RD51/52 RX50 RQDX1/RUX50 EXERCISER
RQDX1 RUX50 CZRQAE0

COPYRIGHT (c) 1983-84
AH-T399E-MC
FICHE 01 OF 03

OCT 1984
digital
Made In USA

RD51/52 RX50 RQDX1/RUX50 EXERCISER
RQDX1 RUX50 CZRQAE0

COPYRIGHT (c) 1983-84
AH-T399E-MC
FICHE 02 OF 03

OCT 1984
digital
Made In USA

RD51/52 RX50 RQDX1/RUX50 EXERCISER
RQDX1 RUX50 CZRQAE0

COPYRIGHT (c) 1983-84
AH-T399E-MC
FICHE 03 OF 03

OCT 1984
digital
Made In USA

```
;    0001  0     module ZRQAM1 (
;    0002  0
;    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                    ident = 'V01.8',
;    0005  0                    addressing_mode (absolute),
;    0006  0                    environment (noeis)
;    0007  0                    ) =
;    0008  0
;    0009  1     begin
;    0010  1
;    0011  1
; C  0012  1     %(
; C  0013  1                                IDENTIFICATION
; C  0014  1                                --------------
; C  0015  1
; C  0016  1             PRODUCT CODE:      AC-T398E-MC
; C  0017  1
; C  0018  1             PRODUCT NAME:      CZRQAE0 RQDX1/RUX50 EXERCISER
; C  0019  1
; C  0020  1             PRODUCT DATE:      6-JUL-84
; C  0021  1
; C  0022  1             MAINTAINER:        DIAGNOSTIC ENGINEERING
; C  0023  1
; C  0024  1             AUTHOR:            RAVINDER K. KARWAN
; C  0025  1                                BOB POWERS
; C  0026  1
; C  0027  1
; C  0028  1             Copyright (C) 1983, 1984
; C  0029  1
; C  0030  1             Digital Equipment Corporation, Maynard, Massachusetts  01754
; C  0031  1
; C  0032  1             This software is furnished  under a license for use only on a single
; C  0033  1             computer  system and  may be  copied only with  the inclusion of the
; C  0034  1             above copyright notice.  This software, or any other copies thereof,
; C  0035  1             may not be provided or  otherwise made available to any other person
; C  0036  1             except for use on such system and to one who agrees to these license
; C  0037  1             terms.  Title to and  ownership of the  software  shall at all times
; C  0038  1             remain in DEC.
; C  0039  1
; C  0040  1             the information in this document is subject to change without notice
; C  0041  1             and should  not be construed  as a commitment  by Digital  Equipment
; C  0042  1             Corporation.
; C  0043  1
; C  0044  1             DEC assumes  no  responsibility  for  the use or  reliability of its
; C  0045  1             software on equipment which is not supplied by DEC.
; C  0046  1
; C  0047  1
; C  0048  1             The following are trademarks of Digital Equipment Corporation:
; C  0049  1
; C  0050  1             DIGITAL          PDP              UNIBUS           MASSBUS
; C  0051  1             DEC              DECUS            DECTAPE
```

```
;  C 0052  1
;  C 0053  1
;  C 0054  1               REVISION HISTORY:
;  C 0055  1
;  C 0056  1               REV 1.6    11-APR-84    MERGED FIELD AND MANUFACTURING VERSIONS OF THE RD/RX EXERCISER.
;  C 0057  1                                       ADDED SUPPORT FOR THE RUX50.
;  C 0058  1
;  C 0059  1               REV 1.7    01-MAY-84    ADDED CODE TO GET DEVICE TYPE FROM CONTROLLER CHARACTERISTICS;
;  C 0060  1                                       ADDED APT BREAKS IN UNIT_INIT ROUTINE; CORRECTED SOFT SEEK ERROR
;  C 0061  1                                       TOTALS; PROTECT MEDIA ON DEFAULT.
;  C 0062  1
;  C 0063  1               REV 1.8    6-JUL-84     ELIMINATE GETTING DISK TYPE FROM ID BLOCK ON A RESTART;
;  C 0064  1
;  C 0065  1
;  C 0066  1
;  C 0067  1
```

TABLE OF CONTENTS

1.0 GENERAL INFORMATION
1.1 PROGRAM ABSTRACT
1.2 SYSTEM REQUIREMENTS
1.2.1 HARDWARE REQUIREMENTS
1.2.2 SOFTWARE REQUIREMENTS
1.3 RELATED DOCUMENTS AND STANDARDS
1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
1.5 ASSUMPTIONS
1.6 MEMORY MAP

2.0 OPERATING INSTRUCTIONS
2.1 HARDWARE QUESTIONS
2.2 SOFTWARE QUESTIONS

3.0 ERROR TYPES
3.1 ERROR INFORMATION
3.2 INITIALIZATION ERRORS
3.3 EXERCISER ERRORS
3.4 ERROR LOG MESSAGES
3.5 MSCP ERRORS
3.6 SAMPLE ERROR STATEMENT

4.0 PERFORMANCE AND PROGRESS REPORTS

5.0 TEST SUMMARY
5.1 INITIALIZATION SUBTEST
5.2 EXERCISER
5.3 DROP UNIT SUMMARY

6.0 ERROR LIST

7.0 DATA PATTERNS

```
; C 0105  1
; C 0106  1
; C 0107  1
; C 0108  1
; C 0109  1
; C 0110  1
; C 0111  1
; C 0112  1
; C 0113  1
; C 0114  1
; C 0115  1
; C 0116  1
; C 0117  1
; C 0118  1
; C 0119  1
; C 0120  1
; C 0121  1
; C 0122  1
; C 0123  1
; C 0124  1
; C 0125  1
; C 0126  1
; C 0127  1
; C 0128  1
; C 0129  1
; C 0130  1
; C 0131  1
; C 0132  1
; C 0133  1
; C 0134  1
; C 0135  1
; C 0136  1
; C 0137  1
; C 0138  1
; C 0139  1
; C 0140  1
; C 0141  1
; C 0142  1
; C 0143  1
; C 0144  1
; C 0145  1
; C 0146  1
; C 0147  1
; C 0148  1
; C 0149  1
; C 0150  1
; C 0151  1
; C 0152  1
; C 0153  1
; C 0154  1
; C 0155  1
```

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

This program will functionally verify and exercise RQDX1 or RUX50 Controller/Disk Drive subsystems. It is designed to verify that the subsystem is functioning correctly and operating within design specifications.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE REQUIREMENTS

LSI - 11/23 processor with 28K or more of memory, console device (eg. VT100) and RQDX1 or RUX50 controller board and attached RD51 or RD52 WINCHESTER drive(s) and RX-50 FLOPPY drive(s)

1.2.2 SOFTWARE REQUIREMENTS

This diagnostic is designed to run with the Diagnostic Supervisor as described in paragraph 2.0.

1.3 RELATED DOCUMENTS AND STANDARDS

XXDP+ SUPERVISOR/USERS MANUAL CHQUS
UQSSP UNIBUS/Q-BUS STORAGE SYSTEMS PORT
MSCP MASS STORAGE SYSTEM PROTOCOL

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

NONE

1.5 ASSUMPTIONS

The hardware, other than the subsystem being tested, is assumed to work properly. False errors may be reported if the processor, memory, etc., do not function properly.

```
; C 0156  1          1.6     MEMORY MAP
; C 0157  1                  ----------
; C 0158  1
; C 0159  1
; C 0160  1                  Memory layout on 28k machine - XXDP environment
; C 0161  1                  ------------------------------------------------
; C 0162  1
; C 0163  1                                                  ADDRESS
; C 0164  1
; C 0165  1                  +-----------------------+    160000
; C 0166  1                  !                       !
; C 0167  1                  !     XXDP+ MONITOR     !
; C 0168  1                  !                       !
; C 0169  1                  +-----------------------+
; C 0170  1                  !                       !
; C 0171  1                  !  XXDP+   TTY I/O AREA !
; C 0172  1                  !                       !
; C 0173  1                  +-----------------------+
; C 0174  1                  !                       !
; C 0175  1                  !                       !
; C 0176  1                  !         DRS           !
; C 0177  1                  !                       !
; C 0178  1                  !                       !
; C 0179  1                  +-----------------------+    121000
; C 0180  1                  !                       !
; C 0181  1                  !      EXERCISER        !
; C 0182  1                  !                       !
; C 0183  1                  +-----------------------+    2000
; C 0184  1                  !                       !
; C 0185  1                  !        STACKS         !
; C 0186  1                  !                       !
; C 0187  1                  +-----------------------+    400
; C 0188  1                  !                       !
; C 0189  1                  !      VECTOR AREA      !
; C 0190  1                  !                       !
; C 0191  1                  +-----------------------+    0
; C 0192  1
; C 0193  1
; C 0194  1                  In a machine  with more memory,  free space will occur
; C 0195  1                  between the exerciser and the DRS.
```

; C 0196 1
; C 0197 1
; C 0198 1
; C 0199 1
; C 0200 1
; C 0201 1
; C 0202 1
; C 0203 1
; C 0204 1
; C 0205 1
; C 0206 1
; C 0207 1
; C 0208 1
; C 0209 1
; C 0210 1
; C 0211 1
; C 0212 1
; C 0213 1
; C 0214 1
; C 0215 1
; C 0216 1
; C 0217 1
; C 0218 1
; C 0219 1
; C 0220 1
; C 0221 1
; C 0222 1
; C 0223 1
; C 0224 1
; C 0225 1
; C 0226 1
; C 0227 1
; C 0228 1
; C 0229 1
; C 0230 1
; C 0231 1
; C 0232 1
; C 0233 1
; C 0234 1
; C 0235 1
; C 0236 1
; C 0237 1
; C 0238 1
; C 0239 1
; C 0240 1
; C 0241 1
; C 0242 1
; C 0243 1
; C 0244 1

2.0 OPERATING INSTRUCTIONS

This is a Rev C Supervisor Diagnostic: for operating instructions, please see chapter 5 of XXDP+ operator's manual. They are no longer included in the diagnostic because it is desired that a change in those instructions not require a re-assembly of all Supervisor Diagnostics.

2.1 HARDWARE QUESTIONS

The following series of questions collect the parameters necessary to identify each disk subsystem.

Hardware Configuration Questions

The program will ask the following questions in response to a START command (non-script).

1. CHANGE HW (L) Y ?

   Answer NO to use the pre-built answers for all hardware questions. This program will be released pre-built to test three units with default answers shown below. The pre-built answers may be changed at any time with the setup utility. Answer YES if you want all the hardware questions to be asked.

2. NUMBER OF UNITS (D) ?

   No default. Answer with the number of disk drive units to be exercised or tested. This answer will determine how many times the following questions are asked. A range of 1 to 4 units may be specified. A unit number will be assigned sequentially from 0 by the Diagnostic supervisor for each unit.

3. IP ADDRESS (O) 172150 ?

   Enter the address of the IP register of one RQDX1 or RUX50 as addressed by the processor with memory management turned off. The program expects an even 16-bit address in the range of 160000 to 177774. 172150 is the default.

```
; C 0245  1
; C 0246  1
; C 0247  1
; C 0248  1
; C 0249  1
; C 0250  1
; C 0251  1
; C 0252  1
; C 0253  1
; C 0254  1
; C 0255  1
; C 0256  1
; C 0257  1
; C 0258  1
; C 0259  1
; C 0260  1
; C 0261  1
; C 0262  1
; C 0263  1
; C 0264  1
; C 0265  1
; C 0266  1
; C 0267  1
; C 0268  1
; C 0269  1
; C 0270  1
; C 0271  1
; C 0272  1
; C 0273  1
; C 0274  1
; C 0275  1
; C 0276  1
; C 0277  1
; C 0278  1
; C 0279  1
; C 0280  1
; C 0281  1
; C 0282  1
; C 0283  1
; C 0284  1
; C 0285  1
; C 0286  1
; C 0287  1
; C 0288  1
; C 0289  1
; C 0290  1
; C 0291  1
; C 0292  1
; C 0293  1
; C 0294  1
; C 0295  1
; C 0296  1
; C 0297  1
```

4. VECTOR ADDRESS (O) 154 ?

   Answer with the interrupt vector ofthe same RQDX1 or RUX50 controller described in the above question. A vector address in the range of 4 to 774 may be specified. 154 is the default.

5. BR LEVEL [USUALLY 4-RQDX1 5-RUX50] (D) 4 ?

   Answer with the bus request interrupt level used by the above controller. Levels 4 through 7 are acceptable. 4 is the default.

6. DRIVE NUMBER (D) 0 ?

   Enter the logical unit number for one drive associated with the IP address above. Drive numbers are in the range of 0 through 15. The number entered here must match the unit plug on the front panel of the drive, and must be within the range implied by the jumper (LUN0-7) on the RQDX1 or RUX50 controller board. 0 is the default answer.

7. TEST ENTIRE CUSTOMER DATA AREA OF THIS DISK (L) Y?

   This question is asked to give the opportunity of limiting the addressing range over which the testing will be performed. An affirmative answer will cause no limits to be imposed for the unit in question. A negative answer will cause limits to be imposed, as defined by the following four questions.

8. LOWER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?

   Enter in octal the less significant 16-bit word of the lowest LBN address in the test range. The value may be from 000000 to 177777.

9. HIGHER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?

   Enter in octal the more significant 16-bit word of the lowest LBN address in the test range.

10. LOWER OCTAL WORD OF ENDING LBN ADDRESS (O) 150477?

    Enter in octal the less significant 16-bit word of the highest LBN address in the test range. 150477 is the highest LBN address for an RD52.

```
; C 0298  1
; C 0299  1
; C 0300  1
; C 0301  1
; C 0302  1
; C 0303  1
; C 0304  1
; C 0305  1
; C 0306  1
; C 0307  1
; C 0308  1
; C 0309  1
; C 0310  1
; C 0311  1
; C 0312  1
; C 0313  1
; C 0314  1
```

11. HIGHER OCTAL WORD OF ENDING LBN ADDRESS (0) 0?

Enter in octal the more significant 16-bit word of the highest LBN address in the test range.

Note:
The four previous questions are usually software Parameter questions, but since three diifferent disk drives exist on the subsystem, this becomes a unit by unit question. It is possible to specify an LBN which is too large since we are dealing with different drives. The program will check for block number bounds, and, if they are exceeded, will assign the maximum bounds for that drive.

```
; C 0315  1
; C 0316  1
; C 0317  1
; C 0318  1
; C 0319  1
; C 0320  1
; C 0321  1
; C 0322  1
; C 0323  1
; C 0324  1
; C 0325  1
; C 0326  1
```

12. WRITE ON CUSTOMER DATA AREA ON THIS DISK UNIT (L) ?

Answering YES will destroy any customer data that is on the disk; therefore, the following warning message will appear, followed by a confirmation prompt:

** WARNING - CUSTOMER DATA AREA WILL BE OVERWRITTEN! ...
CONFIRM (L) ?

This question will default to NO if the operator has decided to bypass the hardware questions. Otherwise, there is no default.

```
; C 0327  1
; C 0328  1
; C 0329  1
; C 0330  1
; C 0331  1
; C 0332  1
; C 0333  1
; C 0334  1
; C 0335  1
; C 0336  1
; C 0337  1
; C 0338  1
; C 0339  1
; C 0340  1
; C 0341  1
; C 0342  1
; C 0343  1
; C 0344  1
; C 0345  1
; C 0346  1
; C 0347  1
; C 0348  1
; C 0349  1
; C 0350  1
; C 0351  1
; C 0352  1
; C 0353  1
; C 0354  1
; C 0355  1
; C 0356  1
; C 0357  1
; C 0358  1
; C 0359  1
; C 0360  1
; C 0361  1
; C 0362  1
; C 0363  1
; C 0364  1
; C 0365  1
; C 0366  1
; C 0367  1
; C 0368  1
; C 0369  1
; C 0370  1
; C 0371  1
; C 0372  1
```

2.2 SOFTWARE QUESTIONS

Software Parameter Questions

The program will ask the following questions in response to the START, RESTART, and CONTINUE commands.

1. CHANGE SW (L) Y ?

   Answer NO to bypass the following questions in this section. This question should normally be answered NO when the Exerciser is first run. A NO answer will cause the Exerciser to select the default parameters shown with each question below. Then, depending on the errors detected, it may be desirable to change this answer to YES to alter the test parameters and further isolate the problem.

2. ENTER TIME AS HHMM (EXAMPLE: 1305) (D) 0 ?

   Enter the time of day (in 24 hour format). DRS does not like to see leading zeros in numeric values entered. For example, for 14 minutes past midnight, you would enter 14, and for 30 minutes past 3 in the afternoon, enter 1530.

3. ERROR LIMIT (D) 32 ?

   Enter the number of hard errors allowed before a unit is dropped from testing. A number in the range of 1 to 65535 will be accepted.

4. TRANSFER LIMIT IN MEGABYTES (0 FOR QUICK PASS) (D) 0 ?

   When the specified number of bytes have been transferred to/from a unit, the unit will be dropped from testing. When all units are dropped, an end-of-pass will be indicated. This is the method used to determine how long the Exerciser is to run.

   The only other way the Exerciser will declare end-of-pass is if all units are dropped because the error limit on each is exceeded. However, the operator can always abort the program at any time by typing CONTROL-C.

```
; C 0373  1
; C 0374  1
; C 0375  1
; C 0376  1
; C 0377  1
; C 0378  1
; C 0379  1
; C 0380  1
; C 0381  1
; C 0382  1
; C 0383  1
; C 0384  1
; C 0385  1
; C 0386  1
; C 0387  1
; C 0388  1
; C 0389  1
; C 0390  1
; C 0391  1
; C 0392  1
; C 0393  1
; C 0394  1
; C 0395  1
; C 0396  1
; C 0397  1
; C 0398  1
; C 0399  1
; C 0400  1
; C 0401  1
; C 0402  1
; C 0403  1
; C 0404  1
; C 0405  1
; C 0406  1
; C 0407  1
; C 0408  1
; C 0409  1
; C 0410  1
; C 0411  1
; C 0412  1
; C 0413  1
; C 0414  1
; C 0415  1
; C 0416  1
; C 0417  1
; C 0418  1
; C 0419  1
; C 0420  1
; C 0421  1
; C 0422  1
; C 0423  1
```

5. PERCENTAGE OF 'FIXED DISK' OPERATIONS OUT OF TOTAL OPERATIONS (D) 99 ?

In order to maintain typical usage for the devices of this exercise, a certain percentage of operations must be directed to the RD51/52s (the rest go to the RX50s). It turns out that this percentage is very high (as indicated by the 99% figure given as the default). It may be desirable in some cases to direct more activity to the RX50s. This is easily done by directing a smaller percentage of the operations to the RD51/52s. The numbers associated with usage are adjusted internally by the program according to drive type and percentage.

6. CLEAR STATISTICAL TABLES AFTER PRINTING (L) N ?

Answering YES causes the statistical fields to be cleared to zero after the report is printed (either at end of pass, or at operator request). Otherwise, cumulative totals are maintained.

7. REWRITE BLOCKS WHEN "FORCED ERROR" DETECTED ON READS (L) Y ?

On encountering a bad block on the RD51 or RD52 disk (during either a read or a write operation), the RQDX1 or RUX50 controller will revector the logical block to another physical location on the disk. This operation is transparent to the user. However, if the revectoring was done subsequent to a write operation (i.e the write operation detected the bad block), the data is flagged with a "Forced Error" code, signifying that the data at the revectored location is suspect. The controller returns an error code whenever the block is re-read. Answer 'Yes' to the question to force a WRITE operation on the same block whenever a "Forced Error" flag is detected on a read. This is to avoid the same error code (the "Forced Error") being reported for the same block repeatedly. The re-write will, however, take place only if writes are enabled for the particular disk unit.

8. HALT ON BAD-BLOCK HARD ERRORS (#s 35, 38) (L) Y ?

When the Exerciser is run with the DRS "Halt on Error" switch set (eg. START/FLAGS:HOE), the Exerciser halts on on encountering ANY error. If it is desired that the testing continue on a bad-block error, even with the HOE switch set, answer No to the question.

9. HALT ON OTHER HARD ERRORS (#s 31-34, 36-37, 39-45) (L) Y ?

This question is similar to question 8, but refers to non-bad block type of Hard Errors.

```
; C 0424  1
; C 0425  1
; C 0426  1
; C 0427  1
; C 0428  1
; C 0429  1
; C 0430  1
; C 0431  1
; C 0432  1
; C 0433  1
; C 0434  1
; C 0435  1
; C 0436  1
; C 0437  1
; C 0438  1
; C 0439  1
; C 0440  1
; C 0441  1
; C 0442  1
; C 0443  1
; C 0444  1
; C 0445  1
; C 0446  1
; C 0447  1
; C 0448  1
; C 0449  1
; C 0450  1
; C 0451  1
; C 0452  1
; C 0453  1
; C 0454  1
; C 0455  1
; C 0456  1
; C 0457  1
; C 0458  1
; C 0459  1
; C 0460  1
; C 0461  1
; C 0462  1
; C 0463  1
; C 0464  1
; C 0465  1
; C 0466  1
; C 0467  1
; C 0468  1
; C 0469  1
; C 0470  1
; C 0471  1
; C 0472  1
; C 0473  1
; C 0474  1
; C 0475  1
; C 0476  1
```

10. HALT ON SOFT ERRORS (#s 50-54) (L) N ?

    This question is similar to question 8, but refers to Soft Errors.

11. COUNT EACH RETRY AS A SEPARATE SOFT ERROR (L) N ?

    On encountering any error on a read/write, the controller retries the operation a number of times. If the operation is eventually successful, this is reported as a Soft Error. The error log packet contains the number of retries performed before the operation was successful. Normally, the whole sequence of retries is classified as one Soft Error. Answer Yes to the question if it is desired to count each internal retry attempt as a separate Soft Error.

12. RANDOM SEEK MODE (L) Y ?

    Answer YES to cause block numbers to be chosen randomly. Answer NO to cause block numbers to be selected sequentially.

13. UNITS TO BE SELECTED AT RANDOM (NO, IMPLIES SEQUENTIAL) (L) N ?

    This question is optionally asked if the answer to the previous question is N[o]. The selection of units for sequential operations is affected by the answer to this question. If the default answer is chosen (N[o]), then units shall be selected in a predetermined manner in accordance with the typical seek time margins for each drive. If the alternate answer is chosen (Y[es]), then the units will be chosen at random in accordance with the percentages specified in Software question 4.

14. READ-COMPARES PERFORMED AT THE CONTROLLER (L) Y ?

    Answering YES causes all read commands to include the "compare" modifier. This essentially forces the controller to perform two read operations on the same disk address, and to compare the results.

    The following message will appear after the operator has answered this question:

    THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED DISK UNITS.

