD  SM.PSE
7927
7928 065422                              GPRML  S.PCK,SO.BIT,SM.PCK,YES   ; PATTERN VERIFICATION ON READS?
     065422  004130                      .WORD  T$CODE
     065424  066005                      .WORD  S.PCK
     065426  000004                      .WORD  SM.PCK
7929
7930 065430                              GPRML  S.CMP,SO.BIT,SM.CMP,YES   ; PATTERN VERIFICATION ON WRITES?
     065430  004130                      .WORD  T$CODE
     065432  066053                      .WORD  S.CMP
     065434  000010                      .WORD  SM.CMP
7931
7932 065436                              GPRML  S.VLT,SO.BIT,SM.VLT,YES   ; VARIABLE LENGTH TRANSFERS?
     065436  004130                      .WORD  T$CODE
     065440  066122                      .WORD  S.VLT
     065442  000020                      .WORD  SM.VLT
7933
7934 065444                              GPRMD  S.MTS,SO.MTS,D,-1,1.,128.,YES; MAXIMUM TRANSFER SIZE?
     065444  003052                      .WORD  T$CODE
     065446  066160                      .WORD  S.MTS
     065450  177777                      .WORD  -1
     065452  000001                      .WORD  T$LOLIM
     065454  000200                      .WORD  T$HILIM
7935
7936 065456                              GPRML  S.EER,SO.BIT,SM.EER,YES   ; ENABLE ERROR RETRIES?
     065456  004130                      .WORD  T$CODE
     065460  066220                      .WORD  S.EER
     065462  000040                      .WORD  SM.EER
7937
7938 065464                              GPRML  S.ECC,SO.BIT,SM.ECC,YES   ; USE ECC?
     065464  004130                      .WORD  T$CODE
     065466  066245                      .WORD  S.ECC
     065470  000100                      .WORD  SM.ECC
7939
7940 065472                              GPRML  S.RAN,SO.BIT,SM.RAN,YES   ; RANDOMLY ACCESS DRIVE?
     065472  004130                      .WORD  T$CODE
     065474  066300                      .WORD  S.RAN
     065476  000200                      .WORD  SM.RAN
7941
7942 065500                              GPRML  S.DET,SO.BIT,SM.DET,YES   ; DO DETERMINISTIC PHASE?
     065500  004130                      .WORD  T$CODE
     065502  066326                      .WORD  S.DET
     065504  000400                      .WORD  SM.DET
7943
7944 065506                              ENDSFT
```

```
                                      .EVEN
      065506                    L10063:
7945
7946 065506    124    110    105  S.HDR:  .ASCIZ  \THE FOLLOWING QUESTIONS APPLY ONLY TO TEST 3:\
7947 065564    040    000              .ASCIZ  \ \
7948 065566    000                     .BYTE   0
7949 065567    105    116    124  S.MAN:  .ASCIZ  \ENTER MANUAL INTERVENTION MODE\
7950 065626    110    101    122  S.EL:   .ASCIZ  \HARD ERROR LIMIT\
7951 065647    105    130    105  S.TL:   .ASCIZ  \EXERCISER TIME LIMIT IN MINUTES\
7952 065707    115    111    116  S.SRI:  .ASCIZ  \MINUTES BETWEEN STATISTICAL REPORTS\
7953 065753    120    122    111  S.SSF:  .ASCIZ  \PRINT SOFT ERROR MESSAGES\
7954 066005    104    117    040  S.PCK:  .ASCIZ  \DO DATA PATTERN VERIFICATION ON READS\
7955 066053    104    117    040  S.CMP:  .ASCIZ  \DO DATA PATTERN VERIFICATION ON WRITES\
7956 066122    125    123    105  S.VLT:  .ASCIZ  \USE VARIABLE LENGTH TRANSFERS\
7957 066160    115    101    130  S.MTS:  .ASCIZ  \MAXIMUM TRANSFER SIZE IN BLOCKS\
7958 066220    105    116    101  S.EER:  .ASCIZ  \ENABLE ERROR RETRIES\
7959 066245    105    116    101  S.ECC:  .ASCIZ  \ENABLE ECC DATA CORRECTION\
7960 066300    122    101    116  S.RAN:  .ASCIZ  \RANDOMLY ACCESS DRIVE\
7961 066326    104    117    040  S.DET:  .ASCIZ  \DO DETERMINISTIC PHASE\
7962                                    .EVEN
7963
```

```
7965                        ;++       THIS IS A PATCH AREA THAT SHOULD BE INCLUDED IN ALL DIAGNOSTICS.
7966                        ;         THE SIZE IS ADJUSTED AS NEEDED.
7967                        ;
7968                        ;--
7969
7970 066356                $PATCH::
7971 066356                          .BLKW   32.
7972
7973 066456                          LASTAD
                                      .EVEN
     066456  066474                   .WORD T$FREE
     066460  000005                   .WORD T$SIZE
     066462                L$LAST::
7974
```

```
7976
7977                          ;++          HARDCODED P-TABLES MAY BE PLACED HERE BY USING THE SETUP MACROS.
7978                          ;            THIS SECTION IS OPTIONAL AND SHOULD BE REMOVED IF IT IS NOT BEING
7979                          ;            USED.  CHANGE THE POINTER MACRO ARGUMENT TO REFLECT THE REMOVAL.
7980                          ;
7981                          ;            THE P-TABLES ARE DELIMITED BY THE "BGNSETUP" AND "ENDSETUP" MACROS.
7982                          ;            THE "BGNSETUP" MACRO HAS ONE ARGUMENT WHICH IS THE NUMBER OF
7983                          ;            P-TABLE ENTRIES.   EACH ENTRY IS DELIMITED BY THE "BGNPTAB" AND
7984                          ;            "ENDPTAB" MACROS.  NEITHER OF THESE MACROS REQUIRE AN ARGUMENT.
7985                          ;--
7986
7987
7988 066462                               BGNSETUP        1
7989 066462                               BGNPTAB
     066462  000000                       .WORD    0
     066464  000003                       .WORD    L10066-./2-1
     066466                  L10064:
7990 066466  172150                       .WORD    172150              ; CSR ADDRESS
7991 066470  000000                       .WORD    0.                  ; DRIVE NUMBER
7992 066472  000000                       .WORD    0.                  ; WRITE COSTUMER DATA AREA
7993 066474                               ENDPTAB
     066474                  L10066:
7994 066474                               ENDSETUP
7995
7996         000001                       .END
```

```
ACCESS  050306        BLDLWD  047616        C$CLEA= 000012        C.CBAS= 000072        DBGMRD  023473
ADR   = 000020 G      BLDME   051214        C$CLOS= 000035        C.CEND= 000074        DBGMWR  023526
ADRCNT  036350        BLDSTR  047552        C$CLP1= 000006        C.CINT= 000106        DBGONL  023304
ADRDAT  035160        BOE   = 000400 G      C$CPBF= 000074        C.CMBL= 000162        DBGRD   023616
ADRDRV  037152        BUFBA   004370        C$CPME= 000075        C.CMFL= 000160        DBGRSP  024132
ALOCM   024322        BUFBLK  004374        C$CVEC= 000036        C.CPTR= 000070        DBGSCC  023174
AMAP    007544        BUFEA   004372        C$DCLN= 000044        C.CRED= 000152        DBGSND  022764
AMAV    007505        BUFMRG  024702        C$DODU= 000051        C.CRNG= 000132        DBGSUC  023340
AMDU    007521        BUFSIZ  004376        C$DRPT= 000024        C.CTO = 000006        DBGWR   023650
ASSEMB= 000010        CALR3   053762        C$DU  = 000053        C.DR0 = 000016        DEBUG = 000001
ATNACC  040032        CALR4   054006        C$EDIT= 000001        C.DR1 = 000020        DETDUN  010415
ATNAVA  037742        CALR5   054136        C$ERDF= 000055        C.DR2 = 000022        DETHDR  010341
ATNDUP  040002        CALR6   054270        C$ERHR= 000056        C.DR3 = 000024        DFPTBL  002134 G
ATNMSG  011147        CALR7   054424        C$ERRO= 000060        C.FLG = 000014        DIAGMC= 000000
AVAIL   050020        CALR8   054446        C$ERSF= 000054        C.IPR = 000000        DISK  = 000000
BADMEM  024310        CBUFF   003364        C$ERSO= 000057        C.JAD = 000012        DIVIDE  047256
BASLO   007403        CF.ATN= 000200        C$ESCA= 000010        C.JSR = 000010        DIV48   047326
BASL1   007406        CF.CTO= 040000        C$ESEG= 000005        C.MSIZ= 000034        DRIVER  026426
BASL2   007437        CF.ERL= 020000        C$ESUB= 000003        C.NEXT= 000042        DRPCNT  027350
BASN0   007254        CF.FAI= 100000        C$ETST= 000001        C.RBAS= 000100        DRPUNT  027424
BASN1   007300        CF.MSC= 000100        C$EXIT= 000032        C.REND= 000102        DTABS   002176
BASN2   007330        CF.OTH= 000040        C$FREQ= 000101        C.RHDR= 000104        DT.AVL= 100000