15. WRITE-COMPARES PERFORMED AT THE CONTROLLER (L) N ?

    Answering YES causes all write I/O requests to be changed to write-compare. After each write, the controller will read the data and compare it to data re-obtained from the host.

```
; C 0477  1
; C 0478  1
; C 0479  1
; C 0480  1
; C 0481  1
; C 0482  1
; C 0483  1
; C 0484  1
; C 0485  1
; C 0486  1
; C 0487  1
; C 0488  1
; C 0489  1
; C 0490  1
; C 0491  1
; C 0492  1
; C 0493  1
; C 0494  1
; C 0495  1
; C 0496  1
; C 0497  1
; C 0498  1
; C 0499  1
; C 0500  1
; C 0501  1
; C 0502  1
; C 0503  1
; C 0504  1
; C 0505  1
; C 0506  1
; C 0507  1
; C 0508  1
```

16. CHECK ALL WRITES AT HOST BY READING (L) Y ?

    This question will only be asked if the previous question was answered NO. Answering YES causes all writes to be checked by the host by reading the data immediately after the write operation. This option consumes extra CPU time, and doubles the amount of storage required for writes. Therefore, it is only recommended when drive write-compare operations are suspect.

17. USER-DEFINED DATA PATTERN (L) N ?

    An answer of YES allows the operator to define his/her own data pattern to be used in all write operations. A NO answer will allow the operator to select a pre-defined data pattern in the next question.

18. SELECT PRE-DEFINED DATA PATTERN (0 FOR SEQUENTIAL SELECTION) (D) 0 ?

    There are 21 pre-defined data patterns available, selected as 1 to 21 (see section 4.9). A zero answer will cause patterns 1 to 21 to be sequentially selected for each write. (Note that pattern 1 consists entirely of random numbers).

19. NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM) (D) 16 ?
    PATTERN VALUES (O) ?

    These questions will only be asked if the operator has decided to define his/her own data pattern. The actual bit patterns will be entered as octal (PDP-11).

```
;C 0509  1               3.0   ERROR TYPES
;C 0510  1                     ----------
;C 0511  1
;C 0512  1                     This program has four types of error  classifications;
;C 0513  1                     system fatal, drive fatal, hard and soft.
;C 0514  1
;C 0515  1                     SYSTEM FATAL ERRORS
;C 0516  1                     --------- ------
;C 0517  1
;C 0518  1                     System fatal errors are used to indicate that an error
;C 0519  1                     was detected by the  Diagnostic Supervisor in relation
;C 0520  1                     to loading/controlling the diagnostic process.
;C 0521  1
;C 0522  1                     The content  of each  error is  such that it should be
;C 0523  1                     self explanatory.  However,  the messages utilize some
;C 0524  1                     terms that are specific to the disk subsystem, and may
;C 0525  1                     require some getting use to.
;C 0526  1
;C 0527  1
;C 0528  1                     DRIVE FATAL ERRORS
;C 0529  1                     -----------------
;C 0530  1
;C 0531  1                     Drive fatal errors are a result of:
;C 0532  1
;C 0533  1                     an error that is  considered fatal to  the drive,  but
;C 0534  1                     testing will continue.
;C 0535  1
;C 0536  1
;C 0537  1                     HARD ERRORS
;C 0538  1                     -----------
;C 0539  1
;C 0540  1                     Hard errors are a result of:
;C 0541  1
;C 0542  1                             1.  retries of a soft error or *
;C 0543  1                             2.  a non-recoverable error
;C 0544  1                             3.  a soft error if retries are not set.
;C 0545  1
;C 0546  1                             *  Note: Retries are executed in the controller
;C 0547  1
;C 0548  1                     SOFT ERRORS
;C 0549  1                     -----------
;C 0550  1
;C 0551  1                     Soft errors  are media related errors.  All soft errors
;C 0552  1                     will be retried by the controller.
;C 0553  1
;C 0554  1                     Note: Soft errors are retrieved from the controller via
;C 0555  1                     the error log  capabilities of  MSCP.
```

```
; C 0556  1
; C 0557  1
; C 0558  1
; C 0559  1
; C 0560  1
; C 0561  1
; C 0562  1
; C 0563  1
; C 0564  1
; C 0565  1
; C 0566  1
; C 0567  1
; C 0568  1
; C 0569  1
; C 0570  1
; C 0571  1
; C 0572  1
; C 0573  1
; C 0574  1
; C 0575  1
; C 0576  1
; C 0577  1
; C 0578  1
; C 0579  1
; C 0580  1
; C 0581  1
; C 0582  1
; C 0583  1
; C 0584  1
; C 0585  1
; C 0586  1
; C 0587  1
; C 0588  1
; C 0589  1
; C 0590  1
; C 0591  1
; C 0592  1
; C 0593  1
; C 0594  1
; C 0595  1
; C 0596  1
; C 0597  1
; C 0598  1
; C 0599  1
; C 0600  1
; C 0601  1
; C 0602  1
; C 0603  1
; C 0604  1
```

3.1 ERROR INFORMATION

All general error messages will include the type of error (system-fatal, drive-fatal, hard, soft) and a unit number. If the error applies to a controller, then only the first unit number of the controller will be given. (The user will know the other unit numbers when subsequent "drop unit" messages are printed).

Basic error messages provide more details about the error. The Exerciser will print all basic error messages, along with the disk address, if applicable. In some cases where a drive-fatal error applies to a controller, the controller's IP address will be printed.

Extended error messages will be used to print the relevant fields of command and end message packets, status codes, SA register contents, and error log messages. All values will be in octal (PDP-11).

The error messages in this section do not include errors detected and printed by the Diagnostic Supervisor.

3.2 INITIALIZATION ERRORS

Two kinds of errors will be reported to the operator during the Initialization Test. The System-fatal error is too many units specified. A system-fatal error will cause the Exerciser to abort.

Drive-fatal errors only affect the unit(s) involved. Testing will continue on all other units. This class of errors includes, but is not limited to, the following:

1. Register Existence Test failure (no drive present)
2. Vector Test failure
3. BR Level Test failure
4. Initialization sequence failure
5. Online failed
6. Access failed

; C 0605 1
; C 0606 1
; C 0607 1
; C 0608 1
; C 0609 1
; C 0610 1
; C 0611 1
; C 0612 1
; C 0613 1
; C 0614 1
; C 0615 1
; C 0616 1
; C 0617 1
; C 0618 1
; C 0619 1
; C 0620 1
; C 0621 1
; C 0622 1
; C 0623 1
; C 0624 1
; C 0625 1
; C 0626 1
; C 0627 1
; C 0628 1
; C 0629 1
; C 0630 1
; C 0631 1
; C 0632 1
; C 0633 1
; C 0634 1
; C 0635 1
; C 0636 1
; C 0637 1
; C 0638 1
; C 0639 1
; C 0640 1
; C 0641 1
; C 0642 1
; C 0643 1
; C 0644 1
; C 0645 1
; C 0646 1
; C 0647 1
; C 0648 1
; C 0649 1
; C 0650 1
; C 0651 1

3.3 EXERCISER ERRORS

Most errors reported during this test will originate from MSCP end message packets. The status code field will be converted to text and printed as part of a basic error message. Any subcode value will follow if extended error messages are enabled.

The following list represents some of the error conditions reported via MSCP:

1. Disk unit went offline (a sub-code may follow detailing the reason)
2. Compare error
3. Data error (a sub-code may follow)
4. Drive error (a sub-code may follow)
5. Host buffer access error
6. Media format error (a sub-code may follow)

3.4 ERROR LOG MESSAGES

The contents of the error-log messages received from the controller are printed as received, and should be deciphered using the MSCP specs.

3.5 MSCP ERRORS

An MSCP error occurs when the host receives an Invalid Command End Message from the controller. In such cases, the host will print out the erroneous command followed by the reason for the error. If extended printouts are enabled, then the entire contents of the end message will be displayed in octal without interpretation of the data.

```
; C 0652  1
; C 0653  1
; C 0654  1
; C 0655  1
; C 0656  1
; C 0657  1
; C 0658  1
; C 0659  1
; C 0660  1
; C 0661  1
; C 0662  1
; C 0663  1
; C 0664  1
; C 0665  1
; C 0666  1
; C 0667  1
; C 0668  1
; C 0669  1
; C 0670  1
; C 0671  1
; C 0672  1
; C 0673  1
; C 0674  1
; C 0675  1
; C 0676  1
; C 0677  1
; C 0678  1
; C 0679  1
; C 0680  1
; C 0681  1
; C 0682  1
```

3.6 SAMPLE ERROR STATEMENT

The errors listed by the exerciser are usually very descriptive and are self explanatory. The following is an example error statement. This error statement is the extended error message.

```
(example)                                (comments)

DISK XXX                                 !DISK UNIT NUMBER
INVALID COMMAND                          !MAJOR STATUS CODE RECEIVED BACK
SUB-CODE XXXX                            !SUB-CODE OF GIVEN COMMAND
COMMAND: READ                            !COMMAND GIVEN TO DRIVE
LBN: XXXXX                               !LOGICAL BLOCK NUMBER GIVEN
BYTE COUNT IN COMMAND XXXXX              !NUMBER OF BYTES WANTED TO READ
ACTUAL # OF BYTES TRANSFERRED XXXXX !NUMBER OF BYTES ACTUALLY READ
```

The status code in an end messages is broken into two pieces. The first 5 bits represent the major status which is given by the "invalid command" message. The 11 remaining bits represent the sub-code, which tells in greater detail the error in the controller. The LBN is the logical block on the disk the controller was trying to read. The byte count refers to the number of bytes the controller was going to read off the LBN. The actual number of bytes transferred refers to the number of bytes read before the error.

```
; C 0683  1
; C 0684  1
; C 0685  1
; C 0686  1
; C 0687  1
; C 0688  1
; C 0689  1
; C 0690  1
; C 0691  1
; C 0692  1
; C 0693  1
; C 0694  1
; C 0695  1
; C 0696  1
; C 0697  1
; C 0698  1
; C 0699  1
; C 0700  1
; C 0701  1
; C 0702  1
; C 0703  1
; C 0704  1
; C 0705  1
; C 0706  1
; C 0707  1
; C 0708  1
; C 0709  1
; C 0710  1
; C 0711  1
; C 0712  1
; C 0713  1
; C 0714  1
; C 0715  1
; C 0716  1
```

4.0 PERFORMANCE AND PROGRESS REPORTS

A summary report is printed at the end of each pass of the Exerciser or upon demand by the operator. The fields may be cleared to zero after the report is printed depending on the operator's response to this option in the software questions. Any units added to the test cycle will also begin with cleared statistics.

Errors are grouped into two basic categories: hard and soft. Each is sub divided into four more categories, depending on the most probable classification for that error.

The sub categories are:

1. disk related errors

2. seek (or format) related errors

3. controller or drive related errors

4. host (the CPU) related errors.

All numeric values are in decimal radix.

UNT #	TYPE	# OF READS	BYTS READ	# OF WRITES	BYTES WRITTEN	HRD ERS DAT	HRD ERS SEK	HRD ERS DRV	HRD ERS HST	SFT ERS DAT	SFT ERS SEK	SFT ERS DRV	SFT ERS HST
x	xxxx	xxxx	xxxx	xxxxx	xxxxxx	x	x	x	x	x	x	x	x
.	..	..	..	...	....	.	.	.	.	.	.	.	.
.	..	..	..	...	....	.	.	.	.	.	.	.	.

```
; C 0717  1
; C 0718  1
; C 0719  1
; C 0720  1
; C 0721  1
; C 0722  1
; C 0723  1
; C 0724  1
; C 0725  1
; C 0726  1
; C 0727  1
; C 0728  1
; C 0729  1
; C 0730  1
; C 0731  1
; C 0732  1
; C 0733  1
; C 0734  1
; C 0735  1
; C 0736  1
; C 0737  1
; C 0738  1
; C 0739  1
; C 0740  1
; C 0741  1
; C 0742  1
; C 0743  1
; C 0744  1
; C 0745  1
; C 0746  1
; C 0747  1
; C 0748  1
; C 0749  1
; C 0750  1
; C 0751  1
; C 0752  1
; C 0753  1
; C 0754  1
; C 0755  1
; C 0756  1
; C 0757  1
; C 0758  1
; C 0759  1
; C 0760  1
; C 0761  1
; C 0762  1
; C 0763  1
; C 0764  1
; C 0765  1
; C 0766  1
```

5.0 TEST SUMMARY

This exerciser consists of two parts: the initialization subtest, and the performance exerciser. The operator is not able to select which of these two parts he/she wishes to run; they both must be executed.

5.1 INITIALIZATION SUBTEST

The purpose of this subtest is to verify the hardware configuration as specified by the operator, and to bring each unit online. The Initialization Subtest will always precede the execution of any other test.

First, the presence of each drive register will be verified, along with a check on the BR level specified by the operator. Then, an initialization will be issued to each controller configured for testing. When the initialization sequence has been completed, an attempt will be made to bring each unit online. If this succeeds, one or two MSCP reads will be issued to the inner-most LBN of each selected disk to to ensure that each disk drive can seek and be read.

Any drive-fatal or hard errors encountered during this test will cause the appropriate unit(s) to be dropped. If basic error messages are enabled, then the program will print out the specific reason for dropping the unit(s). Henceforth, the failed unit(s) will not be tested unless the operator intervenes (adds unit(s) or restarts Exerciser).

Upon successful completion of the Initialization Subtest, the program will begin executing the Exerciser.

5.2 EXERCISER

The purpose of this subtest is to exercise the disk drives in a manner similar to the typical usage under standard operating systems. Execution of this test should give an indication of the operating performance of the disk drive subunits. This test will utilize random disk addresses, random word counts, and data patterns, all subject to the limits and specifications made by the operator. All protected disks will be subject to read-only operations, while unprotected disks may be read or written, depending on the answers given to the software parameter questions. End-of-pass will be declared when the specified number of bytes have been transferred for all the disks taken as a whole.

```
; C 0767  1
; C 0768  1
; C 0769  1
; C 0770  1
; C 0771  1
; C 0772  1
; C 0773  1
; C 0774  1
; C 0775  1
; C 0776  1
; C 0777  1
; C 0778  1
; C 0779  1
; C 0780  1
; C 0781  1
; C 0782  1
; C 0783  1
; C 0784  1
; C 0785  1
; C 0786  1
; C 0787  1
; C 0788  1
; C 0789  1
; C 0790  1
; C 0791  1
; C 0792  1
; C 0793  1
; C 0794  1
; C 0795  1
```

If a read/write error occurs during this test, then the controller will initiate an appropriate number of retries. If all retries fail, then a hard error will be reported to the host, an error message will be displayed on the console terminal and the error will be tallied for the summary report. The unit will be dropped if the hard error count has exceeded the specified limit.

5.3 DROP UNIT SUMMARY

During the Initialization Subtest, individual units will be dropped from the test sequence if they are unable to be brought online or the operator specified drive does not match the hardware.

During the Exercise, the program will drop a unit for one of three reasons. The normal path is for each unit to complete the transfer of N megabytes, where N is specified by the operator during SW questioning and be soft-dropped. Otherwise, a unit will be hard-dropped if the number of hard errors encountered exceeds the operator-specified limit, or if a fatal error is detected. Units hard-dropped may later be added to the test cycle. However, statistics for the hard-added unit will be cleared to zero; if a transfer limit was specified, in which case the unit was soft-dropped, the statistics may or may not be cleared depending on the operators answer to Software question 12.

```
; C 0796  1
; C 0797  1
; C 0798  1
; C 0799  1
; C 0800  1
; C 0801  1
; C 0802  1
; C 0803  1
; C 0804  1
; C 0805  1
; C 0806  1
; C 0807  1
; C 0808  1
; C 0809  1
; C 0810  1
; C 0811  1
; C 0812  1
; C 0813  1
; C 0814  1
; C 0815  1
; C 0816  1
; C 0817  1
; C 0818  1
; C 0819  1
; C 0820  1
; C 0821  1
; C 0822  1
; C 0823  1
; C 0824  1
; C 0825  1
; C 0826  1
; C 0827  1
; C 0828  1
; C 0829  1
; C 0830  1
; C 0831  1
; C 0832  1
; C 0833  1
; C 0834  1
; C 0835  1
; C 0836  1
; C 0837  1
; C 0838  1
; C 0839  1
; C 0840  1
; C 0841  1
; C 0842  1
; C 0843  1
; C 0844  1
```

6.0 ERROR CODES GENERATED BY THIS EXERCISER

SYSTEM FATAL ERRORS

1 More than 4 units specified

DRIVE FATAL ERRORS

10	Controller couldn't be addressed at the address given.	Wrong IP address selected
11	Controller didn't interrupt at the interrupt vector given.	Wrong vector address selected.
12	Controller didn't interrupt at the BR level given.	Wrong BR level selected.
13	Init sequence failed.	Either one of the four initialization steps did not receive the correct response from the Controller, or one of the steps timed-out.
14	Fatal Controller error.	The error bit (bit 15) in the SA register was set.
15	Failed to bring unit on-line.	On-line response had an error code. (see also #s 22 and 23.)
16	Write protect conflict.	The unit was hardware write protected and write operations were requested on the unit.
17	Access to either the inner or the outer track failed.	Innermost or outermost track's header may be correpeated.
18	Unit went off-line.	---
19	Drive type not known.	The version of the Exerciser being run does not support this disk type.

20	Failed to send 'Set Controller Characteristics" command.	Either the unit is off-line or the Diagnostic is corrupted because of any problems with its RAM.
21	Controller returned wrong 'end code' for the 'Set Controller Characteristics' command.	Problem with the Controller microcode or the port/DMA interface.
22	Failed to send 'On-line' command	Either the unit is off-line or the diagnostic is corrupted because of any problems with its RAM.
23	Controller returned wrong 'end code' for the 'On-line' command.	Problem with the Controller's microcode or the port/DMA interface.
24	Drive went to the 'Available' state.	---

HARD ERRORS

31	Controller received an invalid command.	The diagnostic is corrupted because of any problems with its RAM, or there is a problem with the Controller microcode (RAM or ROM) or there is problem with the port/DMA interface.
32	Command aborted by the Controller.	Command timed-out in the Controller.
35	Media format error.	---
36	Drive write protected.	---
37	Controller read or write compare error.	---
38	Data error.	CRC error in the data field of a disk block.
39	Host buffer access error	---
40	Controller error.	Difficult to categorize without looking at the error sub-code or any associated error-log message.

```
;C 0898  1
;C 0899  1
;C 0900  1
;C 0901  1
;C 0902  1
;C 0903  1
;C 0904  1
;C 0905  1
;C 0906  1
;C 0907  1
;C 0908  1
;C 0909  1
;C 0910  1
;C 0911  1
;C 0912  1
;C 0913  1
;C 0914  1
;C 0915  1
;C 0916  1
;C 0917  1
;C 0918  1
;C 0919  1
;C 0920  1
;C 0921  1
;C 0922  1
;C 0923  1
;C 0924  1
;C 0925  1
;C 0926  1
;C 0927  1
;C 0928  1
;C 0929  1
;C 0930  1
;C 0931  1
;C 0932  1
;C 0933  1
;C 0934  1
;C 0935  1
;C 0936  1
;C 0937  1
;C 0938  1
;C 0939  1
;C 0940  1
;C 0941  1
;C 0942  1
;C 0943  1
;C 0944  1
;C 0945  1
;C 0946  1
;C 0947  1
;C 0948  1
;C 0949  1
;C 0950  1
```

41	Drive error.	See #40.
42	Host write compare error.	Error detected when Host CPU compared the data written and read back. May be a problem with the Host or Controller RAM.
43	Message from internal diagnostics	See #40.
44	Duplicate unit number detected by the Controller.	---
45	Unknown end code received.	Problem with the Controller microcode or the port/DMA interface.

SOFT ERRORS

50	Controller error.	See error-log packet for details as the exact cause may not be evident.
51	Host memory access error.	See #50.
52	Disk transfer error.	See #50.
53	'Standard Disk Interconnect' error.	See #50.
54	'Small Disk' error.	See #50.

DUP ERRORS

60	Unable to load local controller DUP media.
61	(Not used)
62	Illegal unit number.
63	Illegal relative or physical block.
64	Device error.
65	Zero length message.
66	Unknown DUP status code.
67	Invalid command.

```
;  C 0951  1
;  C 0952  1                68 No region available.
;  C 0953  1
;  C 0954  1                69 No region suitable.
;  C 0955  1
;  C 0956  1                70 Program not known.
;  C 0957  1
;  C 0958  1                71 Load failure.
;  C 0959  1
;  C 0960  1                72 Standalone.
;  C 0961  1
;  C 0962  1                73 Unknown DUP status code.
;  C 0963  1
;  C 0964  1
```

```
; C 0965  1
; C 0966  1
; C 0967  1
; C 0968  1
; C 0969  1
; C 0970  1
; C 0971  1
; C 0972  1
; C 0973  1
; C 0974  1
; C 0975  1
; C 0976  1
; C 0977  1
; C 0978  1
; C 0979  1
; C 0980  1
; C 0981  1
; C 0982  1
; C 0983  1
; C 0984  1
; C 0985  1
; C 0986  1
; C 0987  1
; C 0988  1
; C 0989  1
; C 0990  1
; C 0991  1
; C 0992  1
; C 0993  1
; C 0994  1
; C 0995  1
; C 0996  1
; C 0997  1
; C 0998  1
; C 0999  1
; C 1000  1
; C 1001  1
; C 1002  1
; C 1003  1
; C 1004  1
; C 1005  1
; C 1006  1
; C 1007  1
; C 1008  1
; C 1009  1
; C 1010  1
; C 1011  1
; C 1012  1
; C 1013  1
; C 1014  1
; C 1015  1
; C 1016  1
```