BASN3   007357        CF.RPL= 100000        C$FRME= 000100        C.RINT= 000110        DT.BES= 020000
BCHI    004410        CF.SHD= 000002        C$GETB= 000026        C.ROFF= 000036        DT.CLR= 147000
BCLO    004406        CF.TMS= 000020        C$GETW= 000027        C.RPTR= 000076        DT.CHP= 004000
BELL  = 000007 G      CF.576= 000001        C$GMAN= 000043        C.RRNG= 000112        DT.DUN= 040000
BESFIT  024570        CHKSTA  032332        C$GPHR= 000042        C.SIZE= 000170        DT.EOM= 002000
BFTOMM  025276        CLRRSP  026132        C$GPRI= 000040        C.STEP= 000030        DT.ONL= 001000
BIT0  = 000001 G      CMDSIZ= 000051        C$INIT= 000011        C.TBSB= 000166        DT.PRM= 020000
BIT00 = 000001 G      CMPHD   050360        C$INLP= 000020        C.TBSF= 000164        DT.UNT= 000077
BIT01 = 000002 G      CNTINT  052220        C$MANI= 000050        C.TO  = 000154        DT.WRT= 010000
BIT02 = 000004 G      CNTRSD  002354        C$MAP = 000102        C.TOM = 000156        DUP   = 001000
BIT03 = 000010 G      CNTRSP  053252        C$MEM = 000031        C.TPAT= 000040        D.BEC = 000004
BIT04 = 000020 G      CNTSRV  054502 G      C$MMU = 000103        C.TYPE= 000026        D.BES = 000004
BIT05 = 000040 G      CR    = 000015 G      C$MSG = 000023        C.UCNT= 000032        D.BGN1= 000006
BIT06 = 000100 G      CRLF    010300        C$OPNR= 000034        C.UID = 000060        D.BGN2= 000016
BIT07 = 000200 G      CTABS   002200        C$OPNW= 000104        C.UNIT= 000002        D.BGN3= 000026
BIT08 = 000400 G      CTLCHK  027140        C$PNTB= 000014        C.VEC = 000004        D.BGN4= 000036
BIT09 = 001000 G      CTRLRS  002202        C$PNTF= 000017        C.XFHR= 000056        D.BLKS= 000050
BIT1  = 000002 G      CT.AVL= 100000        C$PNTS= 000016        C.XFHW= 000050        D.CCNT= 000060
BIT10 = 002000 G      CT.BRL= 007000        C$PNTX= 000015        C.XFLR= 000052        D.CERR= 000116
BIT11 = 004000 G      CT.DET= 002000        C$PUTB= 000072        C.XFLW= 000044        D.DRV = 000000
BIT12 = 010000 G      CT.DI = 004000        C$PUTW= 000073        C.XFMR= 000054        D.ECCC= 000120
BIT13 = 020000 G      CT.DUN= 040000        C$QIO = 000377        C.XFMW= 000046        D.END1= 000012
BIT14 = 040000 G      CT.IOC= 010000        C$RDBU= 000007        C25MSZ= 000000        D.END2= 000022
BIT15 = 100000 G      CT.MRW= 000400        C$REFG= 000047        DBGACC  023404        D.END3= 000032
BIT2  = 000004 G      CT.MSG= 020000        C$REL = 000077        DBGATN  023005        D.END4= 000042
BIT3  = 000010 G      CT.UNT= 000077        C$RESE= 000033        DBGAVL  023245        D.HERR= 000112
BIT4  = 000020 G      CT.VEC= 000777        C$REVI= 000004        DBGCMD  023704        D.LBN = 000052
BIT5  = 000040 G      C$AU  = 000052        C$RFLA= 000021        DBGCMP  023562        D.SERR= 000114
BIT6  = 000100 G      C$AUTO= 000061        C$RPT = 000025        DBGERS  023440        D.SIZE= 000136
BIT7  = 000200 G      C$BRK = 000022        C$SEFG= 000046        DBGFLG  022762        D.UID = 000122
BIT8  = 000400 G      C$BSEG= 000004        C$SPRI= 000041        DBGGCS  023057        D.UNIT= 000002
BIT9  = 001000 G      C$BSUB= 000002        C$SVEC= 000037        DBGGUS  023127        D.VSN = 000132
BLANK   010221        C$CLCK= 000062        C$TOME= 000076        DBGINV  023033        D.WCNT= 000056
```

```
D.XFHA= 000102      ERR006 017332 G     F$SEG = 000003      INITXX 062554       L$AU   062634 G
D.XFHC= 000110      ERR014 017362 G     F$SOFT= 000005      INIT1  060652       L$AUT  002070 G
D.XFHR= 000074      ERR021 017452 G     F$SRV = 000010      INIT2  061064       L$AUTO 062610 G
D.XFHW= 000066      ERR022 017524 G     F$SUB = 000002      INIT3  061122       L$CCP  002106 G
D.XFLA= 000076      ERR023 017606 G     F$SW  = 000014      INIT4  061140       L$CLEA 062612 G
D.XFLC= 000104      ERR024 017770 G     F$TEST= 000001      INIT5  061506       L$CO   002032 G
D.XFLR= 000070      ERR025 020046 G     F.HDR  004624       INIT6  061706       L$DEPO 002011 G
D.XFLW= 000062      ERR026 020126 G     GCSTAT 050056       INIT7  062062       L$DESC 007162 G
D.XFMA= 000100      ERR027 020210 G     GETBC  045502       INP28A 065011       L$DESP 002076 G
D.XFMC= 000106      ERR030 020246 G     GETBES 046060       INP28B 065054       L$DEVP 002060 G
D.XFMR= 000072      ERR031 020264 G     GETBUF 024354       INP28C 065123       L$DISP 002124 G
D.XFMW= 000064      ERR032 020312 G     GETLBN 045550       INVCMD 027744       L$DLY  002116 G
EF.BBR= 000200      ERR033 020346 G     GETME  025550       IPAT16= 000100      L$DTP  002040 G
EF.BBU= 000100      ERR034 020402 G     GTDRVT 047164       IREST = 000004      L$DTYP 002034 G
EF.CON= 000036 G    ERR035 020454 G     GTNXTD 045526       ISR   = 000100 G    L$DU   062632 G
EF.LOG= 000040      ERR037 020510 G     GUSTAT 050114       ISTRT = 000010      L$DUT  002072 G
EF.NEW= 000035 G    ERR038 020576 G     G$CNTO= 000200      ITEMP = 000040      L$DVTY 007142 G
EF.PWR= 000034 G    ERR039 020714 G     G$DELM= 000372      ITOA   047472       L$EF   002052 G
EF.RES= 000037 G    ERR040 021016 G     G$DISP= 000003      IXE   = 004000 G    L$ENVI 002044 G
EF.SEX= 000020      ERR041 021240 G     G$EXCP= 000400      I$AU  = 000041      L$ERRT 002156 G
EF.STA= 000040 G    ERR042 021462 G     G$HILI= 000002      I$AUTO= 000041      L$ETP  002102 G
EF.XM = 000033 G    ERR043 021626 G     G$LOLI= 000001      I$CLN = 000041      L$EXP1 002046 G
ENDACC 031004       ERR050 021654 G     G$NO  = 000000      I$DU  = 000041      L$EXP4 002064 G
ENDADR 027676       ERR051 021706 G     G$OFFS= 000400      I$HRD = 000041      L$EXP5 002066 G
ENDAVL 030274       ERR052 022004 G     G$OFSI= 000376      I$INIT= 000041      L$HARD 065214 G
ENDCMP 031600       ERR053 022122 G     G$PRMA= 000001      I$MOD = 000041      L$HIME 002120 G
ENDERS 031120       ERR054 022236 G     G$PRMD= 000002      I$MSG = 000041      L$HPCP 002016 G
ENDGCS 027774       ERR060 022472 G     G$PRML= 000000      I$PROT= 000040      L$HPTP 002022 G
ENDGUS 030152       ERR061 022544 G     G$RADA= 000140      I$PWR = 000041      L$HW   002134 G
ENDMRD 031240       ERR21A 017516       G$RADB= 000000      I$RPT = 000041      L$ICP  002104 G
ENDMWR 031510       ERR21E 017522       G$RADD= 000040      I$SEG = 000041      L$INIT 060364 G
ENDNUM 027630       ERR23A 017724       G$RADL= 000120      I$SETU= 000041      L$LADP 002026 G
ENDONL 030364       ERR23B 017754       G$RADO= 000020      I$SFT = 000041      L$LAST 066462 G
ENDRD  031720       EVL   = 000004 G    G$XFER= 000004      I$SRV = 000041      L$LOAD 002100 G
ENDSCC 030200       E$END = 002100      G$YES = 000010      I$SUB = 000041      L$LUN  002074 G
ENDSIZ= 000044      E$LOAD= 000035      HISEED 002362       I$TST = 000041      L$MREV 002050 G
ENDSUC 030714       FFREE  002166       HM.WRT= 010000 G    J$JMP = 000167      L$NAME 002000 G
ENDWR  032166       FMADR  002172       HOE   = 100000 G    KW.BRL 002324       L$PRIO 002042 G
ERASE  050532       FMSIZ  002174       H.DRV = 000002      KW.CSR 002322       L$PROT 060356 G
ERLSTA 022334       FSIZE  002170       H.IPR = 000000      KW.EL  002332       L$PRT  002112 G
ERRBLK 002164 G     F$AU  = 000015      H.PRM = 000004      KW.HZ  002330       L$REPP 002062 G