7.0 DATA PATTERNS

	HEX	OCTAL	BINARY
Pattern 1	R A N D O M	N U M B E R S	
Pattern 2	0000	000000	0 000 000 000 000 000
Pattern 3	FFFF	177777	1 111 111 111 111 111
Pattern 4	8B8B	105613	1 000 101 110 001 011
Pattern 5	3333	031463	0 011 001 100 110 011
Pattern 6	3091	030221	0 011 000 010 010 001
Pattern 7	0001	000001	0 000 000 000 000 001
	0003	000003	0 000 000 000 000 011
	0007	000007	0 000 000 000 000 111
	000F	000017	0 000 000 000 001 111
	001F	000037	0 000 000 000 011 111
	003F	000077	0 000 000 000 111 111
	007F	000177	0 000 000 001 111 111
	00FF	000377	0 000 000 011 111 111
	01FF	000777	0 000 000 111 111 111
	03FF	001777	0 000 001 111 111 111
	07FF	003777	0 000 011 111 111 111
	0FFF	007777	0 000 111 111 111 111
	1FFF	017777	0 001 111 111 111 111
	3FFF	037777	0 011 111 111 111 111
	7FFF	077777	0 111 111 111 111 111
	FFFF	177777	1 111 111 111 111 111
Pattern 8	FFFE	177776	1 111 111 111 111 110
	FFFC	177774	1 111 111 111 111 100
	FFF8	177770	1 111 111 111 111 000
	FFF0	177760	1 111 111 111 110 000
	FFE0	177740	1 111 111 111 100 000
	FFC0	177700	1 111 111 111 000 000
	FF80	177600	1 111 111 110 000 000
	FF00	177400	1 111 111 100 000 000
	FE00	177000	1 111 111 000 000 000
	FC00	176000	1 111 110 000 000 000
	F800	174000	1 111 100 000 000 000
	F000	170000	1 111 000 000 000 000
	E000	160000	1 110 000 000 000 000
	C000	140000	1 100 000 000 000 000
	8000	100000	1 000 000 000 000 000
	0000	000000	0 000 000 000 000 000

```
; C 1017  1          Pattern 9       0000    000000          0 000 000 000 000 000
; C 1018  1                          0000    000000          0 000 000 000 000 000
; C 1019  1                          0000    000000          0 000 000 000 000 000
; C 1020  1                          FFFF    177777          1 111 111 111 111 111
; C 1021  1                          FFFF    177777          1 111 111 111 111 111
; C 1022  1                          FFFF    177777          1 111 111 111 111 111
; C 1023  1                          0000    000000          0 000 000 000 000 000
; C 1024  1                          0000    000000          0 000 000 000 000 000
; C 1025  1                          FFFF    177777          1 111 111 111 111 111
; C 1026  1                          FFFF    177777          1 111 111 111 111 111
; C 1027  1                          0000    000000          0 000 000 000 000 000
; C 1028  1                          FFFF    177777          1 111 111 111 111 111
; C 1029  1                          0000    000000          0 000 000 000 000 000
; C 1030  1                          FFFF    177777          1 111 111 111 111 111
; C 1031  1                          0000    000000          0 000 000 000 000 000
; C 1032  1                          FFFF    177777          1 111 111 111 111 111
; C 1033  1
; C 1034  1          Pattern 10      B6D9    133331          1 011 011 011 011 001
; C 1035  1
; C 1036  1          Pattern 11      5555    052525          0 101 010 101 010 101
; C 1037  1                          5555    052525          0 101 010 101 010 101
; C 1038  1                          5555    052525          0 101 010 101 010 101
; C 1039  1                          AAAA    125252          1 010 101 010 101 010
; C 1040  1                          AAAA    125252          1 010 101 010 101 010
; C 1041  1                          AAAA    125252          1 010 101 010 101 010
; C 1042  1                          5555    052525          0 101 010 101 010 101
; C 1043  1                          5555    052525          0 101 010 101 010 101
; C 1044  1                          AAAA    125252          1 010 101 010 101 010
; C 1045  1                          AAAA    125252          1 010 101 010 101 010
; C 1046  1                          5555    052525          0 101 010 101 010 101
; C 1047  1                          AAAA    125252          1 010 101 010 101 010
; C 1048  1                          5555    052525          0 101 010 101 010 101
; C 1049  1                          AAAA    125252          1 010 101 010 101 010
; C 1050  1                          5555    052525          0 101 010 101 010 101
; C 1051  1                          AAAA    125252          1 010 101 010 101 010
```

```
; C 1052  1          Pattern 12      2D2D    026455      0 010 110 100 101 101
; C 1053  1                          2D2D    026455      0 010 110 100 101 101
; C 1054  1                          2D2D    026455      0 010 110 100 101 101
; C 1055  1                          D2D2    151322      1 101 001 011 010 010
; C 1056  1                          D2D2    151322      1 101 001 011 010 010
; C 1057  1                          D2D2    151322      1 101 001 011 010 010
; C 1058  1                          2D2D    026455      0 010 110 100 101 101
; C 1059  1                          2D2D    026455      0 010 110 100 101 101
; C 1060  1                          D2D2    151322      1 101 001 011 010 010
; C 1061  1                          D2D2    151322      1 101 001 011 010 010
; C 1062  1                          2D2D    026455      0 010 110 100 101 101
; C 1063  1                          2D2D    026455      0 010 110 100 101 101
; C 1064  1                          D2D2    151322      1 101 001 011 010 010
; C 1065  1                          2D2D    026455      0 010 110 100 101 101
; C 1066  1                          D2D2    151322      1 101 001 011 010 010
; C 1067  1                          2D2D    026455      0 010 110 100 101 101
; C 1068  1                          D2D2    151322      1 101 001 011 010 010
; C 1069  1                          2D2D    026455      0 010 110 100 101 101
; C 1070  1                          D2D2    151322      1 101 001 011 010 010
; C 1071  1                          2D2D    026455      0 010 110 100 101 101
; C 1072  1
; C 1073  1          Pattern 13      6DB6    066666      0 110 110 110 110 110
; C 1074  1
; C 1075  1          Pattern 14      0001    000001      0 000 000 000 000 001
; C 1076  1                          0002    000002      0 000 000 000 000 010
; C 1077  1                          0004    000004      0 000 000 000 000 100
; C 1078  1                          0008    000010      0 000 000 000 001 000
; C 1079  1                          0010    000020      0 000 000 000 010 000
; C 1080  1                          0020    000040      0 000 000 000 100 000
; C 1081  1                          0040    000100      0 000 000 001 000 000
; C 1082  1                          0080    000200      0 000 000 010 000 000
; C 1083  1                          0100    000400      0 000 000 100 000 000
; C 1084  1                          0200    001000      0 000 001 000 000 000
; C 1085  1                          0400    002000      0 000 010 000 000 000
; C 1086  1                          0800    004000      0 000 100 000 000 000
; C 1087  1                          1000    010000      0 001 000 000 000 000
; C 1088  1                          2000    020000      0 010 000 000 000 000
; C 1089  1                          4000    040000      0 100 000 000 000 000
; C 1090  1                          8000    100000      1 000 000 000 000 000
```

```
; C 1091  1          Pattern 15      FFFE    177776      1 111 111 111 111 110
; C 1092  1                          FFFD    177775      1 111 111 111 111 101
; C 1093  1                          FFFB    177773      1 111 111 111 111 011
; C 1094  1                          FFF7    177767      1 111 111 111 110 111
; C 1095  1                          FFEF    177757      1 111 111 111 101 111
; C 1096  1                          FFDF    177737      1 111 111 111 011 111
; C 1097  1                          FFBF    177677      1 111 111 110 111 111
; C 1098  1                          FF7F    177577      1 111 111 101 111 111
; C 1099  1                          FEFF    177377      1 111 111 011 111 111
; C 1100  1                          FDFF    176777      1 111 110 111 111 111
; C 1101  1                          FBFF    175777      1 111 101 111 111 111
; C 1102  1                          F7FF    173777      1 111 011 111 111 111
; C 1103  1                          EFFF    167777      1 110 111 111 111 111
; C 1104  1                          DFFF    157777      1 101 111 111 111 111
; C 1105  1                          BFFF    137777      1 011 111 111 111 111
; C 1106  1                          7FFF    077777      0 111 111 111 111 111
; C 1107  1
; C 1108  1          Pattern 16      B6D9    133331      1 011 011 011 011 001
; C 1109  1                          B6D9    133331      1 011 011 011 011 001
; C 1110  1                          B6D9    133331      1 011 011 011 011 001
; C 1111  1                          DB6C    155554      1 101 101 101 101 100
; C 1112  1                          DB6C    155554      1 101 101 101 101 100
; C 1113  1                          DB6C    155554      1 101 101 101 101 100
; C 1114  1                          B6D9    133331      1 011 011 011 011 001
; C 1115  1                          B6D9    133331      1 011 011 011 011 001
; C 1116  1                          DB6C    155554      1 101 101 101 101 100
; C 1117  1                          DB6C    155554      1 101 101 101 101 100
; C 1118  1                          B6D9    133331      1 011 011 011 011 001
; C 1119  1                          DB6C    155554      1 101 101 101 101 100
; C 1120  1                          B6D9    133331      1 011 011 011 011 001
; C 1121  1                          DB6C    155554      1 101 101 101 101 100
; C 1122  1                          B6D9    133331      1 011 011 011 011 001
; C 1123  1                          DB6C    155554      1 101 101 101 101 100
```

```
; C 1124  1          Pattern 17      (LBN)*   (LBN)          (LBN)
; C 1125  1                          8D36     106466    1 000 110 100 110 110
; C 1126  1                          8D36     106466    1 000 110 100 110 110
; C 1127  1                          72C9     071311    0 111 001 011 001 001
; C 1128  1                          72C9     071311    0 111 001 011 001 001
; C 1129  1                          72C9     071311    0 111 001 011 001 001
; C 1130  1                          8D36     106466    1 000 110 100 110 110
; C 1131  1                          8D36     106466    1 000 110 100 110 110
; C 1132  1                          8D36     106466    1 000 110 100 110 110
; C 1133  1                          8D36     106466    1 000 110 100 110 110
; C 1134  1                          72C9     071311    0 111 001 011 001 001
; C 1135  1                          72C9     071311    0 111 001 011 001 001
; C 1136  1                          72C9     071311    0 111 001 011 001 001
; C 1137  1                          72C9     071311    0 111 001 011 001 001
; C 1138  1                          72C9     071311    0 111 001 011 001 001
; C 1139  1                          8D36     106466    1 000 110 100 110 110
; C 1140  1                          8D36     106466    1 000 110 100 110 110
; C 1141  1                          8D36     106466    1 000 110 100 110 110
; C 1142  1                          8D36     106466    1 000 110 100 110 110
; C 1143  1                          8D36     106466    1 000 110 100 110 110
; C 1144  1                          8D36     106466    1 000 110 100 110 110
; C 1145  1
; C 1146  1          * This word position contains the number of the locical
; C 1147  1            block to be written.
; C 1148  1
; C 1149  1
; C 1150  1
; C 1151  1          Pattern 18      8D36     106466    1 000 110 100 110 110
; C 1152  1                          (LBN)    (LBN)          (LBN)
; C 1153  1                          72C9     071311    0 111 001 011 001 001
; C 1154  1                          8D36     106466    1 000 110 100 110 110
; C 1155  1                          8D36     106466    1 000 110 100 110 110
; C 1156  1                          8D36     106466    1 000 110 100 110 110
; C 1157  1                          72C9     071311    0 111 001 011 001 001
; C 1158  1                          72C9     071311    0 111 001 011 001 001
; C 1159  1                          72C9     071311    0 111 001 011 001 001
; C 1160  1                          72C9     071311    0 111 001 011 001 001
; C 1161  1                          8D36     106466    1 000 110 100 110 110
; C 1162  1                          8D36     106466    1 000 110 100 110 110
; C 1163  1                          8D36     106466    1 000 110 100 110 110
; C 1164  1                          8D36     106466    1 000 110 100 110 110
; C 1165  1                          8D36     106466    1 000 110 100 110 110
; C 1166  1                          72C9     071311    0 111 001 011 001 001
; C 1167  1                          72C9     071311    0 111 001 011 001 001
; C 1168  1                          72C9     071311    0 111 001 011 001 001
; C 1169  1                          72C9     071311    0 111 001 011 001 001
; C 1170  1                          72C9     071311    0 111 001 011 001 001
; C 1171  1                          72C9     071311    0 111 001 011 001 001
```

```
; C 1172  1          Pattern 19      (LBN)   (LBN)                 (LBN)
; C 1173  1                          B999    134631          1 011 100 110 011 001
; C 1174  1                          B999    134631          1 011 100 110 011 001
; C 1175  1                          4666    043146          0 100 011 001 100 110
; C 1176  1                          4666    043146          0 100 011 001 100 110
; C 1177  1                          4666    043146          0 100 011 001 100 110
; C 1178  1                          B999    134631          1 011 100 110 011 001
; C 1179  1                          B999    134631          1 011 100 110 011 001
; C 1180  1                          B999    134631          1 011 100 110 011 001
; C 1181  1                          B999    134631          1 011 100 110 011 001
; C 1182  1                          4666    043146          0 100 011 001 100 110
; C 1183  1                          4666    043146          0 100 011 001 100 110
; C 1184  1                          4666    043146          0 100 011 001 100 110
; C 1185  1                          4666    043146          0 100 011 001 100 110
; C 1186  1                          4666    043146          0 100 011 001 100 110
; C 1187  1                          B999    134631          1 011 100 110 011 001
; C 1188  1                          B999    134631          1 011 100 110 011 001
; C 1189  1                          B999    134631          1 011 100 110 011 001
; C 1190  1                          B999    134631          1 011 100 110 011 001
; C 1191  1                          B999    134631          1 011 100 110 011 001
; C 1192  1                          B999    134631          1 011 100 110 011 001
; C 1193  1
; C 1194  1          Pattern 20      B999    134631          1 011 100 110 011 001
; C 1195  1                          (LBN)   (LBN)                 (LBN)
; C 1196  1                          4666    043146          0 100 011 001 100 110
; C 1197  1                          B999    134631          1 011 100 110 011 001
; C 1198  1                          B999    134631          1 011 100 110 011 001
; C 1199  1                          B999    134631          1 011 100 110 011 001
; C 1200  1                          4666    043146          0 100 011 001 100 110
; C 1201  1                          4666    043146          0 100 011 001 100 110
; C 1202  1                          4666    043146          0 100 011 001 100 110
; C 1203  1                          4666    043146          0 100 011 001 100 110
; C 1204  1                          B999    134631          1 011 100 110 011 001
; C 1205  1                          B999    134631          1 011 100 110 011 001
; C 1206  1                          B999    134631          1 011 100 110 011 001
; C 1207  1                          B999    134631          1 011 100 110 011 001
; C 1208  1                          B999    134631          1 011 100 110 011 001
; C 1209  1                          4666    043146          0 100 011 001 100 110
; C 1210  1                          4666    043146          0 100 011 001 100 110
; C 1211  1                          4666    043146          0 100 011 001 100 110
; C 1212  1                          4666    043146          0 100 011 001 100 110
; C 1213  1                          4666    043146          0 100 011 001 100 110
; C 1214  1                          4666    043146          0 100 011 001 100 110
; C 1215  1
; C 1216  1          Pattern 21      (LBN)   (LBN)                 (LBN)
; C 1217  1
;   1218  1      )%
```