ERRCNT 040112       F$AUTO= 000020      IBE   = 010000 G    KW.OUT= 000105 G    L$REV  002010 G
ERRDTE 040322       F$BGN = 000040      ICONT = 000002      KW.VEC 002326       L$RPT  054534 G
ERRHMA 040216       F$CLEA= 000007      IDIE  = 040000      KW.60  002336       L$SOFT 065344 G
ERRLIM 010527       F$DU  = 000016      IDU   = 000040 G    KW11I  054512 G     L$SPC  002056 G
ERRMSG 002162 G     F$END = 000041      IDUN  = 010000      LF    = 000012 G    L$SPCP 002020 G
ERRNBR 002160 G     F$HARD= 000004      IER   = 020000 G    LF.CON= 000100      L$SPTP 002024 G
ERRSDE 040572       F$HW  = 000013      IERL  = 020000      LF.SNR= 000001      L$STA  002030 G
ERRSDI 040446       F$INIT= 000006      IFLAGS 002204       LF.SUC= 000200      L$SW   002144 G
ERRTYP 002156 G     F$JMP = 000050      IMMU  = 100000      LOE   = 040000 G    L$TEST 002114 G
ERRVEC= 000004 G    F$MOD = 000000      INITBL 002340       LOSEED 002360       L$TIML 002014 G
ERR002 017056 G     F$MSG = 000011      INITWA 010725       LOT   = 000010 G    L$UNIT 002012 G
ERR003 017124 G     F$PROT= 000021      INITWB 010777       L$ACP  002110 G     L.BADR= 000030
ERR004 017172 G     F$PWR = 000017      INITWC 011044       L$APT  002036 G     L.CHVR= 000025
ERR005 017240 G     F$RPT = 000012      INITWD 010224                           L.CNTI= 000014
```

CZUDJAO UDA50-A/KDA50-Q SUBSY E MACRO V05.03   Wednesday 02-Oct-85 16:03   Page 145-3
Symbol table

L.CRF = 000000	L10046 060354	MM8UF 004400	MSGLOG 016373	OPCGCS 007745
L.CSVR= 000024	L10050 062606	MMOFF 004404	MSGPKT 22616	OPCGUS 007772
L.EVNT= 000012	L10051 062610	MMPKTS 004626	MSGSUC 016205	OPCMRD 010152
L.FLGS= 000011	L10052 062630	MMSIZ 004402	MSHST0 036012	OPCMWR 010175
L.FMT = 000010	L10053 062632	MMTOBF 025154	MSHST1 036046	OPCONL 010014
L.HDCD= 000050	L10054 062634	MNTPAT 007036	MSHST2 036073	OPCRD 010025
L.LVL = 000042	L10055 063210	MNTRD 051112	MSHST3 036112	OPCRPL 010034
L.MLUN= 000026	L10056 063126	MNTWR 051122	MSHST4 036136	OPCSCC 010046
L.RTRY= 000043	L10057 063204	MSAVA0 033760	MSHST5 036167	OPCSUC 010107
L.SCYL= 000042	L10060 063302	MSCMP0 034664	MSINV0 033074	OPCWR 010142
L.SDI = 000054	L10061 065210	MSCNT0 036372	MSMFE0 034154	OP.ABO= 000001
L.SEQ = 000006	L10062 065244	MSCNT1 036416	MSMFE1 034202	OP.ACC= 000020
L.UHVR= 000041	L10063 065506	MSCNT2 036451	MSMFE2 034247	OP.AVA= 000100
L.UNIT= 000004	L10064 066466	MSCNT3 036463	MSMFE3 034263	OP.AVL= 000010
L.UNTI= 000030	L10066 066474	MSCNT4 036521	MSMFE4 034277	OP.CCD= 000021
L.USVR= 000040	MAINT = 177760	MSCNT5 036544	MSOFL0 033440	OP.CMP= 000040
L.VSER= 000044	MAINTP 051130	MSCNT6 036611	MSOFL1 033460	OP.DAP= 000013
L10000 002142	MAPMEM 025006	MSCNT7 036657	MSOFL2 033524	OP.DUP= 000101
L10001 002156	MD.ALL= 000002	MSCNT8 036715	MSOFL3 033545	OP.END= 000200
L10002 017122	MD.CMP= 040000	MSCNT9 036754	MSOFL4 033604	OP.ERS= 000022
L10003 017170	MD.CSE= 020000	MSCPOP 002314	MSOFL5 033632	OP.FLU= 000023
L10004 017236	MD.CWB= 000010	MSDATA 035516	MSSUC0 032632	OP.GCS= 000002
L10005 017330	MD.ERR= 010000	MSDATB 035542	MSSUC1 032645	OP.GUS= 000003
L10006 017360	MD.EXP= 100000	MSDATC 035566	MSSUC2 032667	OP.MRD= 000030
L10007 017450	MD.FEU= 000001	MSDATG 035612	MSSUC3 032707	OP.MWR= 000031
L10010 017522	MD.IMF= 000002	MSDATZ 035230	MSSUC4 032735	OP.ONL= 000011
L10011 017604	MD.NXU= 000001	MSDAT0 035212	MSSUC5 032754	OP.RD = 000041
L10012 017766	MD.PRI= 000001	MSDAT1 035260	MSSUC6 032771	OP.RPL= 000024
L10013 020044	MD.RIP= 000001	MSDAT2 035277	MSUNK0 033164	OP.SCC= 000004
L10014 020124	MD.SCH= 004000	MSDAT4 035323	MSWPR0 034434	OP.SEX= 000007
L10015 020206	MD.SCL= 002000	MSDAT5 035353	MSWPR1 034457	OP.SHC= 000102
L10016 020244	MD.SEC= 000100	MSDAT6 035376	MSWPR2 034520	OP.SUC= 000012