F3


```
;    1219  1     %sbttl 'PROGRAM HEADER'
;    1220  1
;    1221  1     library 'ZRQAE0.L16';                   ! RDRX EXERCISER GLOBAL LIBRARY
;    1222  1
;    1223  1     require 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
;    2714  1
;    2715  1     literal
;    2716  1         DS$NBR_OF_TESTS = 1;                ! NUMBER OF TESTS IN THIS DIAGNOSTIC
;    2717  1
;    2718  1     EQUALS;
;    2719  1
;    2720  1     POINTER (ALL);
;    2721  1
;    2722  1     !+
;    2723  1     ! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;    2724  1     ! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;    2725  1     !-
;    2726  1
;    2727  1     !ZZZ HEADER (%ascii'ZRQA', %ascii'E', %ascii'0', 32767, 1, PRI00);
;    2728  1     HEADER (%ascii'ZRQA', %ascii'E', %ascii'0', 32000, 1, PRI00);       !ZZZ NEED POSITIVE NUMBER
;    2729  1
```

```
;    2730  1     %sbttl 'DISPATCH TABLE'
;    2731  1
;    2732  1     !+
;    2733  1     ! THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;    2734  1     ! IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;    2735  1     !-
;    2736  1
;    2737  1     DISPATCH (DS$NBR_OF_TESTS);
```

```
;    2738  1      %sbttl 'GLOBAL DATA SECTION'
;    2739  1
;    2740  1      !+
;    2741  1      ! THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
;    2742  1      ! IN MORE THAN ONE TEST.
;    2743  1      !-
;    2744  1
;    2745  1      psect
;    2746  1          global = $FFF$ (read, write, noexecute, global, concatenate);
;    2747  1
;    2748  1      global
;    2749  1          CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    2750  1                              ! RUN-TIME CONTROLLER STATUS TABLES
;    2751  1          CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    2752  1                              ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    2753  1          DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    2754  1                              ! DRIVER CONTROLLER TABLES
;    2755  1          DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    2756  1                              ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    2757  1          RDRX_ADDR : ref rdrx field (RC_REG),
;    2758  1                              ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    2759  1          IRDRX_ADDR : ref rdrx field (RC_REG),
;    2760  1                              ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    2761  1
;    2762  1          BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
;    2763  1                              !CONTAINS LO+ HI LBN FIELDS FOR SEQUENTIAL   !ZZZ
;    2764  1                              !I/O TRANSFER FOR EACH UNIT.                 !ZZZ
;    2765  1          TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    2766  1                              ! STATISTICS TABLES
;    2767  1          T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    2768  1                              ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    2769  1
;    2770  1          DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    2771  1                              !INFO FROM RECEIVE + SEND CMDS                ZZZ
;    2772  1          TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED] INITIAL (BYTE (REP    !ZZZ
;    2773  1                  MAX_UNITS OF (1))),          !CURRENT TRACK DIRECTION     ZZZ
;    2774  1          RDM_CNT : WORD INITIAL (RDM_LEN),    !NO OF RANDOM NOS   \\KEEP   ZZZ
;    2775  1          RANDOM : VECTOR [RDM_LEN, WORD],     !RANDOM NO. TABLE   //TOGETHER   ZZZ
;    2776  1
;    2777  1          C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    2778  1                              ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    2779  1          MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;    2780  1                              ! MSCP PACKET POOL
;    2781  1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    2782  1                              ! ADDRESS OF AN MSCP PACKET (INTERUPT PROCESSING)
;    2783  1          PKT_USE : vector [PKT_CNT, byte, signed],
;    2784  1                              ! MSCP PACKET POOL ALLOCATION TABLE
;    2785  1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;    2786  1                              ! RETURN PACKET POOL
;    2787  1          RP_USE : vector [RP_CNT, byte, signed],
;    2788  1                              ! RETURN PACKET POOL ALLOCATION TABLE
;    2789  1          RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;    2790  1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
```

```
;    2791  1                              ! CURRENT RETURN PACKET ADDRESS
;    2792  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;    2793  1                              ! ERROR-LOG PACKET SAVE AREA
;    2794  1          BUFF_ADDR : vector [MAX_BUF_CNT],                  ! TABLE OF I/O BUFFER DESCRIPTORS
;    2795  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],     ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    2796  1          IODQ : vector [IODQ_LEN, byte],                    ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;    2797  1          IODQ_IN : word,                                    ! I/O DONE QUEUE IN POINTER
;    2798  1          IODQ_OUT : word,                                   ! I/O DONE QUEUE OUT POINTER
;    2799  1          ENTRY_REASON : byte,                               ! CURRENT OPERATOR COMMAND
;    2800  1          EOP_FLAG : byte,                                   ! END-OF-PASS FLAG
;    2801  1          DUP_FLACS : WORD,                                  !DUP FLAGS       ZZZ
;    2802  1          CCTLR : word,                                      ! NUMBER OF "CURRENT" CONTROLLER
;    2803  1          CDISK : word,                                      ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2804  1          CUOFF : word,                                      ! CURRENT UNIT CST OFFSET
;    2805  1          CTLR_CNT : word,                                   ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    2806  1          DUR : vector [MAX_UNITS, byte],                    ! DROP UNIT REASON
;    2807  1          QIO : vector [MAX_CTLR, byte],                     ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    2808  1          FREE_MEM_ADDR,                                     ! START OF FREE MEMORY
;    2809  1          BYTS_PER_QIO : word,                               ! SIZE (BYTES) OF AN I/O BUFFER
;    2810  1          ST_CODE : word,                                    ! CURRENT STATUS CODE
;    2811  1          SB_CODE : word,                                    ! CURRENT SUB-CODE
;    2812  1          STEP : word,                                       ! CURRENT STEP IN HARD_INIT
;    2813  1          OF_RC : signed word,                               ! OFFSET (0 OR 2) TO READ IP OR SA
;    2814  1          SA_REG : word,                                     ! STORAGE FOR SA REGISTER READS AND WRITES
;    2815  1          CMD_TIME : word,                                   ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    2816  1          NEX : word,                                        ! NON-EXISTENT MEMORY TRAP INDICATOR
;    2817  1          CRN_LOW : word,                                    ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    2818  1          CRN_HIGH : word,                                   ! COMMAND REF NUMBER (HI ORDER)
;    2819  1          CREDIT_BAL : word,                                 ! CREDIT BALANCE
;    2820  1          NEXT_PKT_USE : byte,                               ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    2821  1          HOURS : byte,                                      ! TIME OF DAY (HOURS)
;    2822  1          MINUTES : byte,                                    ! TIME OF DAY (MINUTES)
;    2823  1          CLK_TICKS : word,                                  ! TIME OF DAY (LINE-CLOCK TICKS)
;    2824  1          CLK_PRESENT : byte,                                ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    2825  1          HOE_FLAG : byte,                                   ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    2826  1
;    2827  1          S_PATTERN : WORD,                 !PATTERN FOR DUP WRITES          ZZZ
;    2828  1          S_DUPPKT : WORD,                  !DBN BYTE COUNTER                ZZZ
;    2829  1          P_INDEX : SIGNED WORD,            !CURRENT MESSAGE PACKET INDEX    ZZZ
;    2830  1          FORCED_ERROR : byte,                                ! "FORCED ERROR" DETECTED IN LAST READ
;    2831  1          FER_LBN : word,                                     ! LBN OF THE "FORCED ERROR" BLOCK
;    2832  1          FER_BC : word,                                      ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    2833  1          INIT_OCCURED : byte initial (byte (FALSE)),         ! EXERCISER INITIALIZATION COMPLETE
;    2834  1          ADDR_VECT_OK : byte initial (byte (FALSE));         ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    2835  1
;    2836  1      ERRTBL;
```

```
;    2837  1     %sbttl 'GLOBAL TEXT SECTION'
;    2838  1
;    2839  1     !+
;    2840  1     ! THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
;    2841  1     ! MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
;    2842  1     ! MORE THAN ONE TEST.
;    2843  1     !-
;    2844  1
;    2845  1     global bind
;    2846  1     !
;    2847  1     ! HARDWARE DIALOG
;    2848  1     !
;    2849  1         HWQ1 = uplit (%asciz'IP address'),
;    2850  1         HWQ2 = uplit (%asciz'Vector'),
;    2851  1         HWQ3 = uplit (%asciz'BR LEVEL [USUALLY 4-RQDX1 5-RUX50]'),          !ZZZ
;    2852  1         HWQ4 = uplit (%asciz'Drive number'),                                !ZZZ
;    2853  1         HWQ5 = uplit (%asciz'Test entire customer area of this disk'),      !ZZZ
;    2854  1         HWQ6A = uplit (%asciz'Lower octal word of beginning LBN address'),  !ZZZ
;    2855  1         HWQ6B = uplit (%asciz'Higher octal word of beginning LBN address'), !ZZZ
;    2856  1         HWQ7A = uplit (%asciz'Lower octal word of ending LBN address'),     !ZZZ
;    2857  1         HWQ7B = uplit (%asciz'Higher octal word of ending LBN address'),    !ZZZ
;    2858  1         HWQ8 = uplit (%asciz'Write on customer data area of this disk'),
;    2859  1         HWQ9 = uplit (%asciz'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
;    2860  1         HWQ10 = uplit (%asciz'Also run DUP exerciser'),                     !ZZZ
;    2861  1         HWQ11 = uplit (%asciz'Write on diagnostic area'),                   !ZZZ
;    2862  1     !
;    2863  1     ! SOFTWARE DIALOG
;    2864  1     !
;    2865  1         SWQ1  = uplit (%asciz'Hard Error limit'),
;    2866  1         SWQ2  = uplit (%asciz'Transfer limit in megabytes (0 for "Quick pass")'),
;    2867  1         SWQ4  = uplit (%asciz'Random seek mode'),
;    2868  1         SWQ7  = uplit (%asciz'Read-compares performed at the Controller'),
;    2869  1         SWQ9  = uplit (%asciz'Write-compares performed at the Controller'),
;    2870  1         SWQ10 = uplit (%asciz'Check all Writes at Host by reading'),
;    2871  1         SWQ11 = uplit (%asciz'User-defined data pattern'),
;    2872  1         SWQ12 = uplit (%asciz'Select pre-defined data pattern (0 for Sequential selection)'),
;    2873  1         SWQ13 = uplit (%asciz'Number of words in data pattern (16 maximum)'),
;    2874  1         SWQ14 = uplit (%asciz'Pattern value (No leading zeros allowed)'),
;    2875  1         SWQ15 = uplit (%asciz'Clear statistical tables after printing'),
;    2876  1         SWQ17 = uplit (%asciz'Percentage of "Fixed Disk" operations out of total operations'),
;    2877  1         SWQ19 = uplit (%asciz'Units to be selected at random (No. implies sequential)'),
;    2878  1         SWQ20 = uplit (%asciz'Want to rewrite blocks when "Forced Error" detected on reads'),
;    2879  1         SWQ21 = uplit (%asciz'Do you want to halt on other Hard Errors (#s 31-34, 36-37, 39-45)'),
;    2880  1         SWQ22 = uplit (%asciz'Do you want to halt on Soft Errors (#s 50-54)'),
;    2881  1         SWQ23 = uplit (%asciz'Do you want to halt on Bad-block Hard Errors (#s 35, 38)'),
;    2882  1         SWQ24 = uplit (%asciz'Enter time as HHMM (Example: 1305 -  No leading zeros allowd)'),
;    2883  1         SWQ25 = uplit (%asciz'Count each retry on a Read/Write error as a seperate Soft Error'),
;    2884  1         SWM1  = uplit (%asciz'The remaining questions only apply to unprotected disks'),
;    2885  1         NULL  = uplit (%asciz''),
;    2886  1
;    2887  1     !++
;    2888  1     !  THE FOLLOWING DBMs ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
;    2889  1     !  RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
```

```
;    2890  1      !  FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
;    2891  1      !--
;    2892  1
;    2893  1          DBM5   = uplit (%asciz'%N%A** Drop unit %D2'),
;    2894  1          DBM12  = uplit (%asciz'%N%A** PROC_RETPKT: Conn ID %O6%A received'),
;    2895  1          DBM15  = uplit (%asciz'%N%A** Multi-drive test'),
;    2896  1          DBM18  = uplit (%asciz'%N%A** FATAL_ERROR: RETPKT not available'),
;    2897  1          DBM19  = uplit (%asciz'%N%A** FSET_UPAR: Can''t find disk %D3%A in CST %D1'),
;    2898  1          DBM20  = uplit (%asciz'%N%A** Bad conn ID %O6%A received from %O6'),
;    2899  1          DBM21  = uplit (%asciz'%N%A** Message type %O2%A received in MSCP packet'),
;    2900  1          DBM22  = uplit (%asciz'%N%A** SEQUEN: RETPKT not available'),
;    2901  1          DBM23  = uplit (%asciz'%N%A** Error in SET_CTLR_CHAR'),
;    2902  1          DBM25  = uplit (%asciz'%N%A** Ctlr timeout = %D3%A. seconds'),
;    2903  1          DBM26  = uplit (%asciz'%N%A** Error in UNIT_INIT'),
;    2904  1          DBM27  = uplit (%asciz'%N%A** UNIT_INIT: RETPKT has bad ENDCODE'),
;    2905  1          DBM28A = uplit (%asciz'%N%A** Unit size (Lo) = %D5%A.'),
;    2906  1          DBM28B = uplit (%asciz'%N%A** Unit size (Hi) = %D5%A.'),
;    2907  1          DBM29  = uplit (%asciz'%N%A** ACCESS: RETPKT has bad ENDCODE'),
;    2908  1          DBM32  = uplit (%asciz'%N%A** QIO_UNIT: CST %D1%A no unit selected'),
;    2909  1          DBM101 = uplit (%asciz'%N%A** Unit # is: %O6'),
;    2910  1          DBM104 = uplit (%asciz'%N%A** Removable disk is selected'),
;    2911  1          DBM105 = uplit (%asciz'%N%A** Fixed disk is selected'),
;    2912  1          DBM107 = uplit (%asciz'%N%A** Illegal function: %O6'),
;    2913  1          DBM108 = uplit (%asciz'%N%A** Command ref # %O6%A/%O6%A (Oct) not sent by Host'),
;    2914  1          DBM109 = uplit (%asciz'%N%A** Unknown Error Log format %O3%A received'),
;    2915  1      !   DBM110 = uplit (%asciz'%N%A** Error-Log save area full'),
;    2916  1          DBM111 = uplit (%asciz'%N%A** Op-code %O3%A, End-code %O3%A for ref # %O6%A/%O6%A (8)'),
;    2917  1          DBM112 = uplit (%asciz'%N%A** Cmd-bc %O6%A/%O6%A Rsp-bc %O6%A/%O6%A for %O6%A/%O6%A (8)'),
;    2918  1          DBM120 = uplit (%asciz'%N%A** Response already received for cmd %O6%A/%O6%A (8)'),
;    2919  1          DBM121 = uplit (%asciz'%N%A** Failure to send command after # %O6%A/%O6%A (8)'),
;    2920  1      !
;    2921  1      ! DROP UNIT MESSAGES
;    2922  1      !
;    2923  1          DU_MSG = uplit (%asciz'%N%AUNIT%D2%A DROPPED - '),
;    2924  1          DU_RSN = uplit (
;    2925  1                   uplit (%asciz'%AUSER COMMAND%N'),
;    2926  1                   uplit (%asciz'%ACONFIGURATION ERROR%N'),
;    2927  1                   uplit (%asciz'%AINIT ERROR%N'),
;    2928  1                   uplit (%asciz'%ATRANSFER LIMIT REACHED%N'),
;    2929  1                   uplit (%asciz'%AERROR LIMIT REACHED%N'),
;    2930  1                   uplit (%asciz'%AUNRECOVERABLE DRIVE ERROR%N'),
;    2931  1                   uplit (%asciz'%AUNRECOVERABLE CONTROLLER ERROR%N'),
;    2932  1                   uplit (%asciz'%AFAILED TO COME ONLINE%N'),
;    2933  1                   uplit (%asciz'%AFAILED TO ACCESS EITHER FIRST OR LAST TRACK DURING INIT%N'),
;    2934  1                   uplit (%asciz'%ADISK WRITE PROTECTED%N'),
;    2935  1                   uplit (%asciz'%ACOMMAND TIME OUT%N')) : vector [11],
;    2936  1      !
;    2937  1      ! SYSTEM MESSAGES (PRINTF)
;    2938  1      !
;    2939  1          MSG_01 = uplit (%asciz'%N%APOWER DELAY - WAITING'),
;    2940  1          MSG_02 = uplit (%asciz'%N%AFUNCTIONAL TEST STARTED'),
;    2941  1          MSG_03 = uplit (%asciz'%N%N%AEXERCISER STARTED%N'),
;    2942  1      !
```

```
;    2943  1      ! REPORT MESSAGES (PRINTS)
;    2944  1      !
;    2945  1          RPT1  = uplit (%asciz'%N%N%AUNT DSK%S8%A# OF    # BYTES    # OF     # BYTES'),
;    2946  1          RPT2  = uplit (%asciz'%A  --HARD ERRORS-- --SOFT ERRORS--'),
;    2947  1          RPT3  = uplit (%asciz'%N%A #    #  TYPE  READS     READ   WRITES   WRITTEN'),
;    2948  1          RPT4  = uplit (%asciz'%A  SEK DAT DRV HST SEK DAT DRV HST'),
;    2949  1          RPT5  = uplit (%asciz'%N%A--- --- ----  -----  --------- ------  ---------'),
;    2950  1          RPT6  = uplit (%asciz'%A --- --- --- --- --- --- --- ---'),
;    2951  1          RPT7  = uplit (%asciz'%N%D2%D4%S2%T'),
;    2952  1          RPT8  = uplit (%asciz'%D4%Z3%D3%A,%Z3%A,%Z3'),
;    2953  1          RPT9  = uplit (%asciz'%D4%D4%D4%D4%D4%D4%D4%D4'),
;    2954  1          RPT10 = uplit (%asciz'%N%A .    .  CNTR      .  .........      .  .........'),
;    2955  1          RPT11 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;    2956  1          RPT12 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;    2957  1          RPT13 = UPLIT(%ASCIZ'%N%N%AUNIT  DISK            # OF   # BLKS       # OF    # BLKS '),
;    2958  1          RPT14 = UPLIT(%ASCIZ'%N%A  #     #     TYPE   READS     READ      WRITES  WRITTEN '),
;    2959  1          RPT15 = UPLIT(%ASCIZ'%N%A----  ----  -------  ------   ------     ------   ------  '),
;    2960  1          RPT16 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBN I/O  %D6%S3%D6%S5%D6%S3%D6'),
;    2961  1      !ZZZ    RPT17  = uplit (%asciz'%N%D2%D4%A  RD52'),
;    2962  1      !ZZZ    RPT18 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBNRD52  %D6%S3%D6%S5%D6%S3%D6'),
;    2963  1      !ZZZ    RPT19  = uplit (%asciz'%N%D2%D4%A  ????'),
;    2964  1
;    2965  1      !
;    2966  1      ! GENERAL ERROR MESSAGES
;    2967  1      !
;    2968  1      !       SYSTEM FATAL (ERRSF)
;    2969  1      !
;    2970  1          EGS_01 = uplit (%asciz'TOO MANY UNITS'),
;    2971  1          EGS_02 = uplit (%asciz'NOT ENOUGH FREE MEMORY FOR ALLOCATING READ/WRITE BUFFERS'),
;    2972  1      !
;    2973  1      !       DRIVE FATAL (ERRDF)
;    2974  1      !
;    2975  1          EGD_10 = uplit (%asciz'REGISTER EXISTENCE TEST FAILED'),
;    2976  1          EGD_11 = uplit (%asciz'VECTOR TEST FAILED'),
;    2977  1          EGD_12 = uplit (%asciz'BR LEVEL TEST FAILED'),
;    2978  1          EGD_13 = uplit (%asciz'INIT SEQUENCE FAILED'),
;    2979  1          EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
;    2980  1          EGD_15 = uplit (%asciz'ONLINE FAILED'),
;    2981  1          EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
;    2982  1          EGD_17 = uplit (%asciz'ACCESS FAILED'),
;    2983  1          EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
;    2984  1      !   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
;    2985  1          EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER'),
;    2986  1          EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND'),
;    2987  1          EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
;    2988  1          EGD_22 = uplit (%asciz'FAILED TO SEND ON-LINE COMMAND'),
;    2989  1          EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
;    2990  1          EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
;    2991  1      !
;    2992  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    2993  1      !
;    2994  1          EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
;    2995  1      !
```

```
;    2996  1      ! BASIC ERROR MESSAGES (PRINTB)
;    2997  1      !
;    2998  1      !       SYSTEM FATAL (ERRSF)
;    2999  1      !
;    3000  1          EBS_01 = uplit (%asciz'%AMORE THAN %D2%A UNITS SPECIFIED'),
;    3001  1      !
;    3002  1      !       DRIVE FATAL (ERRDF)
;    3003  1      !
;    3004  1          EBD_10 = uplit (%asciz'%A* NO RESPONSE AT ADDRESS %O6'),
;    3005  1          EBD_12 = uplit (%asciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
;    3006  1          EBD_13 = uplit (%asciz'%A* STEP %D1%A READ ERROR'),
;    3007  1          EBD_14 = uplit (%asciz'%A* BAD SA CODE FROM DRIVE %O6'),
;    3008  1          EBD_18 = uplit (%asciz'%A* DISK%D2%A WENT OFFLINE'),
;    3009  1          EBD_19 = uplit (%asciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
;    3010  1          EBD_24 = uplit (%asciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
;    3011  1      !
;    3012  1      !
;    3013  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    3014  1      !
;    3015  1      EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE'),                    !ZZZ
;    3016  1      EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),                   !ZZZ
;    3017  1      EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),                  !ZZZ
;    3018  1      EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),                   !ZZZ
;    3019  1      EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                             !ZZZ
;    3020  1      EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                          !ZZZ
;    3021  1      EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                         !ZZZ
;    3022  1      EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                          !ZZZ
;    3023  1      EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),                     !ZZZ
;    3024  1      EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                         !ZZZ
;    3025  1      EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                        !ZZZ
;    3026  1      EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),                    !ZZZ
;    3027  1      EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'),      !ZZZ
;    3028  1      !
;    3029  1          ERR_00  = uplit (%asciz'%A* DISK%D2'),
;    3030  1          ERR_COD = uplit (
;    3031  1                    uplit (%asciz'%AINVALID COMMAND'),
;    3032  1                    uplit (%asciz'%ACOMMAND ABORTED'),
;    3033  1                    uplit (%asciz'%AUNIT OFFLINE'),
;    3034  1                    uplit (%asciz'%ATRANSITION TO AVAILABLE STATE'),
;    3035  1                    uplit (%asciz'%AMEDIA FORMAT ERROR'),
;    3036  1                    uplit (%asciz'%AWRITE-PROTECTED'),
;    3037  1                    uplit (%asciz'%ADEVICE COMPARE ERROR'),
;    3038  1                    uplit (%asciz'%ADATA ERROR'),
;    3039  1                    uplit (%asciz'%AHOST BUFFER ACCESS ERROR'),
;    3040  1                    uplit (%asciz'%ACONTROLLER ERROR'),
;    3041  1                    uplit (%asciz'%ADRIVE ERROR'),
;    3042  1                    uplit (%asciz'%AMESSAGE FROM INTERNAL DIAGNOSTICS'),
;    3043  1                    uplit (%asciz'%AHOST COMPARE ERROR'),
;    3044  1                    uplit (%asciz'%ACOMMAND TIMEOUT')) : vector [14],
;    3045  1      !
;    3046  1      !       ERROR LOG MESSAGE (ERRSOFT)
;    3047  1      !
;    3048  1          ELG_00  = uplit (%asciz'%AERROR LOG MESSAGE RECEIVED:%N'),
```

```
;    3049  1          ELG_FMT = uplit (
;    3050  1                    uplit (%asciz'%A* CONTROLLER ERROR%N'),
;    3051  1                    uplit (%asciz'%A* HOST MEMORY ACCESS ERROR%N'),
;    3052  1                    uplit (%asciz'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
;    3053  1                    uplit (%asciz'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
;    3054  1                    uplit (%asciz'%A* DISK%D2%A - "SMALL DISK" ERROR%N')) : vector [5],
;    3055  1          !
;    3056  1          ! EXTENDED ERROR MESSAGES (PRINTX)
;    3057  1          !
;    3058  1              EX_SA  = uplit (%asciz'%N%A* SA: %O6'),
;    3059  1              EX_SC  = uplit (%asciz'%N%A* STATUS CODE: %O2'),
;    3060  1              EX_SBO = uplit (%asciz'%O4'),