L10017 020262	MD.SEQ= 000020	MSDAT7 035422	MULT 047420	OP.WR = 000042
L10020 020310	MD.SER= 000400	MSDAT8 035446	MUTOFF 004412	O$APTS= 000001
L10021 020344	MD.SHD= 000020	MSDAT9 035472	MUTSIZ 004516	O$AU = 000001
L10022 020400	MD.SPD= 000001	MSDRVA 037562	M.HDR 004622	O$BGNR= 000001
L10023 020452	MD.SSH= 000200	MSDRVB 037616	NOCLOC 011071	O$BGNS= 000001
L10024 020506	MD.SWP= 000004	MSDRVC 037653	NOMEM 062574	O$DU = 000001
L10025 020574	MD.VOL= 000002	MSDRV0 037202	NUMCNT 036326	O$ERRT= 000001
L10026 020712	MD.WBN= 000100	MSDRV1 037221	NUMDAT 035126	O$GNSW= 000001
L10027 021014	MD.WBV= 000040	MSDRV2 037250	NUMDRV 037122	O$POIN= 000001
L10030 021236	MEMIN 006232	MSDRV3 037317	NXMAD 002264	O$SETU= 000001
L10031 021460	ME.CBL= 000002	MSDRV4 037340	NXMI 054472 G	PATBLD 046474
L10032 021624	ME.CFG= 000016	MSDRV5 037366	OBUFF 002364	PATCHK 046100
L10033 021652	ME.CFL= 000000	MSDRV6 037412	OFF = 000000	PATDSP 006776
L10034 021704	ME.CL = 000000	MSDRV7 037436	ON = 000001	PAT00 006246
L10035 022002	ME.CP = 000026	MSDRV8 037463	ONLINE 050150	PAT01 006312
L10036 022120	ME.CRF= 000004	MSDRV9 037517	OPCABO 007562	PAT02 006320
L10037 022234	ME.CTH= 000010	MSGBBR 016267	OPCACC 007572	PAT03 006326
L10040 022332	ME.CTO= 000006	MSGBBU 016326	OPCAVL 007603	PAT04 006334
L10041 022542	ME.CV = 000024	MSGCON 016236	OPCCCD 007617	PAT05 006400
L10042 022614	ME.OFF= 000020	MSGCST 065306	OPCCMP 007651	PAT06 006444
L10043 054500	ME.PKT= 000022	MSGFAI 016160	OPCDAP 007675	PAT07 006510
L10044 054510	ME.ST1= 000012	MSGIPR 065244	OPCERS 007725	PAT08 006516
L10045 054532	ME.ST2= 000014	MSGLDR 065276	OPCFLU 007735	PAT09 006562

```
PAT10    006570       P.UNFL= 000016       RSPEND  027554       SC.DII= 000600       SM.EER= 000040 G
PAT11    006634       P.UNIT= 000004       RSPERL  040036       SC.DIS= 000400       SM.MAN= 000001 G
PAT12    006642       P.UNSZ= 000044       RSPIN   027500       SC.DSY= 000140       SM.PCK= 000004 G
PAT13    006706       P.UNTI= 000024       RSPSIZ= 000103       SC.DUP= 000200       SM.PSE= 000002 G
PAT14    006752       P.USVR= 000052       RSPSTR  006230       SC.EDC= 000100       SM.RAN= 000200 G
PAT15    006764       P.VRSN= 000014       RSP.ER  053226       SC.FCT= 000300       SM.VLT= 000020 G
PCBAD    004366       P.VSER= 000050       RSP.SU  053244       SC.FOR= 000000       SNDMSG  051504
PCGOOD   004364       QDXMSZ= 000000       RSP.S1  053122       SC.HDR= 000100       SND.S1  002342
PKTSIZ=  000006       Q50MSZ= 032413       RSP.S2  053160       SC.HMP= 000200       SND.S2  002346
PNT   =  001000 G     RANDOM  026250       RSP.S3  053204       SC.HWP= 020000       SO.BIT= 000010
PRI   =  002000 G     READ    050556       RTNALL  025606       SC.IEE= 000200       SO.EL = 000000
PRI00 =  000000 G     REINIT  053420       RTNME   026006       SC.IIC= 000140       SO.MTS= 000006
PRI01 =  000040 G     RESET   053452       SA.BST= 000374       SC.IOP= 000100       SO.SRI= 000004
PRI02 =  000100 G     RETBUF  025420       SA.CMD= 034000       SC.IPT= 000240       SO.TL = 000002
PRI03 =  000140 G     RE.RFL= 000000       SA.CME= 000070       SC.LPI= 000240       SSTEP4  002352
PRI04 =  000200 G     RE.RL = 000002       SA.CM1= 004000       SC.LPO= 000300       STAABO  033220
PRI05 =  000240 G     RE.RP = 000006       SA.CNT= 000360       SC.LRR= 000300       STAADR  032422
PRI06 =  000300 G     RE.RV = 000004       SA.CTP= 003400       SC.LRW= 000200       STAAVA  033700
PRI07 =  000340 G     RG.FLG= 040000       SA.DI = 000400       SC.NVL= 000040       STACMP  034562
P.ADBA=  000020       RG.OWN= 100000       SA.ERC= 003777       SC.NXM= 000140       STACNT  036220
P.ADEA=  000022       RNGLEN= 000004       SA.ERR= 100000       SC.OBC= 000100       STACOD  032372
P.BCNT=  000014       RNGPWR= 011000       SA.GO = 000001       SC.OTA= 000040       STADAT  034702
P.BUFF=  000020       RNTHH   002266       SA.INE= 000200       SC.POE= 000140       STADRV  037004
P.CHVR=  000023       RNTIM   010303       SA.INT= 000200       SC.PRO= 000500       STAHST  035630
P.CMST=  000020       RNTIME  053550       SA.LFC= 000002       SC.PSE= 000400       STAINV  033024
P.CNTF=  000016       RNTHM   002270       SA.MCV= 000017       SC.RBN= 000440       STAMFE  034000
P.CNTI=  000024       RNTSS   002272       SA.MP = 000100       SC.RCT= 000400       STAMSO  002316
P.CPSP=  000042       RPTAF   002306       SA.MSE= 000007       SC.ROV= 010000       STAMS1  002320
P.CRF =  000000       RPTAV   002304       SA.MSG= 003400       SC.RRC= 000640       STAOFL  033250
P.CSVR=  000022       RPTB    060001       SA.MS1= 000400       SC.SOS= 000040       STASIZ= 000026
P.CTMO=  000020       RPTCF   002312       SA.NV = 002000       SC.SQL= 001000       STASUC  032452
P.CYLS=  000050       RPTCV   002310       SA.NVE= 000400       SC.SPI= 000040       STAUNK  033114
P.DVPM=  000034       RPTDED  060253       SA.PRG= 000001       SC.STC= 000100       STAWPR  034320
P.FBBK=  000034       RPTDES  056750       SA.QB = 001000       SC.SWP= 010000       STIME   002250
P.FLGS=  000011       RPTDE1  060007       SA.SM = 000040       SC.UEC= 000340       ST.ABO= 000002
P.GRPS=  000046       RPTDE2  060042       SA.STE= 000200       SC.UNK= 000000       ST.AVL= 000004
P.HTMO=  000020       RPTDE3  060145       SA.STP= 100000       SC.XME= 000100       ST.CMD= 000001
P.LBN =  000034       RPTHDR  057300       SA.S1 = 004000       SC.1EC= 000400       ST.CMP= 000007
P.MEDI=  000034       RPTIOC  057623       SA.S2 = 010000       SC.2EC= 000440       ST.CNT= 000012
P.MLUN=  000014       RPTIOD  057674       SA.S3 = 020000       SC.3EC= 000500       ST.DAT= 000010
P.MOD =  000012       RPTIOE  057743       SA.S4 = 040000       SC.4EC= 000540       ST.DIA= 000057
P.OPCD=  000010       RPTIO1  057333       SA.TST= 100000       SC.5EC= 000600       ST.DRV= 000013
P.QTRF=  000014       RPTIO2  057370       SA.VCE= 000177       SC.576= 000240       ST.HST= 000011
P.RBN =  000014       RPTIO3  057505       SA.VEC= 000177       SC.6EC= 000640       ST.MFE= 000005
P.RBNS=  000056       RPTKB   060003       SA.WMP= 040000       SC.7EC= 000700       ST.MSK= 000037
P.RCTC=  000057       RPTLUN  002274       SCC     050210       SC.8EC= 000740       ST.OFL= 000003
P.RCTS=  000054       RPTMB   060005       SC.AQL= 000400       SETMIN  052144       ST.SUB= 000040
P.RGID=  000034       RPTRF   002302       SC.CEC= 000200       SETTO   052070       ST.SUC= 000000
P.RGOF=  000040       RPTRV   002300       SC.CIP= 000340       SFPTBL  002144 G     ST.WPR= 000006
P.SHST=  000042       RPTSTA  060327       SC.CME= 000440       SIZCNT= 000020       SUCHAR  050246
P.SHUN=  000040       RPTSTB  060341       SC.COU= 000400       SIZDAT= 000030       SVCGBL= 000000
P.STS =  000012       RPTWF   002276       SC.CTO= 000040       SIZDRV= 000026       SVCINS= 000000
P.TIME=  000024       RPTWV   002274       SC.DCD= 000240       SM.CMP= 000010 G     SVCSUB= 000000
P.TRKS=  000044       RSPATN  037704       SC.DDE= 000340       SM.DET= 000400 G     SVCTAG= 000000
P.UHVR=  000053       RSPCNT  006226       SC.DFI= 000550       SM.ECC= 000100 G     SVCTST= 000000
```

Symbol	Value		Symbol	Value		Symbol	Value		Symbol	Value		Symbol	Value
S$LSYM=	010000		T$SIZE=	000005		T2S1	041666		T3S4E	044044		X22B	012273
S.CMP	066053		T$SUBN=	000000		T2S2	041720		T3S5	044140		X23A	012325