;    3061  1              EX_SB  = uplit (%asciz'%N%A* SUB_CODE: '),
;    3062  1              EX_CMD = uplit (%asciz'%N%A* COMMAND: '),
;    3063  1              EX_RD  = uplit (%asciz'%AREAD'),
;    3064  1              EX_WRT = uplit (%asciz'%AWRITE'),
;    3065  1              EX_CMP = uplit (%asciz'%A-COMPARE'),
;    3066  1              EX_ONL = uplit (%asciz'%AONLINE'),
;    3067  1              EX_ACC = uplit (%asciz'%AACCESS'),
;    3068  1              EX_OP  = uplit (%asciz'%O3'),
;    3069  1              EX_BB  = uplit (%asciz'%N%A* BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
;    3070  1              EX_BB1 = uplit (%asciz'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A. (OCT %O6%A)'),
;    3071  1              EX_BBU = uplit (%asciz'%N%A* BAD BLOCK REPORTED (Replaced): %D%A. (OCT %O6%A)'),
;    3072  1              EX_LBN = uplit (%asciz'%N%A* LBN: %D5%A. (OCT %O6%A)'),
;    3073  1              EX_PBN = uplit (%asciz'%N%A* PBN: %D5%A. (OCT %O6%A)'),
;    3074  1              EX_LBR = uplit (%asciz'%N%A* LBN: (READ) %D5%A. (OCT %O6%A)'),
;    3075  1              EX_LBW = uplit (%asciz'%N%A* LBN: (WRITE) %D5%A. (OCT %O6%A)'),
;    3076  1              EX_RBN = uplit (%asciz'%N%A* REPLACEMENT BLOCK NO. %D5%A. (OCT %O6%A)'),
;    3077  1              EX_CBC = uplit (%asciz'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
;    3078  1              EX_CBR = uplit (%asciz'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
;    3079  1              EX_CBW = uplit (%asciz'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
;    3080  1              EX_BC  = uplit (%asciz'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
;    3081  1              EX_BD  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
;    3082  1              EX_BDR = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
;    3083  1              EX_BDW = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
;    3084  1              EX_RP  = uplit (%asciz'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
;    3085  1              EX_WRD = uplit (%asciz'%A %O6'),
;    3086  1              EX_TIM = uplit (%asciz'%N%A* TIME: %Z2%A:%Z2%A HOURS%N'),
;    3087  1
;    3088  1
;    3089  1              XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                          !ZZZ
;    3090  1              XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A. (OCT %O6%A)'),                !ZZZ
;    3091  1              XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                       !ZZZ
;    3092  1              XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),              !ZZZ
;    3093  1              XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),    !ZZZ
;    3094  1
;    3095  1          !
;    3096  1          ! CONFIGURATION ERROR MESSAGES (PRINTF)
;    3097  1          !
;    3098  1              CER_01 = uplit (%asciz'%N%ADUPLICATE UNIT:%D2%A AT IP: %O6'),
;    3099  1              CER_02 = uplit (%asciz'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
;    3100  1          !
;    3101  1          ! ERROR/EVENT SUB CODES (PRINTX)
```

```
;    3102  1        !
;    3103  1            SC_SDI = uplit (%asciz'%ASPIN-DOWN IGNORED'),
;    3104  1            SC_CON = uplit (%asciz'%ASTILL CONNECTED'),
;    3105  1            SC_DUP = uplit (%asciz'%ADUPLICATE UNIT NUMBER'),
;    3106  1            SC_ONL = uplit (%asciz'%AALREADY ONLINE'),
;    3107  1            SC_SON = uplit (%asciz'%ASTILL ONLINE'),
;    3108  1            SC_UNK = uplit (%asciz'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
;    3109  1            SC_VOL = uplit (%asciz'%ANO VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
;    3110  1            SC_IOP = uplit (%asciz'%AUNIT INOPERATIVE (RD51/52 write fault)'),
;    3111  1            SC_DIS = uplit (%asciz'%A'UNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS'),
;    3112  1            SC_FER = uplit (%asciz'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
;    3113  1            SC_FE2 = uplit (%asciz'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER'),
;    3114  1            SC_ISH = uplit (%asciz'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
;    3115  1            SC_IS2 = uplit (%asciz'%AHEADER COMPARE ERROR (Valid header not found)'),
;    3116  1            SC_DST = uplit (%asciz'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
;    3117  1            SC_DS2 = uplit (%asciz'%ADATA SYNC NOT FOUND (Data sync timeout)'),
;    3118  1            SC_ECC = uplit (%asciz'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
;    3119  1            SC_ECD = uplit (%asciz'%AUNCORRECTABLE ECC ERROR'),
;    3120  1            SC_RCT = uplit (%asciz'%ARCT CORRUPTED'),
;    3121  1            SC_FUL = uplit (%asciz'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
;    3122  1            SC_576 = uplit (%asciz'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
;    3123  1            SC_FCT = uplit (%asciz'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
;    3124  1            SC_EC1 = uplit (%asciz'%AONE SYMBOL ECC ERROR'),
;    3125  1            SC_EC2 = uplit (%asciz'%ATWO SYMBOL ECC ERROR'),
;    3126  1            SC_EC3 = uplit (%asciz'%ATHREE SYMBOL ECC ERROR'),
;    3127  1            SC_EC4 = uplit (%asciz'%AFOUR SYMBOL ECC ERROR'),
;    3128  1            SC_EC5 = uplit (%asciz'%AFI'E SYMBOL ECC ERROR'),
;    3129  1            SC_EC6 = uplit (%asciz'%ASIX SYMBOL ECC ERROR'),
;    3130  1            SC_EC7 = uplit (%asciz'%ASEVEN SYMBOL ECC ERROR'),
;    3131  1            SC_EC8 = uplit (%asciz'%AEIGHT SYMBOL ECC ERROR'),
;    3132  1            SC_EC9 = uplit (%asciz'%ACORRECTABLE ERROR IN ECC FIELD'),
;    3133  1            SC_SWP = uplit (%asciz'%AUNIT SOFTWARE WRITE PROTECTED'),
;    3134  1            SC_HWP = uplit (%asciz'%AUNIT HARDWARE WRITE PROTECTED'),
;    3135  1            SC_ODA = uplit (%asciz'%AODD TRANSFER ADDRESS'),
;    3136  1            SC_ODB = uplit (%asciz'%AODD BYTE COUNT'),
;    3137  1            SC_NXM = uplit (%asciz'%ANON-EXISTENT HOST MEMORY'),
;    3138  1            SC_PAR = uplit (%asciz'%AHOST MEMORY PARITY ERROR'),
;    3139  1            SC_CTO = uplit (%asciz'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
;    3140  1            SC_SDS = uplit (%asciz'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
;    3141  1            SC_EDC = uplit (%asciz'%A"ERROR DETECTION CODE" ERROR'),
;    3142  1            SC_IDS = uplit (%asciz'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
;    3143  1            SC_SRT = uplit (%asciz'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete)'),
;    3144  1            SC_SRI = uplit (%asciz'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
;    3145  1            SC_POE = uplit (%asciz'%APOSITION ERROR (Mis-seek)'),
;    3146  1            SC_RDY = uplit (%asciz'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
;    3147  1            SC_CLK = uplit (%asciz'%ADRIVE CLOCK DROPOUT'),
;    3148  1            SC_RSP = uplit (%asciz'%ALOST RECEIVER READY BETWEEN SECTORS'),
;    3149  1            SC_SUR = uplit (%asciz'%ADRIVE DETECTED ERROR'),
;    3150  1            SC_PSP = uplit (%asciz'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR'),
;    3151  1        !
;    3152  1        ! CONTROLLER GENERIC ERROR CODES
;    3153  1        !
;    3154  1            CNTR_ERR = uplit (
```

```
;    3155  1                    uplit (%asciz'%ACONTROLLER TIMEOUT'),
;    3156  1                    uplit (%asciz'%AENVELOPE/PACKET READ ERROR (Parity or timeout)'),
;    3157  1                    uplit (%asciz'%AENVELOPE/PACKET WRITE ERROR (Parity or timeout)'),
;    3158  1                    uplit (%asciz'%ACONTROLLER ROM AND RAM PARITY ERROR'),
;    3159  1                    uplit (%asciz'%ACONTROLLER RAM PARITY ERROR'),
;    3160  1                    uplit (%asciz'%ACONTROLLER ROM PARITY ERROR'),
;    3161  1                    uplit (%asciz'%ARING READ ERROR (Parity or timeout)'),
;    3162  1                    uplit (%asciz'%ARING WRITE ERROR (Parity or timeout)'),
;    3163  1                    uplit (%asciz'INTERRUPT MASTER FAILURE'),
;    3164  1                    uplit (%asciz'%AHOST ACCESS TIMEOUT (Higher level protocol dependent)'),
;    3165  1                    uplit (%asciz'%ACREDIT LIMIT EXCEEDED'),
;    3166  1                    uplit (%asciz'%AQ-BUS MASTER ERROR'),
;    3167  1                    uplit (%asciz'%ACONTROLLER FATAL ERROR'),
;    3168  1                    uplit (%asciz'%AINSTRUCTION LOOP TIMEOUT'),
;    3169  1                    uplit (%asciz'%AILLEGAL VIRTUAL CIRCUIT ID'),
;    3170  1                    uplit (%asciz'%AINTERRUPT VECTOR ILLEGAL'),
;    3171  1                    uplit (%asciz'%AMAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
;    3172  1                    uplit (%asciz'%AMAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
;    3173  1                    uplit (%asciz'%ACONTROLLER RAM ERROR (Non-parity)'),
;    3174  1                    uplit (%asciz'%AINIT SEQUENCE ERROR'),
;    3175  1                    uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
;    3176  1                    uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
;    3177  1                    uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
;    3178  1            !
;    3179  1            ! RD/RX CONTROLLER DEPENDENT ERRORS CODES
;    3180  1            !
;    3181  1                RDRX_ERR = uplit (
;    3182  1                           uplit (%asciz'%AT11 CPU FAILURE'),
;    3183  1                           uplit (%asciz'%ANON-PARITY RAM ERROR'),
;    3184  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
;    3185  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
;    3186  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
;    3187  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
;    3188  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) : vector [7],
;    3189  1            !
;    3190  1            ! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
;    3191  1            !
;    3192  1                DF_MSG  = uplit (%asciz'%N%AZRQA DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3193  1                HRD_MSG = uplit (%asciz'%N%AZRQA HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3194  1                SFT_MSG = uplit (%asciz'%N%AZRQA SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06%N'),
;    3195  1                HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),
;    3196  1
;    3197  1
;    3198  1            !
;    3199  1            ! MISCELLANEOUS
;    3200  1            !
;    3201  1                SPACE4   = uplit (%asciz'%S4'),
;    3202  1                CRLF     = uplit (%asciz'%N'),
;    3203  1                DASH     = uplit (%asciz'%A - '),
;    3204  1                ASTERISK = uplit (%asciz'%A* ');
```

```
;    3205  1      %sbttl 'DEFAULT HARDWARE P-TABLE'
;    3206  1
;    3207  1      !+
;    3208  1      ! THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
;    3209  1      ! THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
;    3210  1      ! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
;    3211  1      ! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
;    3212  1      !-
;    3213  1
;    3214  1      BGNHW (DFPTBL);
;    3215  1
;    3216  1      global
;    3217  1          HWPT_IP_ADDR : word initial (INIT_IP_ADDR),          ! IP ADDRESS
;    3218  1          HWPT_VECTOR : word initial (INIT_INTR_VECT),         ! VECTOR ADDRESS
;    3219  1          HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),        ! BR LEVEL
;    3220  1          HWPT_DISK : WORD INITIAL (%o'000340'),               !PROTECT, WHOLE DISK, DUP WT, RUN DUP   ZZZ
;    3221  1                                                               !DUP WT, DK 0          ZZZ
;    3222  1          HWPTS0_LBN : word initial (0),                       ! STARTING TRACK LO ZZZ
;    3223  1          HWPTS1_LBN : word initial (0),                       ! STARTING TRACK HI ZZZ
;    3224  1          HWPTE0_LBN : word initial (%o'177777'),              ! ENDING TRACK LO   ZZZ
;    3225  1          HWPTE1_LBN : word initial (0),                       ! ENDING TRACK HI   ZZZ
;    3226  1          NAME_LO    : WORD INITIAL (%o'020040'),              !DISK TYPE          ZZZ
;    3227  1          NAME_HI    : WORD INITIAL (%o'020040');              !DISK TYPE          ZZZ
;    3228  1
;    3229  1      ENDHW;
```

```
;    3230  1     %sbttl 'SOFTWARE P-TABLE'
;    3231  1
;    3232  1     !+
;    3233  1     ! THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;    3234  1     ! PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;    3235  1     ! SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;    3236  1     ! AT RUN TIME.
;    3237  1     !-
;    3238  1
;    3239  1     BGNSW (SFPTBL);
;    3240  1
;    3241  1     global
;    3242  1         SWP_ERROR : word initial (32),                              ! HARD ERROR LIMIT FOR DROPPING UNIT
;    3243  1         SWP_XFER : WORD INITIAL (0),                                !XFER LIMIT. DEFAULT = QUICK PASS       !ZZZ
;    3244  1         SWP_FLAGS : word initial (SWF_RDM or SWF_CRC or SWF_HWC or SWF_FER or SWF_HRD or SWF_BLK),  ! FLAGS (SEE DOCUMEN
TATION)
;    3245  1         SWP_DPAT : word initial (0),                                ! DATA PATTERN NUMBER
;    3246  1         SWP_RAT : word initial (99),                                ! RD51/52 OPERATION RATIO
;    3247  1         SWP_TIME : word initial (0),                                ! START TIME (HHMM)
;    3248  1         DUPROUND : WORD INITIAL (11),                 !NO OF I/Os PER DBN TEST ZZZ
;    3249  1
;    3250  1     !       THE NEXT TWO LOCATIONS SHOULD BE TOGETHER
;    3251  1
;    3252  1         SWP_UCNT : word initial (MAX_UDP_CNT),                      ! USER DATA PATTERN COUNT
;    3253  1         SWP_UDPAT : vector [MAX_UDP_CNT, word];                     ! USER DATA PATTERN
;    3254  1
;    3255  1     ENDSW;
```

```
;    3256  1     %sbttl 'PROTECTION TABLE'
;    3257  1
;    3258  1     !+
;    3259  1     ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;    3260  1     ! TO PROTECT THE LOAD MEDIA.
;    3261  1     !-
;    3262  1
;    3263  1     BGNPROT (0, -1, 6);
;    3264  1
;    3265  1     !1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;    3266  1     !2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;    3267  1     !3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;    3268  1
;    3269  1     ENDPROT;
;    3270  1     end
;    3271  1
;    3272  0     eludom


                                        .TITLE  ZRQAM1 RD/RX EXERCISER
                                        .IDENT  /V01.8/
                                        .ENABL  AMA


000000                                  .PSECT  $CODE$,  RO
000000      132     122     121 L$NAME::.ASCII  /ZRQ/
000003      101                         .ASCII  /A/
000004      000                         .BYTE   0
000005      000                         .BYTE   0
000006      000                         .BYTE   0
000007      000                         .BYTE   0
000010                          L$REV::
000010      105                         .ASCII  /E/
000011      060                         .ASCII  /0/
000012  000000G                 L$UNIT::.WORD   T$PTHV
000014  076400                  L$TIML::.WORD   76400
000016  000000G                 L$HPCP::.WORD   L$HARD
000020  000000G                 L$SPCP::.WORD   L$SOFT
000022  022656'                  L$HPTP::.WORD   L$HW
000024  022706'                  L$SPTP::.WORD   L$SW
000026  000000G                 L$LADP::.WORD   L$LAST
000030  000000                  L$STA:: .WORD   0
000032  000000                  L$CO::  .WORD   0
000034  000001                  L$DTYP::.WORD   1
000036  000000                  L$APT:: .WORD   0
000040  000124'                 L$DTP:: .WORD   L$DISPATCH
000042  000000                  L$PRIO::.WORD   0
000044  000000                  L$ENVI::.WORD   0
000046  000000                  L$EXP1::.WORD   0
000050                          L$MREV::
000050      003                         .BYTE   3
000051      003                         .BYTE   3
000052  000000                  L$EF::  .WORD   0
```

```
000054  000000                          .WORD   0
000056  000000                  L$SPC:: .WORD   0
000060  000000G                 L$DEVP::.WORD   L$DVTYP
000062  000000G                 L$REPP::.WORD   L$RPT
000064  000000                  L$EXP4::.WORD   0
000066  000000                  L$EXP5::.WORD   0
000070  000000G                 L$AUT:: .WORD   L$AU
000072  000000G                 L$DUT:: .WORD   L$DU
000074  000000                  L$LUN:: .WORD   0
000076  000000G                 L$DESP::.WORD   L$DESC
000100  104035                  L$LOAD::.WORD   -73743
000102  000126'                 L$ETP:: .WORD   L$ERRTBL
000104  000000G                 L$ICP:: .WORD   L$INIT
000106  000000G                 L$CCP:: .WORD   L$CLEAN
000110  000000G                 L$ACP:: .WORD   L$AUTO
000112  022770'                 L$PRT:: .WORD   L$PROT
000114  000000                  L$TEST::.WORD   0
000116  000000                  L$DLY:: .WORD   0
000120  000000                  L$HIME::.WORD   0
000122  000001                  D$PCNT::.WORD   1
000124  000000G                 L$DISPATCH::
                                        .WORD   T1
000126                          ERRTYP::.BLKW   1
000130                          ERRNBR::.BLKW   1
000132                          ERRMSG::.BLKW   1
000134                          ERRBLK::.BLKW   1
000136  111  120  040           P.AAA:  .ASCII  /IP /
000141  141  144  144                   .ASCII  /add/
000144  162  145  163                   .ASCII  /res/
000147  163  000  000                   .ASCII  /s/<00><00>
000152  126  145  143           P.AAB:  .ASCII  /Vec/
000155  164  157  162                   .ASCII  /tor/
000160  000  000                         .ASCII  <00><00>
000162  102  122  040           P.AAC:  .ASCII  /BR /
000165  114  105  126                   .ASCII  /LEV/
000170  105  114  040                   .ASCII  /EL /
000173  133  125  123                   .ASCII  /[US/
000176  125  101  114                   .ASCII  /UAL/
000201  114  131  040                   .ASCII  /LY /
000204  064  055  122                   .ASCII  /4-R/
000207  121  104  130                   .ASCII  /QDX/
000212  061  040  065                   .ASCII  /1 5/
000215  055  122  125                   .ASCII  /-RU/
000220  130  065  060                   .ASCII  /X50/
000223  135  000  000                   .ASCII  /]/<00><00>
000226  104  162  151           P.AAD:  .ASCII  /Dri/
000231  166  145  040                   .ASCII  /ve /
000234  156  165  155                   .ASCII  /num/
000237  142  145  162                   .ASCII  /ber/
000242  000  000                         .ASCII  <00><00>
000244  124  145  163           P.AAE:  .ASCII  /Tes/
000247  164  040  145                   .ASCII  /t e/
000252  156  164  151                   .ASCII  /nti/
```

```
000255  162  145  040                    .ASCII  /re /
000260  143  165  163                    .ASCII  /cus/
000263  164  157  155                    .ASCII  /tom/
000266  145  162  040                    .ASCII  /er /
000271  141  162  145                    .ASCII  /are/
000274  141  040  157                    .ASCII  /a o/
000277  146  040  164                    .ASCII  /f t/
000302  150  151  163                    .ASCII  /his/
000305  040  144  151                    .ASCII  / di/
000310  163  153  000                    .ASCII  /sk/<00>
000313  000                             .ASCII  <00>
000314  114  157  167          P.AAF:    .ASCII  /Low/
000317  145  162  040                    .ASCII  /er /
000322  157  143  164                    .ASCII  /oct/
000325  141  154  040                    .ASCII  /al /
000330  167  157  162                    .ASCII  /wor/
000333  144  040  157                    .ASCII  /d o/
000336  146  040  142                    .ASCII  /f b/
000341  145  147  151                    .ASCII  /egi/
000344  156  156  151                    .ASCII  /nni/
000347  156  147  040                    .ASCII  /ng /
000352  114  102  116                    .ASCII  /LBN/
000355  040  141  144                    .ASCII  / ad/
000360  144  162  145                    .ASCII  /dre/
000363  163  163  000                    .ASCII  /ss/<00>
000366  110  151  147          P.AAG:    .ASCII  /Hig/
000371  150  145  162                    .ASCII  /her/
000374  040  157  143                    .ASCII  / oc/
000377  164  141  154                    .ASCII  /tal/
000402  040  167  157                    .ASCII  / wo/
000405  162  144  040                    .ASCII  /rd /
000410  157  146  040                    .ASCII  /of /
000413  142  145  147                    .ASCII  /beg/
000416  151  156  156                    .ASCII  /inn/
000421  151  156  147                    .ASCII  /ing/
000424  040  114  102                    .ASCII  / LB/
000427  116  040  141                    .ASCII  /N a/
000432  144  144  162                    .ASCII  /ddr/
000435  145  163  163                    .ASCII  /ess/
000440  000  000                        .ASCII  <00><00>
000442  114  157  167          P.AAH:    .ASCII  /Low/
000445  145  162  040                    .ASCII  /er /
000450  157  143  164                    .ASCII  /oct/
000453  141  154  040                    .ASCII  /al /
000456  167  157  162                    .ASCII  /wor/
000461  144  040  157                    .ASCII  /d o/
000464  146  040  145                    .ASCII  /f e/
000467  156  144  151                    .ASCII  /ndi/
000472  156  147  040                    .ASCII  /ng /
000475  114  102  116                    .ASCII  /LBN/
000500  040  141  144                    .ASCII  / ad/
000503  144  162  145                    .ASCII  /dre/
000506  163  163  000                    .ASCII  /ss/<00>
```

```
000511  000                             .ASCII  <00>
000512  110     151     147     P.AAI:  .ASCII  /Hig/
000515  150     145     162             .ASCII  /her/
000520  040     157     143             .ASCII  / oc/
000523  164     141     154             .ASCII  /tal/
000526  040     167     157             .ASCII  / wo/
000531  162     144     040             .ASCII  /rd /
000534  157     146     040             .ASCII  /of /
000537  145     156     144             .ASCII  /end/
000542  151     156     147             .ASCII  /ing/
000545  040     114     102             .ASCII  / LB/
000550  116     040     141             .ASCII  /N a/
000553  144     144     162             .ASCII  /ddr/
000556  145     163     163             .ASCII  /ess/
000561  000                             .ASCII  <00>
000562  127     162     151     P.AAJ:  .ASCII  /Wri/
000565  164     145     040             .ASCII  /te /
000570  157     156     040             .ASCII  /on /
000573  143     165     163             .ASCII  /cus/
000576  164     157     155             .ASCII  /tom/
000601  145     162     040             .ASCII  /er /
000604  144     141     164             .ASCII  /dat/
000607  141     040     141             .ASCII  /a a/
000612  162     145     141             .ASCII  /rea/
000615  040     157     146             .ASCII  / of/
000620  040     164     150             .ASCII  / th/
000623  151     163     040             .ASCII  /is /
000626  144     151     163             .ASCII  /dis/
000631  153     000     000             .ASCII  /k/<00><00>
000634  052     052     040     P.AAK:  .ASCII  /** /
000637  127     101     122             .ASCII  /WAR/
000642  116     111     116             .ASCII  /NIN/
000645  107     040     055             .ASCII  /G -/
000650  040     103     125             .ASCII  / CU/
000653  123     124     117             .ASCII  /STO/
000656  115     105     122             .ASCII  /MER/
000661  040     104     101             .ASCII  / DA/
000664  124     101     040             .ASCII  /TA /
000667  101     122     105             .ASCII  /ARE/
000672  101     040     115             .ASCII  /A M/
000675  101     131     040             .ASCII  /AY /
000700  102     105     040             .ASCII  /BE /
000703  117     126     105             .ASCII  /OVE/
000706  122     127     122             .ASCII  /RWR/
000711  111     124     124             .ASCII  /ITT/
000714  105     116     041             .ASCII  /EN!/
000717  040     056     056             .ASCII  / ../
000722  056     040     103             .ASCII  /. C/
000725  117     116     106             .ASCII  /ONF/
000730  111     122     115             .ASCII  /IRM/
000733  000                             .ASCII  <00>
000734  101     154     163     P.AAL:  .ASCII  /Als/
000737  157     040     162             .ASCII  /o r/
```

```
000742  165  156  040                .ASCII  /un /
000745  104  125  120                .ASCII  /DUP/
000750  040  145  170                .ASCII  / ex/
000753  145  162  143                .ASCII  /erc/
000756  151  163  145                .ASCII  /ise/
000761  162  000  000                .ASCII  /r/<00><00>
000764  127  162  151      P.AAM:    .ASCII  /Wri/
000767  164  145  040                .ASCII  /te /
000772  157  156  040                .ASCII  /on /
000775  144  151  141                .ASCII  /dia/
001000  147  156  157                .ASCII  /gno/
001003  163  164  151                .ASCII  /sti/
001006  143  040  141                .ASCII  /c a/
001011  162  145  141                .ASCII  /rea/
001014  000  000                     .ASCII  <00><00>
001016  110  141  162      P.AAN:    .ASCII  /Har/
001021  144  040  105                .ASCII  /d E/
001024  162  162  157                .ASCII  /rro/
001027  162  040  154                .ASCII  /r l/
001032  151  155  151                .ASCII  /imi/
001035  164  000  000                .ASCII  /t/<00><00>
001040  124  162  141      P.AAO:    .ASCII  /Tra/
001043  156  163  146                .ASCII  /nsf/
001046  145  162  040                .ASCII  /er /
001051  154  151  155                .ASCII  /lim/
001054  151  164  040                .ASCII  /it /
001057  151  156  040                .ASCII  /in /
001062  155  145  147                .ASCII  /meg/
001065  141  142  171                .ASCII  /aby/
001070  164  145  163                .ASCII  /tes/
001073  040  050  060                .ASCII  / (0/
001076  040  146  157                .ASCII  / fo/
001101  162  040  042                .ASCII  /r "/
001104  121  165  151                .ASCII  /Qui/
001107  143  153  040                .ASCII  /ck /
001112  160  141  163                .ASCII  /pas/
001115  163  042  051                .ASCII  /s")/
001120  000  000                     .ASCII  <00><00>
001122  122  141  156      P.AAP:    .ASCII  /Ran/
001125  144  157  155                .ASCII  /dom/
001130  040  163  145                .ASCII  / se/
001133  145  153  040                .ASCII  /ek /
001136  155  157  144                .ASCII  /mod/
001141  145  000  000                .ASCII  /e/<00><00>
001144  122  145  141      P.AAQ:    .ASCII  /Rea/
001147  144  055  143                .ASCII  /d-c/
001152  157  155  160                .ASCII  /omp/
001155  141  162  145                .ASCII  /are/
001160  163  040  160                .ASCII  /s p/
001163  145  162  146                .ASCII  /erf/
001166  157  162  155                .ASCII  /orm/
001171  145  144  040                .ASCII  /ed /
001174  141  164  040                .ASCII  /at /
```

```
001177  164  150  145          .ASCII  /the/
001202  040  103  157          .ASCII  / Co/
001205  156  164  162          .ASCII  /ntr/
001210  157  154  154          .ASCII  /oll/
001213  145  162  000          .ASCII  /er/<00>
001216  127  162  151  P.AAR:   .ASCII  /Wri/
001221  164  145  055          .ASCII  /te-/
001224  143  157  155          .ASCII  /com/
001227  160  141  162          .ASCII  /par/
001232  145  163  040          .ASCII  /es /
001235  160  145  162          .ASCII  /per/
001240  146  157  162          .ASCII  /for/
001243  155  145  144          .ASCII  /med/
001246  040  141  164          .ASCII  / at/
001251  040  164  150          .ASCII  / th/
001254  145  040  103          .ASCII  /e C/
001257  157  156  164          .ASCII  /ont/
001262  162  157  154          .ASCII  /rol/
001265  154  145  162          .ASCII  /ler/
001270  000  000               .ASCII  <00><00>
001272  103  150  145  P.AAS:   .ASCII  /Che/
001275  143  153  040          .ASCII  /ck /
001300  141  154  154          .ASCII  /all/
001303  040  127  162          .ASCII  / Wr/
001306  151  164  145          .ASCII  /ite/
001311  163  040  141          .ASCII  /s a/
001314  164  040  110          .ASCII  /t H/
001317  157  163  164          .ASCII  /ost/
001322  040  142  171          .ASCII  / by/
001325  040  162  145          .ASCII  / re/
001330  141  144  151          .ASCII  /adi/
001333  156  147  000          .ASCII  /ng/<00>
001336  125  163  145  P.AAT:   .ASCII  /Use/
001341  162  055  144          .ASCII  /r-d/
001344  145  146  151          .ASCII  /efi/
001347  156  145  144          .ASCII  /ned/
001352  040  144  141          .ASCII  / da/
001355  164  141  040          .ASCII  /ta /
001360  160  141  164          .ASCII  /pat/
001363  164  145  162          .ASCII  /ter/
001366  156  000               .ASCII  /n/<00>
001370  123  145  154  P.AAU:   .ASCII  /Sel/
001373  145  143  164          .ASCII  /ect/
001376  040  160  162          .ASCII  / pr/
001401  145  055  144          .ASCII  /e-d/
001404  145  146  151          .ASCII  /efi/
001407  156  145  144          .ASCII  /ned/
001412  040  144  141          .ASCII  / da/
001415  164  141  040          .ASCII  /ta /
001420  160  141  164          .ASCII  /pat/
001423  164  145  162          .ASCII  /ter/
001426  156  040  050          .ASCII  /n (/
001431  060  040  146          .ASCII  /0 f/
```

```
001434  157  162  040                .ASCII  /or /
001437  123  145  161                .ASCII  /Seq/
001442  165  145  156                .ASCII  /uen/
001445  164  151  141                .ASCII  /tia/
001450  154  040  163                .ASCII  /l s/
001453  145  154  145                .ASCII  /ele/
001456  143  164  151                .ASCII  /cti/
001461  157  156  051                .ASCII  /on)/
001464  000  000                    .ASCII  <00><00>
001466  116  165  155       P.AAV:   .ASCII  /Num/
001471  142  145  162                .ASCII  /ber/
001474  040  157  146                .ASCII  / of/
001477  040  167  157                .ASCII  / wo/
001502  162  144  163                .ASCII  /rds/
001505  040  151  156                .ASCII  / in/
001510  040  144  141                .ASCII  / da/
001513  164  141  040                .ASCII  /ta /
001516  160  141  164                .ASCII  /pat/
001521  164  145  162                .ASCII  /ter/
001524  156  040  050                .ASCII  /n (/
001527  061  066  040                .ASCII  /16 /
001532  155  141  170                .ASCII  /max/
001535  151  155  165                .ASCII  /imu/
001540  155  051  000                .ASCII  /m)/<00>
001543  000                         .ASCII  <00>
001544  120  141  164       P.AAW:   .ASCII  /Pat/
001547  164  145  162                .ASCII  /ter/
001552  156  040  166                .ASCII  /n v/
001555  141  154  165                .ASCII  /alu/
001560  145  040  050                .ASCII  /e (/
001563  116  157  040                .ASCII  /No /
001566  154  145  141                .ASCII  /lea/
001571  144  151  156                .ASCII  /din/
001574  147  040  172                .ASCII  /g z/
001577  145  162  157                .ASCII  /ero/
001602  163  040  141                .ASCII  /s a/
001605  154  154  157                .ASCII  /llo/
001610  167  145  144                .ASCII  /wed/
001613  051  000  000                .ASCII  /)/<00><00>
001616  103  154  145       P.AAX:   .ASCII  /Cle/
001621  141  162  040                .ASCII  /ar /
001624  163  164  141                .ASCII  /sta/
001627  164  151  163                .ASCII  /tis/
001632  164  151  143                .ASCII  /tic/
001635  141  154  040                .ASCII  /al /
001640  164  141  142                .ASCII  /tab/
001643  154  145  163                .ASCII  /les/
001646  040  141  146                .ASCII  / af/
001651  164  145  162                .ASCII  /ter/
001654  040  160  162                .ASCII  / pr/
001657  151  156  164                .ASCII  /int/
001662  151  156  147                .ASCII  /ing/
001665  000                         .ASCII  <00>
```

```
001666  120  145  162        P.AAY:  .ASCII  /Per/
001671  143  145  156                .ASCII  /cen/
001674  164  141  147                .ASCII  /tag/
001677  145  040  157                .ASCII  /e o/
001702  146  040  042                .ASCII  /f "/
001705  106  151  170                .ASCII  /Fix/
001710  145  144  040                .ASCII  /ed /
001713  104  151  163                .ASCII  /Dis/
001716  153  042  040                .ASCII  /k" /
001721  157  160  145                .ASCII  /ope/
001724  162  141  164                .ASCII  /rat/
001727  151  157  156                .ASCII  /ion/
001732  163  040  157                .ASCII  /s o/
001735  165  164  040                .ASCII  /ut /
001740  157  146  040                .ASCII  /of /
001743  164  157  164                .ASCII  /tot/
001746  141  154  040                .ASCII  /al /
001751  157  160  145                .ASCII  /ope/
001754  162  141  164                .ASCII  /rat/
001757  151  157  156                .ASCII  /ion/
001762  163  000                    .ASCII  /s/<00>
001764  125  156  151        P.AAZ:  .ASCII  /Uni/
001767  164  163  040                .ASCII  /ts /
001772  164  157  040                .ASCII  /to /
001775  142  145  040                .ASCII  /be /
002000  163  145  154                .ASCII  /sel/
002003  145  143  164                .ASCII  /ect/
002006  145  144  040                .ASCII  /ed /
002011  141  164  040                .ASCII  /at /
002014  162  141  156                .ASCII  /ran/
002017  144  157  155                .ASCII  /dom/
002022  040  050  116                .ASCII  / (N/
002025  157  054  040                .ASCII  /o, /
002030  151  155  160                .ASCII  /imp/
002033  154  151  145                .ASCII  /lie/
002036  163  040  163                .ASCII  /s s/
002041  145  161  165                .ASCII  /equ/
002044  145  156  164                .ASCII  /ent/
002047  151  141  154                .ASCII  /ial/
002052  051  000                    .ASCII  /)/<00>
002054  127  141  156        P.ABA:  .ASCII  /Wan/
002057  164  040  164                .ASCII  /t t/
002062  157  040  162                .ASCII  /o r/
002065  145  167  162                .ASCII  /ewr/
002070  151  164  145                .ASCII  /ite/
002073  040  142  154                .ASCII  / bl/
002076  157  143  153                .ASCII  /ock/
002101  163  040  167                .ASCII  /s w/
002104  150  145  156                .ASCII  /hen/
002107  040  042  106                .ASCII  / "F/
002112  157  162  143                .ASCII  /orc/
002115  145  144  040                .ASCII  /ed /
002120  105  162  162                .ASCII  /Err/
```

```
002123  157     162     042             .ASCII  /or"/
002126  040     144     145             .ASCII  / de/
002131  164     145     143             .ASCII  /tec/
002134  164     145     144             .ASCII  /ted/
002137  040     157     156             .ASCII  / on/
002142  040     162     145             .ASCII  / re/
002145  141     144     163             .ASCII  /ads/
002150  000     000                     .ASCII  <00><00>
002152  104     157     040     P.ABB:  .ASCII  /Do /
002155  171     157     165             .ASCII  /you/
002160  040     167     141             .ASCII  / wa/
002163  156     164     040             .ASCII  /nt /
002166  164     157     040             .ASCII  /to /
002171  150     141     154             .ASCII  /hal/
002174  164     040     157             .ASCII  /t o/
002177  156     040     157             .ASCII  /n o/
002202  164     150     145             .ASCII  /the/
002205  162     040     110             .ASCII  /r H/
002210  141     162     144             .ASCII  /ard/
002213  040     105     162             .ASCII  / Er/
002216  162     157     162             .ASCII  /ror/
002221  163     040     050             .ASCII  /s (/
002224  043     163     040             .ASCII  /#s /
002227  063     061     055             .ASCII  /31-/
002232  063     064     054             .ASCII  /34,/
002235  040     063     066             .ASCII  / 36/
002240  055     063     067             .ASCII  /-37/
002243  054     040     063             .ASCII  /, 3/
002246  071     055     064             .ASCII  /9-4/
002251  065     051     000             .ASCII  /5)/<00>
002254  104     157     040     P.ABC:  .ASCII  /Do /
002257  171     157     165             .ASCII  /you/
002262  040     167     141             .ASCII  / wa/
002265  156     164     040             .ASCII  /nt /
002270  164     157     040             .ASCII  /to /
002273  150     141     154             .ASCII  /hal/
002276  164     040     157             .ASCII  /t o/
002301  156     040     123             .ASCII  /n S/
002304  157     146     164             .ASCII  /oft/
002307  040     105     162             .ASCII  / Er/
002312  162     157     162             .ASCII  /ror/
002315  163     040     050             .ASCII  /s (/
002320  043     163     040             .ASCII  /#s /
002323  065     060     055             .ASCII  /50-/
002326  065     064     051             .ASCII  /54)/
002331  000                             .ASCII  <00>
002332  104     157     040     P.ABD:  .ASCII  /Do /
002335  171     157     165             .ASCII  /you/
002340  040     167     141             .ASCII  / wa/
002343  156     164     040             .ASCII  /nt /
002346  164     157     040             .ASCII  /to /
002351  150     141     154             .ASCII  /hal/
002354  164     040     157             .ASCII  /t o/
```

```
002357  156  040  102            .ASCII  /n B/
002362  141  144  055            .ASCII  /ad-/
002365  142  154  157            .ASCII  /blo/
002370  143  153  040            .ASCII  /ck /
002373  110  141  162            .ASCII  /Har/
002376  144  040  105            .ASCII  /d E/
002401  162  162  157            .ASCII  /rro/
002404  162  163  040            .ASCII  /rs /
002407  050  043  163            .ASCII  /(#s/
002412  040  063  065            .ASCII  / 35/
002415  054  040  063            .ASCII  /, 3/
002420  070  051  000            .ASCII  /8)/<00>
002423  000                      .ASCII  <00>
002424  105  156  164  P.ABE:    .ASCII  /Ent/
002427  145  162  040            .ASCII  /er /
002432  164  151  155            .ASCII  /tim/
002435  145  040  141            .ASCII  /e a/
002440  163  040  110            .ASCII  /s H/
002443  110  115  115            .ASCII  /HMM/
002446  040  050  105            .ASCII  / (E/
002451  170  141  155            .ASCII  /xam/
002454  160  154  145            .ASCII  /ple/
002457  072  040  061            .ASCII  /: 1/
002462  063  060  065            .ASCII  /305/
002465  040  055  040            .ASCII  / - /
002470  040  116  157            .ASCII  / No/
002473  040  154  145            .ASCII  / le/
002476  141  144  151            .ASCII  /adi/
002501  156  147  040            .ASCII  /ng /
002504  172  145  162            .ASCII  /zer/
002507  157  163  040            .ASCII  /os /
002512  141  154  154            .ASCII  /all/
002515  157  167  144            .ASCII  /owd/
002520  051  000                 .ASCII  /)/<00>
002522  103  157  165  P.ABF:    .ASCII  /Cou/
002525  156  164  040            .ASCII  /nt /
002530  145  141  143            .ASCII  /eac/
002533  150  040  162            .ASCII  /h r/
002536  145  164  162            .ASCII  /etr/
002541  171  040  157            .ASCII  /y o/
002544  156  040  141            .ASCII  /n a/
002547  040  122  145            .ASCII  / Re/
002552  141  144  057            .ASCII  /ad/<57>
002555  127  162  151            .ASCII  /Wri/
002560  164  145  040            .ASCII  /te /
002563  145  162  162            .ASCII  /err/
002566  157  162  040            .ASCII  /or /
002571  141  163  040            .ASCII  /as /
002574  141  040  163            .ASCII  /a s/
002577  145  160  145            .ASCII  /epe/
002602  162  141  164            .ASCII  /rat/
002605  145  040  123            .ASCII  /e S/
002610  157  146  164            .ASCII  /oft/
```

```
002613  040     105     162             .ASCII  / Er/
002616  162     157     162             .ASCII  /ror/
002621  000                             .ASCII  <00>
002622  124     150     145     P.ABG:  .ASCII  /The/
002625  040     162     145             .ASCII  / re/
002630  155     141     151             .ASCII  /mai/
002633  156     151     156             .ASCII  /nin/
002636  147     040     161             .ASCII  /g q/
002641  165     145     163             .ASCII  /ues/
002644  164     151     157             .ASCII  /tio/
002647  156     163     040             .ASCII  /ns /
002652  157     156     154             .ASCII  /onl/
002655  171     040     141             .ASCII  /y a/
002660  160     160     154             .ASCII  /ppl/
002663  171     040     164             .ASCII  /y t/
002666  157     040     165             .ASCII  /o u/
002671  156     160     162             .ASCII  /npr/
002674  157     164     145             .ASCII  /ote/
002677  143     164     145             .ASCII  /cte/
002702  144     040     144             .ASCII  /d d/
002705  151     163     153             .ASCII  /isk/
002710  163     000                     .ASCII  /s/<00>
002712  000     000             P.ABH:  .ASCII  <00><00>
002714  045     116     045     P.ABI:  .ASCII  /%N%/
002717  101     052     052             .ASCII  /A**/
002722  040     104     162             .ASCII  / Dr/
002725  157     160     040             .ASCII  /op /
002730  165     156     151             .ASCII  /uni/
002733  164     040     045             .ASCII  /t %/
002736  104     062     000             .ASCII  /D2/<00>
002741  000                             .ASCII  <00>
002742  045     116     045     P.ABJ:  .ASCII  /%N%/
002745  101     052     052             .ASCII  /A**/
002750  040     120     122             .ASCII  / PR/
002753  117     103     137             .ASCII  /OC_/
002756  122     105     124             .ASCII  /RET/
002761  120     113     124             .ASCII  /PKT/
002764  072     040     103             .ASCII  /: C/
002767  157     156     156             .ASCII  /onn/
002772  040     111     104             .ASCII  / ID/
002775  040     045     117             .ASCII  / %O/
003000  066     045     101             .ASCII  /6%A/
003003  040     162     145             .ASCII  / re/
003006  143     145     151             .ASCII  /cei/
003011  166     145     144             .ASCII  /ved/
003014  000     000                     .ASCII  <00><00>
003016  045     116     045     P.ABK:  .ASCII  /%N%/
003021  101     052     052             .ASCII  /A**/
003024  040     115     165             .ASCII  / Mu/
003027  154     164     151             .ASCII  /lti/
003032  055     144     162             .ASCII  /-dr/
003035  151     166     145             .ASCII  /ive/
003040  040     164     145             .ASCII  / te/
```

```
003043  163  164  000          .ASCII  /st/<00>
003046  045  116  045  P.ABL:   .ASCII  /%N%/
003051  101  052  052          .ASCII  /A**/
003054  040  106  101          .ASCII  / FA/
003057  124  101  114          .ASCII  /TAL/
003062  137  105  122          .ASCII  /_ER/
003065  122  117  122          .ASCII  /ROR/
003070  072  040  122          .ASCII  /: R/
003073  105  124  120          .ASCII  /ETP/
003076  113  124  040          .ASCII  /KT /
003101  156  157  164          .ASCII  /not/
003104  040  141  166          .ASCII  / av/
003107  141  151  154          .ASCII  /ail/
003112  141  142  154          .ASCII  /abl/
003115  145  000  000          .ASCII  /e/<00><00>
003120  045  116  045  P.ABM:   .ASCII  /%N%/
003123  101  052  052          .ASCII  /A**/
003126  040  106  123          .ASCII  / FS/
003131  105  124  137          .ASCII  /ET_/
003134  125  120  101          .ASCII  /UPA/
003137  122  072  040          .ASCII  /R: /
003142  103  141  156          .ASCII  /Can/
003145  047  164  040          .ASCII  /'t /
003150  146  151  156          .ASCII  /fin/
003153  144  040  144          .ASCII  /d d/
003156  151  163  153          .ASCII  /isk/
003161  040  045  104          .ASCII  / %D/
003164  063  045  101          .ASCII  /3%A/
003167  040  151  156          .ASCII  / in/
003172  040  103  123          .ASCII  / CS/
003175  124  040  045          .ASCII  /T %/
003200  104  061  000          .ASCII  /D1/<00>
003203  000                    .ASCII  <00>
003204  045  116  045  P.ABN:   .ASCII  /%N%/
003207  101  052  052          .ASCII  /A**/
003212  040  102  141          .ASCII  / Ba/
003215  144  040  143          .ASCII  /d c/
003220  157  156  156          .ASCII  /onn/
003223  040  111  104          .ASCII  / ID/
003226  040  045  117          .ASCII  / %O/
003231  066  045  101          .ASCII  /6%A/
003234  040  162  145          .ASCII  / re/
003237  143  145  151          .ASCII  /cei/
003242  166  145  144          .ASCII  /ved/
003245  040  146  162          .ASCII  / fr/
003250  157  155  040          .ASCII  /om /
003253  045  117  066          .ASCII  /%06/
003256  000  000               .ASCII  <00><00>
003260  045  116  045  P.ABO:   .ASCII  /%N%/
003263  101  052  052          .ASCII  /A**/
003266  040  115  145          .ASCII  / Me/
003271  163  163  141          .ASCII  /ssa/
003274  147  145  040          .ASCII  /ge /
```

```
003277  164  171  160            .ASCII  /typ/
003302  145  040  045            .ASCII  /e %/
003305  117  062  045            .ASCII  /02%/
003310  101  040  162            .ASCII  /A r/
003313  145  143  145            .ASCII  /ece/
003316  151  166  145            .ASCII  /ive/
003321  144  040  151            .ASCII  /d i/
003324  156  040  115            .ASCII  /n M/
003327  123  103  120            .ASCII  /SCP/
003332  040  160  141            .ASCII  / pa/
003335  143  153  145            .ASCII  /cke/
003340  164  000                 .ASCII  /t/<00>
003342  045  116  045  P.ABP:    .ASCII  /%N%/
003345  101  052  052            .ASCII  /A**/
003350  040  123  105            .ASCII  / SE/
003353  121  125  105            .ASCII  /QUE/
003356  116  072  040            .ASCII  /N: /
003361  122  105  124            .ASCII  /RET/
003364  120  113  124            .ASCII  /PKT/
003367  040  156  157            .ASCII  / no/
003372  164  040  141            .ASCII  /t a/
003375  166  141  151            .ASCII  /vai/
003400  154  141  142            .ASCII  /lab/
003403  154  145  000            .ASCII  /le/<00>
003406  045  116  045  P.ABQ:    .ASCII  /%N%/
003411  101  052  052            .ASCII  /A**/
003414  040  105  162            .ASCII  / Er/
003417  162  157  162            .ASCII  /ror/
003422  040  151  156            .ASCII  / in/
003425  040  123  105            .ASCII  / SE/
003430  124  137  103            .ASCII  /T_C/
003433  124  114  122            .ASCII  /TLR/
003436  137  103  110            .ASCII  /_CH/
003441  101  122  000            .ASCII  /AR/<00>
003444  045  116  045  P.ABR:    .ASCII  /%N%/
003447  101  052  052            .ASCII  /A**/
003452  040  103  164            .ASCII  / Ct/
003455  154  162  040            .ASCII  /lr /
003460  164  151  155            .ASCII  /tim/
003463  145  157  165            .ASCII  /eou/
003466  164  040  075            .ASCII  /t =/
003471  040  045  104            .ASCII  / %D/
003474  063  045  101            .ASCII  /3%A/
003477  056  040  163            .ASCII  /. s/
003502  145  143  157            .ASCII  /eco/
003505  156  144  163            .ASCII  /nds/
003510  000  000                 .ASCII  <00><00>
003512  045  116  045  P.ABS:    .ASCII  /%N%/
003515  101  052  052            .ASCII  /A**/
003520  040  105  162            .ASCII  / Er/
003523  162  157  162            .ASCII  /ror/
003526  040  151  156            .ASCII  / in/
003531  040  125  116            .ASCII  / UN/
```

```
003534  111  124  137            .ASCII  /IT_/
003537  111  116  111            .ASCII  /INI/
003542  124  000                 .ASCII  /T/<00>
003544  045  116  045  P.ABT:    .ASCII  /%N%/
003547  101  052  052            .ASCII  /A**/
003552  040  125  116            .ASCII  / UN/
003555  111  124  137            .ASCII  /IT_/
003560  111  116  111            .ASCII  /INI/
003563  124  072  040            .ASCII  /T: /
003566  122  105  124            .ASCII  /RET/
003571  120  113  124            .ASCII  /PKT/
003574  040  150  141            .ASCII  / ha/
003577  163  040  142            .ASCII  /s b/
003602  141  144  040            .ASCII  /ad /
003605  105  116  104            .ASCII  /END/
003610  103  117  104            .ASCII  /COD/
003613  105  000  000            .ASCII  /E/<00><00>
003616  045  116  045  P.ABU:    .ASCII  /%N%/
003621  101  052  052            .ASCII  /A**/
003624  040  125  156            .ASCII  / Un/