S.DET	066326		T$TAGL=	177777		T2S2E	042010		T3S5E	044274		X23C	012441
S.ECC	066245		T$TAGN=	010067		T2S2N	041764		T3S6	044524		X23D	012530
S.EER	066220		T$TEMP=	000005		T2S3	042050		T3S6E	045310		X23E	012602
S.EL	065626		T$TEST=	000003		T2S4	042122		T3S7	045410		X23F	012636
S.HDR	065506		T$TSTM=	177777		T2S4E	042204		T3TED	064360		X24A	012655
S.MAN	065567		T$TSTS=	000001		T2S4N	042160		T3WARN	010625		X24B	012762
S.MTS	066160		T$$AU=	010054		T2S5	042216		T3WRD	064612		X25	013034
S.PCK	066005		T$$AUT=	010051		T2S6	042306		T3W1	043166		X26	013135
S.RAN	066300		T$$CLE=	010052		T2S6E	042402		T3W2	043634		X27	013215
S.SRI	065707		T$$DAT=	010066		T2S6N	042364		T3W4	044352		X3	011255
S.SSF	065753		T$$DU=	010053		T2S6R	042412		T3W5	045366		X31	013317
S.TL	065647		T$$HAR=	010062		T2S7	042440		UAM =	000200 G		X32	013415
S.VLT	066122		T$$HW =	010000		T2S7E	042554		UF.CMR=	000001		X33	013526
TAPE =	000400		T$$INI=	010050		T2S7N	042536		UF.CMW=	000002		X34	013635
TBSMIN	006234		T$$MSG=	010042		T2S7R	042564		UF.INA=	040000		X35	013754
TBSSIZ	006236		T$$PC =	000001		T2S8	042612		UF.RMV=	000200		X37A	014102
TBSTRT	006240		T$$PRO=	010047		T2S8E	042664		UF.RPL=	100000		X37B	014167
TEMP	002212		T$$PTA=	010065		T2S8R	042702		UF.SCH=	004000		X37C	014267
TIMLIM	010453		T$$RPT=	010046		T2W1	042026		UF.SCL=	002000		X38A	014343
TINDEX=	000012		T$$SOF=	010063		T2W2	042264		UF.WBN=	000100		X38B	014433
TNAMES	007244		T$$SRV=	010045		T2W3	042416		UF.WPH=	020000		X38C	014474
TNUM	002206		T$$SUB=	010057		T2W4	042570		UF.WPS=	010000		X39A	014545
TSPGEN	002246		T$$SW =	010001		T3	063304 G		UF.576=	000004		X39B	014632
TSTIM	002254		T$$TES=	010061		T3BEG	064452		UTEST	002210		X39C	014675
TSTOFF	002242		T.HDR	006242		T3BES	064412		U5OMSZ=	036413		X4A	011340
TSTOPC	002244		T1	062636 G		T3DSP	042706		VECSAV	002262		X4B	011421
TS.PAT	002260		T1DSP	040716		T3END	064467		WCHNG	053652		X41	014744
T$ARGC=	000001		T1LOE	041164		T3L1E	043162		WCHNGD	002356		X42A	015001
T$CODE=	0C4130		T1L1E	041474		T3L2E	043630		WRITE	050646		X42B	015077
T$ERRN=	000032		T1L2E	041560		T3L3E	044134		XBHW	016457		X42C	015145
T$EXCP=	000000		T1NEXT	062664		T3L4E	044346		XFCE	016543		X43	015154
T$FLAG=	000040		T1PGEN	040734		T3L5E	045362		XFERPK	050744		X5A	011504
T$FREE=	066474		T1SKIP	063116		T3L6E	045476		XFRU	016743		X5B	011550
T$GMAN=	000000		T1S0	040764		T3NWD	064555		XFRUA	017013		X5C	011626
T$HILI=	000200		T1S1	041052		T3PAT	064502		XMSG1	016612		X50A	015253
T$LAST=	000001		T1S2	041212		T3PGEN	042736		XMSG2	016647		X51A	015325
T$LOLI=	000001		T1S3	041300		T3P16	064526		XSA	016705		X51B	015407
T$LSYM=	010000		T1S4	041500		T3QHDR	064625		XSAE	016656		X52A	015443
T$LTNO=	000003		T1W1	041170		T3QHED	064701		XSTA	016423		X52B	015520
T$NEST=	177777		T1W2	041416		T3QUES	063520		X$ALWA=	000000		X53A	015573
T$NS0 =	000005		T1.1	062652		T3S0	042766		X$FALS=	000040		X53B	015636
T$NS1 =	000002		T1.2	063130		T3S1	043026		X$OFFS=	000400		X54A	015731
T$PCNT=	000000		T2	063212 G		T3S2	043060		X$TRUE=	000020		X54B	016003
T$PTAB=	010065		T2DSP	041564		T3S2E	043150		X14A	011750		X6	011652
T$PTHV=	000001		T2L1E	042022		T3S2N	043124		X14B	012044		X60	016024
T$PTNU=	000001		T2L2E	042260		T3S3	043350		X14C	012126		X61	016077
T$SAVL=	177777		T2PGEN	041616		T3S3E	043534		X2	011711		$PATCH	066356 G
T$SEGL=	177777		T2S0	041646		T3S4	043656		X22A	012175			

```
. ABS.   066474     000    (RW,I,GBL,ABS,OVR)
         000000     001    (RW,I,LCL,REL,CON)
Errors detected:   0
```

*** Assembler statistics

Work file reads: 407
Work file writes: 396
Size of work file: 37386 Words (147 Pages)
Size of core pool: 19372 Words (74 Pages)
Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:10:16.25
ZUDJAO.BIC,ZUDJAO/CR/-SP=SVC41R.MLB/ML,ZUDJAO.MEM,ZUDJAO.MAC