003627  151  164  040            .ASCII  /it /
003632  163  151  172            .ASCII  /siz/
003635  145  040  050            .ASCII  /e (/
003640  114  157  051            .ASCII  /Lo)/
003643  040  075  040            .ASCII  / = /
003646  045  104  065            .ASCII  /%D5/
003651  045  101  056            .ASCII  /%A./
003654  000  000                 .ASCII  <00><00>
003656  045  116  045  P.ABV:    .ASCII  /%N%/
003661  101  052  052            .ASCII  /A**/
003664  040  125  156            .ASCII  / Un/
003667  151  164  040            .ASCII  /it /
003672  163  151  172            .ASCII  /siz/
003675  145  040  050            .ASCII  /e (/
003700  110  151  051            .ASCII  /Hi)/
003703  040  075  040            .ASCII  / = /
003706  045  104  065            .ASCII  /%D5/
003711  045  101  056            .ASCII  /%A./
003714  000  000                 .ASCII  <00><00>
003716  045  116  045  P.ABW:    .ASCII  /%N%/
003721  101  052  052            .ASCII  /A**/
003724  040  101  103            .ASCII  / AC/
003727  103  105  123            .ASCII  /CES/
003732  123  072  040            .ASCII  /S: /
003735  122  105  124            .ASCII  /RET/
003740  120  113  124            .ASCII  /PKT/
003743  040  150  141            .ASCII  / ha/
003746  163  040  142            .ASCII  /s b/
003751  141  144  040            .ASCII  /ad /
003754  105  116  104            .ASCII  /END/
003757  103  117  104            .ASCII  /COD/
003762  105  000                 .ASCII  /E/<00>
003764  045  116  045  P.ABX:    .ASCII  /%N%/
```

```
003767  101  052  052            .ASCII  /A**/
003772  040  121  111            .ASCII  / QI/
003775  117  137  125            .ASCII  /O_U/
004000  116  111  124            .ASCII  /NIT/
004003  072  040  103            .ASCII  /: C/
004006  123  124  040            .ASCII  /ST /
004011  045  104  061            .ASCII  /%D1/
004014  045  101  040            .ASCII  /%A /
004017  156  157  040            .ASCII  /no /
004022  165  156  151            .ASCII  /uni/
004025  164  040  163            .ASCII  /t s/
004030  145  154  145            .ASCII  /ele/
004033  143  164  145            .ASCII  /cte/
004036  144  000                 .ASCII  /d/<00>
004040  045  116  045  P.ABY:    .ASCII  /%N%/
004043  101  052  052            .ASCII  /A**/
004046  040  125  156            .ASCII  / Un/
004051  151  164  040            .ASCII  /it /
004054  043  040  151            .ASCII  /# i/
004057  163  072  040            .ASCII  /s: /
004062  045  117  066            .ASCII  /%06/
004065  000                      .ASCII  <00>
004066  045  116  045  P.ABZ:    .ASCII  /%N%/
004071  101  052  052            .ASCII  /A**/
004074  040  122  145            .ASCII  / Re/
004077  155  157  166            .ASCII  /mov/
004102  141  142  154            .ASCII  /abl/
004105  145  040  144            .ASCII  /e d/
004110  151  163  153            .ASCII  /isk/
004113  040  151  163            .ASCII  / is/
004116  040  163  145            .ASCII  / se/
004121  154  145  143            .ASCII  /lec/
004124  164  145  144            .ASCII  /ted/
004127  000                      .ASCII  <00>
004130  045  116  045  P.ACA:    .ASCII  /%N%/
004133  101  052  052            .ASCII  /A**/
004136  040  106  151            .ASCII  / Fi/
004141  170  145  144            .ASCII  /xed/
004144  040  144  151            .ASCII  / di/
004147  163  153  040            .ASCII  /sk /
004152  151  163  040            .ASCII  /is /
004155  163  145  154            .ASCII  /sel/
004160  145  143  164            .ASCII  /ect/
004163  145  144  000            .ASCII  /ed/<00>
004166  045  116  045  P.ACB:    .ASCII  /%N%/
004171  101  052  052            .ASCII  /A**/
004174  040  111  154            .ASCII  / Il/
004177  154  145  147            .ASCII  /leg/
004202  141  154  040            .ASCII  /al /
004205  146  165  156            .ASCII  /fun/
004210  143  164  151            .ASCII  /cti/
004213  157  156  072            .ASCII  /on:/
004216  040  045  117            .ASCII  / %O/
```

```
004221  066  000  000          .ASCII  /6/<00><00>
004224  045  116  045  P.ACC:   .ASCII  /%N%/
004227  101  052  052          .ASCII  /A**/
004232  040  103  157          .ASCII  / Co/
004235  155  155  141          .ASCII  /mma/
004240  156  144  040          .ASCII  /nd /
004243  162  145  146          .ASCII  /ref/
004246  040  043  040          .ASCII  / # /
004251  045  117  066          .ASCII  /%06/
004254  045  101  057          .ASCII  /%A/<57>
004257  045  117  066          .ASCII  /%06/
004262  045  101  040          .ASCII  /%A /
004265  050  117  143          .ASCII  /(Oc/
004270  164  051  040          .ASCII  /t) /
004273  156  157  164          .ASCII  /not/
004276  040  163  145          .ASCII  / se/
004301  156  164  040          .ASCII  /nt /
004304  142  171  040          .ASCII  /by /
004307  110  157  163          .ASCII  /Hos/
004312  164  000               .ASCII  /t/<00>
004314  045  116  045  P.ACD:   .ASCII  /%N%/
004317  101  052  052          .ASCII  /A**/
004322  040  125  156          .ASCII  / Un/
004325  153  156  157          .ASCII  /kno/
004330  167  156  040          .ASCII  /wn /
004333  105  162  162          .ASCII  /Err/
004336  157  162  040          .ASCII  /or /
004341  114  157  147          .ASCII  /Log/
004344  040  146  157          .ASCII  / fo/
004347  162  155  141          .ASCII  /rma/
004352  164  040  045          .ASCII  /t %/
004355  117  063  045          .ASCII  /03%/
004360  101  040  162          .ASCII  /A r/
004363  145  143  145          .ASCII  /ece/
004366  151  166  145          .ASCII  /ive/
004371  144  000  000          .ASCII  /d/<00><00>
004374  045  116  045  P.ACE:   .ASCII  /%N%/
004377  101  052  052          .ASCII  /A**/
004402  040  117  160          .ASCII  / Op/
004405  055  143  157          .ASCII  /-co/
004410  144  145  040          .ASCII  /de /
004413  045  117  063          .ASCII  /%03/
004416  045  101  054          .ASCII  /%A,/
004421  040  105  156          .ASCII  / En/
004424  144  055  143          .ASCII  /d-c/
004427  157  144  145          .ASCII  /ode/
004432  040  045  117          .ASCII  / %O/
004435  063  045  101          .ASCII  /3%A/
004440  040  146  157          .ASCII  / fo/
004443  162  040  162          .ASCII  /r r/
004446  145  146  040          .ASCII  /ef /
004451  043  040  045          .ASCII  /# %/
004454  117  066  045          .ASCII  /O6%/
```

```
004457  101  057  045            .ASCII  /A/<57>/%/
004462  117  066  045            .ASCII  /06%/
004465  101  040  050            .ASCII  /A (/
004470  070  051  000            .ASCII  /8)/<00>
004473  000                      .ASCII  <00>
004474  045  116  045  P.ACF:    .ASCII  /%N%/
004477  101  052  052            .ASCII  /A**/
004502  040  103  155            .ASCII  / Cm/
004505  144  055  142            .ASCII  /d-b/
004510  143  040  045            .ASCII  /c %/
004513  117  066  045            .ASCII  /06%/
004516  101  057  045            .ASCII  /A/<57>/%/
004521  117  066  045            .ASCII  /06%/
004524  101  040  122            .ASCII  /A R/
004527  163  160  055            .ASCII  /sp-/
004532  142  143  040            .ASCII  /bc /
004535  045  117  066            .ASCII  /%06/
004540  045  101  057            .ASCII  /%A/<57>
004543  045  117  066            .ASCII  /%06/
004546  045  101  040            .ASCII  /%A /
004551  146  157  162            .ASCII  /for/
004554  040  045  117            .ASCII  / %0/
004557  066  045  101            .ASCII  /6%A/
004562  057  045  117            .ASCII  <57>/%0/
004565  066  045  101            .ASCII  /6%A/
004570  040  050  070            .ASCII  / (8/
004573  051  000  000            .ASCII  /)/<00><00>
004576  045  116  045  P.ACG:    .ASCII  /%N%/
004601  101  052  052            .ASCII  /A**/
004604  040  122  145            .ASCII  / Re/
004607  163  160  157            .ASCII  /spo/
004612  156  163  145            .ASCII  /nse/
004615  040  141  154            .ASCII  / al/
004620  162  145  141            .ASCII  /rea/
004623  144  171  040            .ASCII  /dy /
004626  162  145  143            .ASCII  /rec/
004631  145  151  166            .ASCII  /eiv/
004634  145  144  040            .ASCII  /ed /
004637  146  157  162            .ASCII  /for/
004642  040  143  155            .ASCII  / cm/
004645  144  040  045            .ASCII  /d %/
004650  117  066  045            .ASCII  /06%/
004653  101  057  045            .ASCII  /A/<57>/%/
004656  117  066  045            .ASCII  /06%/
004661  101  040  050            .ASCII  /A (/
004664  070  051  000            .ASCII  /8)/<00>
004667  000                      .ASCII  <00>
004670  045  116  045  P.ACH:    .ASCII  /%N%/
004673  101  052  052            .ASCII  /A**/
004676  040  106  141            .ASCII  / Fa/
004701  151  154  165            .ASCII  /ilu/
004704  162  145  040            .ASCII  /re /
004707  164  157  040            .ASCII  /to /
```

```
004712  163  145  156            .ASCII  /sen/
004715  144  040  143            .ASCII  /d c/
004720  157  155  155            .ASCII  /omm/
004723  141  156  144            .ASCII  /and/
004726  040  141  146            .ASCII  / af/
004731  164  145  162            .ASCII  /ter/
004734  040  043  040            .ASCII  / # /
004737  045  117  066            .ASCII  /%O6/
004742  045  101  057            .ASCII  /%A/<57>
004745  045  117  066            .ASCII  /%O6/
004750  045  101  040            .ASCII  /%A /
004753  050  070  051            .ASCII  /(8)/
004756  000  000                 .ASCII  <00><00>
004760  045  116  045  P.ACI:    .ASCII  /%N%/
004763  101  125  116            .ASCII  /AUN/
004766  111  124  045            .ASCII  /IT%/
004771  104  062  045            .ASCII  /D2%/
004774  101  040  104            .ASCII  /A D/
004777  122  117  120            .ASCII  /ROP/
005002  120  105  104            .ASCII  /PED/
005005  040  055  040            .ASCII  / - /
005010  000  000                 .ASCII  <00><00>
005012  045  101  125  P.ACK:    .ASCII  /%AU/
005015  123  105  122            .ASCII  /SER/
005020  040  103  117            .ASCII  / CO/
005023  115  115  101            .ASCII  /MMA/
005026  116  104  045            .ASCII  /ND%/
005031  116  000  000            .ASCII  /N/<00><00>
005034  045  101  103  P.ACL:    .ASCII  /%AC/
005037  117  116  106            .ASCII  /ONF/
005042  111  107  125            .ASCII  /IGU/
005045  122  101  124            .ASCII  /RAT/
005050  111  117  116            .ASCII  /ION/
005053  040  105  122            .ASCII  / ER/
005056  122  117  122            .ASCII  /ROR/
005061  045  116  000            .ASCII  /%N/<00>
005064  045  101  111  P.ACM:    .ASCII  /%AI/
005067  116  111  124            .ASCII  /NIT/
005072  040  105  122            .ASCII  / ER/
005075  122  117  122            .ASCII  /ROR/
005100  045  116  000            .ASCII  /%N/<00>
005103  000                      .ASCII  <00>
005104  045  101  124  P.ACN:    .ASCII  /%AT/
005107  122  101  116            .ASCII  /RAN/
005112  123  106  105            .ASCII  /SFE/
005115  122  040  114            .ASCII  /R L/
005120  111  115  111            .ASCII  /IMI/
005123  124  040  122            .ASCII  /T R/
005126  105  101  103            .ASCII  /EAC/
005131  110  105  104            .ASCII  /HED/
005134  045  116  000            .ASCII  /%N/<00>
005137  000                      .ASCII  <00>
005140  045  101  105  P.ACO:    .ASCII  /%AE/
```

```
005143  122  122  117          .ASCII  /RRO/
005146  122  040  114          .ASCII  /R L/
005151  111  115  111          .ASCII  /IMI/
005154  124  040  122          .ASCII  /T R/
005157  105  101  103          .ASCII  /EAC/
005162  110  105  104          .ASCII  /HED/
005165  045  116  000          .ASCII  /%N/<00>
005170  045  101  125  P.ACP:   .ASCII  /%AU/
005173  116  122  105          .ASCII  /NRE/
005176  103  117  126          .ASCII  /COV/
005201  105  122  101          .ASCII  /ERA/
005204  102  114  105          .ASCII  /BLE/
005207  040  104  122          .ASCII  / DR/
005212  111  126  105          .ASCII  /IVE/
005215  040  105  122          .ASCII  / ER/
005220  122  117  122          .ASCII  /ROR/
005223  045  116  000          .ASCII  /%N/<00>
005226  045  101  125  P.ACQ:   .ASCII  /%AU/
005231  116  122  105          .ASCII  /NRE/
005234  103  117  126          .ASCII  /COV/
005237  105  122  101          .ASCII  /ERA/
005242  102  114  105          .ASCII  /BLE/
005245  040  103  117          .ASCII  / CO/
005250  116  124  122          .ASCII  /NTR/
005253  117  114  114          .ASCII  /OLL/
005256  105  122  040          .ASCII  /ER /
005261  105  122  122          .ASCII  /ERR/
005264  117  122  045          .ASCII  /OR%/
005267  116  000  000          .ASCII  /N/<00><00>
005272  045  101  106  P.ACR:   .ASCII  /%AF/
005275  101  111  114          .ASCII  /AIL/
005300  105  104  040          .ASCII  /ED /
005303  124  117  040          .ASCII  /TO /
005306  103  117  115          .ASCII  /COM/
005311  105  040  117          .ASCII  /E O/
005314  116  114  111          .ASCII  /NLI/
005317  116  105  045          .ASCII  /NE%/
005322  116  000               .ASCII  /N/<00>
005324  045  101  106  P.ACS:   .ASCII  /%AF/
005327  101  111  114          .ASCII  /AIL/
005332  105  104  040          .ASCII  /ED /
005335  124  117  040          .ASCII  /TO /
005340  101  103  103          .ASCII  /ACC/
005343  105  123  123          .ASCII  /ESS/
005346  040  105  111          .ASCII  / EI/
005351  124  110  105          .ASCII  /THE/
005354  122  040  106          .ASCII  /R F/
005357  111  122  123          .ASCII  /IRS/
005362  124  040  117          .ASCII  /T O/
005365  122  040  114          .ASCII  /R L/
005370  101  123  124          .ASCII  /AST/
005373  040  124  122          .ASCII  / TR/
005376  101  103  113          .ASCII  /ACK/
```

```
005401      040     104     125             .ASCII  / DU/
005404      122     111     116             .ASCII  /RIN/
005407      107     040     111             .ASCII  /G I/
005412      116     111     124             .ASCII  /NIT/
005415      045     116     000             .ASCII  /%N/<00>
005420      045     101     104     P.ACT:  .ASCII  /%AD/
005423      111     123     113             .ASCII  /ISK/
005426      040     127     122             .ASCII  / WR/
005431      111     124     105             .ASCII  /ITE/
005434      040     120     122             .ASCII  / PR/
005437      117     124     105             .ASCII  /OTE/
005442      103     124     105             .ASCII  /CTE/
005445      104     045     116             .ASCII  /D%N/
005450      000     000                     .ASCII  <00><00>
005452      045     101     103     P.ACU:  .ASCII  /%AC/
005455      117     115     115             .ASCII  /OMM/
005460      101     116     104             .ASCII  /AND/
005463      040     124     111             .ASCII  / TI/
005466      115     105     040             .ASCII  /ME /
005471      117     125     124             .ASCII  /OUT/
005474      045     116     000             .ASCII  /%N/<00>
005477      000                             .ASCII  <00>
005500  005012'                     P.ACJ:  .WORD   P.ACK
005502  005034'                             .WORD   P.ACL
005504  005064'                             .WORD   P.ACM
005506  005104'                             .WORD   P.ACN
005510  005140'                             .WORD   P.ACO
005512  005170'                             .WORD   P.ACP
005514  005226'                             .WORD   P.ACQ
005516  005272'                             .WORD   P.ACR
005520  005324'                             .WORD   P.ACS
005522  005420'                             .WORD   P.ACT
005524  005452'                             .WORD   P.ACU
005526      045     116     045     P.ACV:  .ASCII  /%N%/
005531      101     120     117             .ASCII  /APO/
005534      127     105     122             .ASCII  /WER/
005537      040     104     105             .ASCII  / DE/
005542      114     101     131             .ASCII  /LAY/
005545      040     055     040             .ASCII  / - /
005550      127     101     111             .ASCII  /WAI/
005553      124     111     116             .ASCII  /TIN/
005556      107     000                     .ASCII  /G/<00>
005560      045     116     045     P.ACW:  .ASCII  /%N%/
005563      101     106     125             .ASCII  /AFU/
005566      116     103     124             .ASCII  /NCT/
005571      111     117     116             .ASCII  /ION/
005574      101     114     040             .ASCII  /AL /
005577      124     105     123             .ASCII  /TES/
005602      124     040     123             .ASCII  /T S/
005605      124     101     122             .ASCII  /TAR/
005610      124     105     104             .ASCII  /TED/
005613      000                             .ASCII  <00>
005614      045     116     045     P.ACX:  .ASCII  /%N%/
```

```
005617  116  045  101          .ASCII  /N%A/
005622  105  130  105          .ASCII  /EXE/
005625  122  103  111          .ASCII  /RCI/
005630  123  105  122          .ASCII  /SER/
005633  040  123  124          .ASCII  / ST/
005636  101  122  124          .ASCII  /ART/
005641  105  104  045          .ASCII  /ED%/
005644  116  000               .ASCII  /N/<00>
005646  045  116  045  P.ACY:   .ASCII  /%N%/
005651  116  045  101          .ASCII  /N%A/
005654  125  116  124          .ASCII  /UNT/
005657  040  104  123          .ASCII  / DS/
005662  113  045  123          .ASCII  /K%S/
005665  070  045  101          .ASCII  /8%A/
005670  043  040  117          .ASCII  /# O/
005673  106  040  040          .ASCII  /F  /
005676  040  043  040          .ASCII  / # /
005701  102  131  124          .ASCII  /BYT/
005704  105  123  040          .ASCII  /ES /
005707  040  040  043          .ASCII  /  #/
005712  040  117  106          .ASCII  / OF/
005715  040  040  040          .ASCII  /   /
005720  040  043  040          .ASCII  / # /
005723  102  131  124          .ASCII  /BYT/
005726  105  123  000          .ASCII  /ES/<00>
005731  000                    .ASCII  <00>
005732  045  101  040  P.ACZ:   .ASCII  /%A /
005735  040  055  055          .ASCII  / --/
005740  110  101  122          .ASCII  /HAR/
005743  104  040  105          .ASCII  /D E/
005746  122  122  117          .ASCII  /RRO/
005751  122  123  055          .ASCII  /RS-/
005754  055  040  055          .ASCII  /- -/
005757  055  123  117          .ASCII  /-SO/
005762  106  124  040          .ASCII  /FT /
005765  105  122  122          .ASCII  /ERR/
005770  117  122  123          .ASCII  /ORS/
005773  055  055  000          .ASCII  /--/<00>
005776  045  116  045  P.ADA:   .ASCII  /%N%/
006001  101  040  043          .ASCII  /A #/
006004  040  040  040          .ASCII  /   /
006007  043  040  040          .ASCII  /#  /
006012  124  131  120          .ASCII  /TYP/
006015  105  040  040          .ASCII  /E  /
006020  122  105  101          .ASCII  /REA/
006023  104  123  040          .ASCII  /DS /
006026  040  040  040          .ASCII  /   /
006031  040  122  105          .ASCII  / RE/
006034  101  104  040          .ASCII  /AD /
006037  040  040  127          .ASCII  /  W/
006042  122  111  124          .ASCII  /RIT/
006045  105  123  040          .ASCII  /ES /
006050  040  040  127          .ASCII  /  W/
```

```
006053  122  111  124          .ASCII  /RIT/
006056  124  105  116          .ASCII  /TEN/
006061  000                    .ASCII  <00>
006062  045  101  040  P.ADB:   .ASCII  /%A /
006065  040  123  105          .ASCII  / SE/
006070  113  040  104          .ASCII  /K D/
006073  101  124  040          .ASCII  /AT /
006076  104  122  126          .ASCII  /DRV/
006101  040  110  123          .ASCII  / HS/
006104  124  040  123          .ASCII  /T S/
006107  105  113  040          .ASCII  /EK /
006112  104  101  124          .ASCII  /DAT/
006115  040  104  122          .ASCII  / DR/
006120  126  040  110          .ASCII  /V H/
006123  123  124  000          .ASCII  /ST/<00>
006126  045  116  045  P.ADC:   .ASCII  /%N%/
006131  101  055  055          .ASCII  /A--/
006134  055  040  055          .ASCII  /- -/
006137  055  055  040          .ASCII  /-- /
006142  055  055  055          .ASCII  /---/
006145  055  040  040          .ASCII  /-  /
006150  055  055  055          .ASCII  /---/
006153  055  055  040          .ASCII  /-- /
006156  040  055  055          .ASCII  / --/
006161  055  055  055          .ASCII  /---/
006164  055  055  055          .ASCII  /---/
006167  055  040  055          .ASCII  /- -/
006172  055  055  055          .ASCII  /---/
006175  055  055  040          .ASCII  /-- /
006200  040  055  055          .ASCII  / --/
006203  055  055  055          .ASCII  /---/
006206  055  055  055          .ASCII  /---/
006211  055  000  000          .ASCII  /-/<00><00>
006214  045  101  040  P.ADD:   .ASCII  /%A /
006217  055  055  055          .ASCII  /---/
006222  040  055  055          .ASCII  / --/
006225  055  040  055          .ASCII  /- -/
006230  055  055  040          .ASCII  /-- /
006233  055  055  055          .ASCII  /---/
006236  040  055  055          .ASCII  / --/
006241  055  040  055          .ASCII  /- -/
006244  055  055  040          .ASCII  /-- /
006247  055  055  055          .ASCII  /---/
006252  040  055  055          .ASCII  / --/
006255  055  000  000          .ASCII  /-/<00><00>
006260  045  116  045  P.ADE:   .ASCII  /%N%/
006263  104  062  045          .ASCII  /D2%/
006266  104  064  045          .ASCII  /D4%/
006271  123  062  045          .ASCII  /S2%/
006274  124  000               .ASCII  /T/<00>
006276  045  104  064  P.ADF:   .ASCII  /%D4/
006301  045  132  063          .ASCII  /%Z3/
006304  045  104  063          .ASCII  /%D3/
```

```
006307  045  101  054                .ASCII  /%A,/
006312  045  132  063                .ASCII  /%Z3/
006315  045  101  054                .ASCII  /%A,/
006320  045  132  063                .ASCII  /%Z3/
006323  000                          .ASCII  <00>
006324  045  104  064        P.ADG:  .ASCII  /%D4/
006327  045  104  064                .ASCII  /%D4/
006332  045  104  064                .ASCII  /%D4/
006335  045  104  064                .ASCII  /%D4/
006340  045  104  064                .ASCII  /%D4/
006343  045  104  064                .ASCII  /%D4/
006346  045  104  064                .ASCII  /%D4/
006351  045  104  064                .ASCII  /%D4/
006354  000  000                     .ASCII  <00><00>
006356  045  116  045        P.ADH:  .ASCII  /%N%/
006361  101  040  056                .ASCII  /A ./
006364  040  040  040                .ASCII  /   /
006367  056  040  040                .ASCII  /.  /
006372  103  116  124                .ASCII  /CNT/
006375  122  040  040                .ASCII  /R  /
006400  040  040  040                .ASCII  /   /
006403  040  056  040                .ASCII  / . /
006406  040  056  056                .ASCII  / ../
006411  056  056  056                .ASCII  /.../
006414  056  056  056                .ASCII  /.../
006417  056  040  040                .ASCII  /.  /
006422  040  040  040                .ASCII  /   /
006425  040  056  040                .ASCII  / . /
006430  040  056  056                .ASCII  / ../
006433  056  056  056                .ASCII  /.../
006436  056  056  056                .ASCII  /.../
006441  056  000  000                .ASCII  /./<00><00>
006444  045  101  040        P.ADI:  .ASCII  /%A /
006447  040  040  056                .ASCII  /  ./
006452  040  040  040                .ASCII  /   /
006455  056  045  104                .ASCII  /.%D/
006460  064  045  101                .ASCII  /4%A/
006463  040  040  040                .ASCII  /   /
006466  056  040  040                .ASCII  /.  /
006471  040  056  040                .ASCII  / . /
006474  040  040  056                .ASCII  /  ./
006477  045  104  064                .ASCII  /%D4/
006502  045  101  040                .ASCII  /%A /
006505  040  040  056                .ASCII  /  ./
006510  000  000                     .ASCII  <00><00>
006512  045  101  040        P.ADJ:  .ASCII  /%A /
006515  040  040  056                .ASCII  /  ./
006520  040  040  040                .ASCII  /   /
006523  056  045  104                .ASCII  /.%D/
006526  064  045  101                .ASCII  /4%A/
006531  040  040  040                .ASCII  /   /
006534  056  040  040                .ASCII  /.  /
006537  040  056  040                .ASCII  / . /
```

```
006542  040  040  056            .ASCII  /  ./
006545  045  104  064            .ASCII  /%D4/
006550  045  101  040            .ASCII  /%A /
006553  040  040  056            .ASCII  /  ./
006556  000  000                 .ASCII  <00><00>
006560  045  116  045  P.ADK:    .ASCII  /%N%/
006563  116  045  101            .ASCII  /N%A/
006566  125  116  111            .ASCII  /UNI/
006571  124  040  040            .ASCII  /T  /
006574  104  111  123            .ASCII  /DIS/
006577  113  040  040            .ASCII  /K  /
006602  040  040  040            .ASCII  /   /
006605  040  040  040            .ASCII  /   /
006610  040  040  040            .ASCII  /   /
006613  040  043  040            .ASCII  / # /
006616  117  106  040            .ASCII  /OF /
006621  040  040  043            .ASCII  /  #/
006624  040  102  114            .ASCII  / BL/
006627  113  123  040            .ASCII  /KS /
006632  040  040  040            .ASCII  /   /
006635  040  040  040            .ASCII  /   /
006640  043  040  117            .ASCII  /# O/
006643  106  040  040            .ASCII  /F  /
006646  040  040  043            .ASCII  /  #/
006651  040  102  114            .ASCII  / BL/
006654  113  123  040            .ASCII  /KS /
006657  000                      .ASCII  <00>
006660  045  116  045  P.ADL:    .ASCII  /%N%/
006663  101  040  040            .ASCII  /A  /
006666  043  040  040            .ASCII  /#  /
006671  040  040  040            .ASCII  /   /
006674  043  040  040            .ASCII  /#  /
006677  040  040  040            .ASCII  /   /
006702  124  131  120            .ASCII  /TYP/
006705  105  040  040            .ASCII  /E  /
006710  040  122  105            .ASCII  / RE/
006713  101  104  123            .ASCII  /ADS/
006716  040  040  040            .ASCII  /   /
006721  040  040  122            .ASCII  /  R/
006724  105  101  104            .ASCII  /EAD/
006727  040  040  040            .ASCII  /   /
006732  040  040  040            .ASCII  /   /
006735  127  122  111            .ASCII  /WRI/
006740  124  105  123            .ASCII  /TES/
006743  040  040  127            .ASCII  /  W/
006746  122  111  124            .ASCII  /RIT/
006751  124  105  116            .ASCII  /TEN/
006754  040  000                 .ASCII  / /<00>
006756  045  116  045  P.ADM:    .ASCII  /%N%/
006761  101  055  055            .ASCII  /A--/
006764  055  055  040            .ASCII  /-- /
006767  040  055  055            .ASCII  / --/
006772  055  055  040            .ASCII  /-- /
```

```
006775   040   055   055                .ASCII  / --/
007000   055   055   055                .ASCII  /---/
007003   055   055   040                .ASCII  /-- /
007006   040   055   055                .ASCII  / --/
007011   055   055   055                .ASCII  /---/
007014   055   040   040                .ASCII  /-  /
007017   040   055   055                .ASCII  / --/
007022   055   055   055                .ASCII  /---/
007025   055   040   040                .ASCII  /-  /
007030   040   040   040                .ASCII  /   /
007033   055   055   055                .ASCII  /---/
007036   055   055   055                .ASCII  /---/
007041   040   040   040                .ASCII  /   /
007044   055   055   055                .ASCII  /---/
007047   055   055   055                .ASCII  /---/
007052   040   040   000                .ASCII  /  /<00>
007055   000                            .ASCII  <00>
007056   045   116   045        P.ADN:  .ASCII  /%N%/
007061   123   061   045                .ASCII  /S1%/
007064   104   062   045                .ASCII  /D2%/
007067   123   064   045                .ASCII  /S4%/
007072   104   062   045                .ASCII  /D2%/
007075   101   040   040                .ASCII  /A  /
007100   040   104   102                .ASCII  / DB/
007103   116   040   111                .ASCII  /N I/
007106   057   117   040                .ASCII  <57>/O /
007111   040   045   104                .ASCII  / %D/
007114   066   045   123                .ASCII  /6%S/
007117   063   045   104                .ASCII  /3%D/
007122   066   045   123                .ASCII  /6%S/
007125   065   045   104                .ASCII  /5%D/
007130   066   045   123                .ASCII  /6%S/
007133   063   045   104                .ASCII  /3%D/
007136   066   000                      .ASCII  /6/<00>
007140   124   117   117        P.ADO:  .ASCII  /TOO/
007143   040   115   101                .ASCII  / MA/
007146   116   131   040                .ASCII  /NY /
007151   125   116   111                .ASCII  /UNI/
007154   124   123   000                .ASCII  /TS/<00>
007157   000                            .ASCII  <00>
007160   116   117   124        P.ADP:  .ASCII  /NOT/
007163   040   105   116                .ASCII  / EN/
007166   117   125   107                .ASCII  /OUG/
007171   110   040   106                .ASCII  /H F/
007174   122   105   105                .ASCII  /REE/
007177   040   115   105                .ASCII  / ME/
007202   115   117   122                .ASCII  /MOR/
007205   131   040   106                .ASCII  /Y F/
007210   117   122   040                .ASCII  /OR /
007213   101   114   114                .ASCII  /ALL/
007216   117   103   101                .ASCII  /OCA/
007221   124   111   116                .ASCII  /TIN/
007224   107   040   122                .ASCII  /G R/
```

```
007227  105     101     104             .ASCII  /EAD/
007232  057     127     122             .ASCII  <57>/WR/
007235  111     124     105             .ASCII  /ITE/
007240  040     102     125             .ASCII  / BU/
007243  106     106     105             .ASCII  /FFE/
007246  122     123     000             .ASCII  /RS/<00>
007251  000                             .ASCII  <00>
007252  122     105     107     P.ADQ:  .ASCII  /REG/
007255  111     123     124             .ASCII  /IST/
007260  105     122     040             .ASCII  /ER /
007263  105     130     111             .ASCII  /EXI/
007266  123     124     105             .ASCII  /STE/
007271  116     103     105             .ASCII  /NCE/
007274  040     124     105             .ASCII  / TE/
007277  123     124     040             .ASCII  /ST /
007302  106     101     111             .ASCII  /FAI/
007305  114     105     104             .ASCII  /LED/
007310  000     000                     .ASCII  <00><00>
007312  126     105     103     P.ADR:  .ASCII  /VEC/
007315  124     117     122             .ASCII  /TOR/
007320  040     124     105             .ASCII  / TE/
007323  123     124     040             .ASCII  /ST /
007326  106     101     111             .ASCII  /FAI/
007331  114     105     104             .ASCII  /LED/
007334  000     000                     .ASCII  <00><00>
007336  102     122     040     P.ADS:  .ASCII  /BR /
007341  114     105     126             .ASCII  /LEV/
007344  105     114     040             .ASCII  /EL /
007347  124     105     123             .ASCII  /TES/
007352  124     040     106             .ASCII  /T F/
007355  101     111     114             .ASCII  /AIL/
007360  105     104     000             .ASCII  /ED/<00>
007363  000                             .ASCII  <00>
007364  111     116     111     P.ADT:  .ASCII  /INI/
007367  124     040     123             .ASCII  /T S/
007372  105     121     125             .ASCII  /EQU/
007375  105     116     103             .ASCII  /ENC/
007400  105     040     106             .ASCII  /E F/
007403  101     111     114             .ASCII  /AIL/
007406  105     104     000             .ASCII  /ED/<00>
007411  000                             .ASCII  <00>
007412  106     101     124     P.ADU:  .ASCII  /FAT/
007415  101     114     040             .ASCII  /AL /
007420  103     117     116             .ASCII  /CON/
007423  124     122     117             .ASCII  /TRO/
007426  114     114     105             .ASCII  /LLE/
007431  122     040     105             .ASCII  /R E/
007434  122     122     117             .ASCII  /RRO/
007437  122     000     000             .ASCII  /R/<00><00>
007442  117     116     114     P.ADV:  .ASCII  /ONL/
007445  111     116     105             .ASCII  /INE/
007450  040     106     101             .ASCII  / FA/
007453  111     114     105             .ASCII  /ILE/
```

```
007456   104   000                        .ASCII  /D/<00>
007460   127   122   111        P.ADW:    .ASCII  /WRI/
007463   124   105   055                  .ASCII  /TE-/
007466   120   122   117                  .ASCII  /PRO/
007471   124   105   103                  .ASCII  /TEC/
007474   124   040   103                  .ASCII  /T C/
007477   117   116   106                  .ASCII  /ONF/
007502   114   111   103                  .ASCII  /LIC/
007505   124   000   000                  .ASCII  /T/<00><00>
007510   101   103   103        P.ADX:    .ASCII  /ACC/
007513   105   123   123                  .ASCII  /ESS/
007516   040   106   101                  .ASCII  / FA/
007521   111   114   105                  .ASCII  /ILE/
007524   104   000                        .ASCII  /D/<00>
007526   106   101   124        P.ADY:    .ASCII  /FAT/
007531   101   114   040                  .ASCII  /AL /
007534   111   057   117                  .ASCII  /I/<57>/O/
007537   040   105   122                  .ASCII  / ER/
007542   122   117   122                  .ASCII  /ROR/
007545   000                              .ASCII  <00>
007546   104   111   123        P.ADZ:    .ASCII  /DIS/
007551   113   040   124                  .ASCII  /K T/
007554   131   120   105                  .ASCII  /YPE/
007557   040   125   116                  .ASCII  / UN/
007562   113   116   117                  .ASCII  /KNO/
007565   127   116   040                  .ASCII  /WN /
007570   124   117   040                  .ASCII  /TO /
007573   105   130   105                  .ASCII  /EXE/
007576   122   103   111                  .ASCII  /RCI/
007601   123   105   122                  .ASCII  /SER/
007604   000   000                        .ASCII  <00><00>
007606   106   101   111        P.AEA:    .ASCII  /FAI/
007611   114   105   104                  .ASCII  /LED/
007614   040   124   117                  .ASCII  / TO/
007617   040   123   105                  .ASCII  / SE/
007622   116   104   040                  .ASCII  /ND /
007625   123   105   124                  .ASCII  /SET/
007630   055   103   117                  .ASCII  /-CO/
007633   116   124   122                  .ASCII  /NTR/
007636   117   114   114                  .ASCII  /OLL/
007641   105   122   055                  .ASCII  /ER-/
007644   103   110   101                  .ASCII  /CHA/
007647   122   101   103                  .ASCII  /RAC/
007652   124   105   122                  .ASCII  /TER/
007655   111   123   124                  .ASCII  /IST/
007660   111   103   123                  .ASCII  /ICS/
007663   040   103   117                  .ASCII  / CO/
007666   115   115   101                  .ASCII  /MMA/
007671   116   104   000                  .ASCII  /ND/<00>
007674   123   105   124        P.AEB:    .ASCII  /SET/
007677   055   103   117                  .ASCII  /-CO/
007702   116   124   122                  .ASCII  /NTR/
007705   117   114   114                  .ASCII  /OLL/
```

```
007710      105     122     055             .ASCII  /ER-/
007713      103     110     101             .ASCII  /CHA/
007716      122     101     103             .ASCII  /RAC/
007721      124     105     122             .ASCII  /TER/
007724      111     123     124             .ASCII  /IST/
007727      111     103     123             .ASCII  /ICS/
007732      040     122     105             .ASCII  / RE/
007735      123     120     117             .ASCII  /SPO/
007740      116     123     105             .ASCII  /NSE/
007743      040     110     101             .ASCII  / HA/
007746      123     040     102             .ASCII  /S B/
007751      101     104     040             .ASCII  /AD /
007754      105     116     104             .ASCII  /END/
007757      103     117     104             .ASCII  /COD/
007762      105     040     117             .ASCII  /E O/
007765      122     040     106             .ASCII  /R F/
007770      114     101     107             .ASCII  /LAG/
007773      123     040     111             .ASCII  /S I/
007776      116     040     105             .ASCII  /N E/
010001      122     122     117             .ASCII  /RRO/
010004      122     000                     .ASCII  /R/<00>
010006      106     101     111     P.AEC:  .ASCII  /FAI/
010011      114     105     104             .ASCII  /LED/
010014      040     124     117             .ASCII  / TO/
010017      040     123     105             .ASCII  / SE/
010022      116     104     040             .ASCII  /ND /
010025      117     116     055             .ASCII  /ON-/
010030      114     111     116             .ASCII  /LIN/
010033      105     040     103             .ASCII  /E C/
010036      117     115     115             .ASCII  /OMM/
010041      101     116     104             .ASCII  /AND/
010044      000     000                     .ASCII  <00><00>
010046      117     116     055     P.AED:  .ASCII  /ON-/
010051      114     111     116             .ASCII  /LIN/
010054      105     040     122             .ASCII  /E R/
010057      105     123     120             .ASCII  /ESP/
010062      117     116     123             .ASCII  /ONS/
010065      105     040     110             .ASCII  /E H/
010070      101     123     040             .ASCII  /AS /
010073      102     101     104             .ASCII  /BAD/
010076      040     105     116             .ASCII  / EN/
010101      104     103     117             .ASCII  /DCO/
010104      104     105     000             .ASCII  /DE/<00>
010107      000                             .ASCII  <00>
010110      117     116     055     P.AEE:  .ASCII  /ON-/
010113      114     111     116             .ASCII  /LIN/
010116      105     040     122             .ASCII  /E R/
010121      105     123     120             .ASCII  /ESP/
010124      117     116     123             .ASCII  /ONS/
010127      105     040     110             .ASCII  /E H/
010132      101     123     040             .ASCII  /AS /
010135      125     116     113             .ASCII  /UNK/
010140      116     117     127             .ASCII  /NOW/
```

```
010143  116  040  104            .ASCII  /N D/
010146  105  126  111            .ASCII  /EVI/
010151  103  105  000            .ASCII  /CE/<00>
010154  111  057  117  P.AEF:    .ASCII  /I/<57>/O/
010157  040  122  105            .ASCII  / RE/
010162  121  125  105            .ASCII  /QUE/
010165  123  124  040            .ASCII  /ST /
010170  106  101  111            .ASCII  /FAI/
010173  114  105  104            .ASCII  /LED/
010176  000  000                 .ASCII  <00><00>
010200  045  101  115  P.AEG:    .ASCII  /%AM/
010203  117  122  105            .ASCII  /ORE/
010206  040  124  110            .ASCII  / TH/
010211  101  116  040            .ASCII  /AN /
010214  045  104  062            .ASCII  /%D2/
010217  045  101  040            .ASCII  /%A /
010222  125  116  111            .ASCII  /UNI/
010225  124  123  040            .ASCII  /TS /
010230  123  120  105            .ASCII  /SPE/
010233  103  111  106            .ASCII  /CIF/
010236  111  105  104            .ASCII  /IED/
010241  000                      .ASCII  <00>
010242  045  101  052  P.AEH:    .ASCII  /%A*/
010245  040  116  117            .ASCII  / NO/
010250  040  122  105            .ASCII  / RE/
010253  123  120  117            .ASCII  /SPO/
010256  116  123  105            .ASCII  /NSE/
010261  040  101  124            .ASCII  / AT/
010264  040  101  104            .ASCII  / AD/
010267  104  122  105            .ASCII  /DRE/
010272  123  123  040            .ASCII  /SS /
010275  045  117  066            .ASCII  /%O6/
010300  000  000                 .ASCII  <00><00>
010302  045  101  052  P.AEI:    .ASCII  /%A*/
010305  040  111  116            .ASCII  / IN/
010310  103  117  122            .ASCII  /COR/
010313  122  105  103            .ASCII  /REC/
010316  124  040  102            .ASCII  /T B/
010321  122  040  114            .ASCII  /R L/
010324  105  126  105            .ASCII  /EVE/
010327  114  040  106            .ASCII  /L F/
010332  117  122  040            .ASCII  /OR /
010335  104  122  111            .ASCII  /DRI/
010340  126  105  040            .ASCII  /VE /
010343  045  117  066            .ASCII  /%O6/
010346  000  000                 .ASCII  <00><00>
010350  045  101  052  P.AEJ:    .ASCII  /%A*/
010353  040  123  124            .ASCII  / ST/
010356  105  120  040            .ASCII  /EP /
010361  045  104  061            .ASCII  /%D1/
010364  045  101  040            .ASCII  /%A /
010367  122  105  101            .ASCII  /REA/
010372  104  040  105            .ASCII  /D E/
```

```
010375  122  122  117            .ASCII  /RRO/
010400  122  000                 .ASCII  /R/<00>
010402  045  101  052    P.AEK:  .ASCII  /%A*/
010405  040  102  101            .ASCII  / BA/
010410  104  040  123            .ASCII  /D S/
010413  101  040  103            .ASCII  /A C/
010416  117  104  105            .ASCII  /ODE/
010421  040  106  122            .ASCII  / FR/
010424  117  115  040            .ASCII  /OM /
010427  104  122  111            .ASCII  /DRI/
010432  126  105  040            .ASCII  /VE /
010435  045  117  066            .ASCII  /%06/
010440  000  000                 .ASCII  <00><00>
010442  045  101  052    P.AEL:  .ASCII  /%A*/
010445  040  104  111            .ASCII  / DI/
010450  123  113  045            .ASCII  /SK%/
010453  104  062  045            .ASCII  /D2%/
010456  101  040  127            .ASCII  /A W/
010461  105  116  124            .ASCII  /ENT/
010464  040  117  106            .ASCII  / OF/
010467  106  114  111            .ASCII  /FLI/
010472  116  105  000            .ASCII  /NE/<00>
010475  000                      .ASCII  <00>
010476  045  101  052    P.AEM:  .ASCII  /%A*/
010501  040  104  122            .ASCII  / DR/
010504  111  126  105            .ASCII  /IVE/
010507  040  045  117            .ASCII  / %O/
010512  066  045  101            .ASCII  /6%A/
010515  040  116  117            .ASCII  / NO/
010520  124  040  120            .ASCII  /T P/
010523  122  117  103            .ASCII  /ROC/
010526  105  123  123            .ASCII  /ESS/
010531  111  116  107            .ASCII  /ING/
010534  040  103  117            .ASCII  / CO/
010537  115  115  101            .ASCII  /MMA/
010542  116  104  040            .ASCII  /ND /
010545  120  101  103            .ASCII  /PAC/
010550  113  105  124            .ASCII  /KET/
010553  123  000  000            .ASCII  /S/<00><00>
010556  045  101  052    P.AEN:  .ASCII  /%A*/
010561  040  104  111            .ASCII  / DI/
010564  123  113  045            .ASCII  /SK%/
010567  104  062  045            .ASCII  /D2%/
010572  101  040  127            .ASCII  /A W/
010575  105  116  124            .ASCII  /ENT/
010600  040  124  117            .ASCII  / TO/
010603  040  124  110            .ASCII  / TH/
010606  105  040  042            .ASCII  /E "/
010611  101  126  101            .ASCII  /AVA/
010614  111  114  101            .ASCII  /ILA/
010617  102  114  105            .ASCII  /BLE/
010622  042  040  123            .ASCII  /" S/
010625  124  101  124            .ASCII  /TAT/
```

```
010630   105   000                        .ASCII  /E/<00>
010632   040   055   040        P.AEO:    .ASCII  / - /
010635   125   116   122                  .ASCII  /UNR/
010640   105   103   117                  .ASCII  /ECO/
010643   107   116   111                  .ASCII  /GNI/
010646   132   105   104                  .ASCII  /ZED/
010651   040   115   105                  .ASCII  / ME/
010654   123   123   101                  .ASCII  /SSA/
010657   107   105   040                  .ASCII  /GE /
010662   124   131   120                  .ASCII  /TYP/
010665   105   000   000                  .ASCII  /E/<00><00>
010670   040   055   040        P.AEP:    .ASCII  / - /
010673   125   116   122                  .ASCII  /UNR/
010676   105   103   117                  .ASCII  /ECO/
010701   107   116   111                  .ASCII  /GNI/
010704   132   105   104                  .ASCII  /ZED/
010707   040   103   117                  .ASCII  / CO/
010712   116   116   105                  .ASCII  /NNE/
010715   103   124   111                  .ASCII  /CTI/
010720   117   116   040                  .ASCII  /ON /
010723   111   104   000                  .ASCII  /ID/<00>
010726   040   055   040        P.AEQ:    .ASCII  / - /
010731   125   116   122                  .ASCII  /UNR/
010734   105   103   117                  .ASCII  /ECO/
010737   107   116   111                  .ASCII  /GNI/
010742   132   105   104                  .ASCII  /ZED/
010745   040   122   105                  .ASCII  / RE/
010750   124   125   122                  .ASCII  /TUR/
010753   116   040   115                  .ASCII  /N M/
010756   105   123   123                  .ASCII  /ESS/
010761   101   107   105                  .ASCII  /AGE/
010764   000   000                        .ASCII  <00><00>
010766   040   055   040        P.AER:    .ASCII  / - /
010771   125   116   122                  .ASCII  /UNR/
010774   105   103   117                  .ASCII  /ECO/
010777   107   116   111                  .ASCII  /GNI/
011002   132   105   104                  .ASCII  /ZED/
011005   040   122   105                  .ASCII  / RE/
011010   124   125   122                  .ASCII  /TUR/
011013   116   040   120                  .ASCII  /N P/
011016   101   103   113                  .ASCII  /ACK/
011021   105   124   000                  .ASCII  /ET/<00>
011024   040   055   040        P.AES:    .ASCII  / - /
011027   125   116   122                  .ASCII  /UNR/
011032   105   103   117                  .ASCII  /ECO/
011035   107   116   111                  .ASCII  /GNI/
011040   132   105   104                  .ASCII  /ZED/
011043   040   103   122                  .ASCII  / CR/
011046   116   000                        .ASCII  /N/<00>
011050   040   055   040        P.AET:    .ASCII  / - /
011053   125   116   122                  .ASCII  /UNR/
011056   105   103   117                  .ASCII  /ECO/
011061   107   116   111                  .ASCII  /GNI/
```

```
011064  132  105  104            .ASCII  /ZED/
011067  040  117  120            .ASCII  / OP/
011072  103  117  104            .ASCII  /COD/
011075  105  000  000            .ASCII  /E/<00><00>
011100  040  055  040  P.AEU:    .ASCII  / - /
011103  115  123  103            .ASCII  /MSC/
011106  120  040  123            .ASCII  /P S/
011111  124  101  124            .ASCII  /TAT/
011114  125  123  040            .ASCII  /US /
011117  103  117  104            .ASCII  /COD/
011122  105  040  105            .ASCII  /E E/
011125  122  122  000            .ASCII  /RR/<00>
011130  040  055  040  P.AEV:    .ASCII  / - /
011133  104  125  120            .ASCII  /DUP/
011136  040  123  124            .ASCII  / ST/
011141  101  124  125            .ASCII  /ATU/
011144  123  040  103            .ASCII  /S C/
011147  117  104  105            .ASCII  /ODE/
011152  040  105  122            .ASCII  / ER/
011155  122  000  000            .ASCII  /R/<00><00>
011160  040  055  040  P.AEW:    .ASCII  / - /
011163  125  116  122            .ASCII  /UNR/
011166  105  103  117            .ASCII  /ECO/
011171  107  116  111            .ASCII  /GNI/
011174  132  105  104            .ASCII  /ZED/
011177  040  123  124            .ASCII  / ST/
011202  101  124  125            .ASCII  /ATU/
011205  123  040  103            .ASCII  /S C/
011210  117  104  105            .ASCII  /ODE/
011213  000                      .ASCII  <00>
011214  040  055  040  P.AEX:    .ASCII  / - /
011217  114  102  116            .ASCII  /LBN/
011222  040  110  117            .ASCII  / HO/
011225  123  124  040            .ASCII  /ST /
011230  103  117  115            .ASCII  /COM/
011233  120  101  122            .ASCII  /PAR/
011236  105  040  105            .ASCII  /E E/
011241  122  122  000            .ASCII  /RR/<00>
011244  040  055  040  P.AEY:    .ASCII  / - /
011247  104  102  116            .ASCII  /DBN/
011252  040  110  117            .ASCII  / HO/
011255  123  124  040            .ASCII  /ST /
011260  103  117  115            .ASCII  /COM/
011263  120  101  122            .ASCII  /PAR/
011266  105  040  105            .ASCII  /E E/
011271  122  122  000            .ASCII  /RR/<00>
011274  040  055  040  P.AEZ:    .ASCII  / - /
011277  125  116  101            .ASCII  /UNA/
011302  102  114  105            .ASCII  /BLE/
011305  040  124  117            .ASCII  / TO/
011310  040  114  117            .ASCII  / LO/
011313  101  104  040            .ASCII  /AD /
011316  104  125  120            .ASCII  /DUP/
```

```
011321  040  115  105                .ASCII  / ME/
011324  104  111  101                .ASCII  /DIA/
011327  000                         .ASCII  <00>
011330  040  055  040       P.AFA:   .ASCII  / - /
011333  105  122  122                .ASCII  /ERR/
011336  040  111  116                .ASCII  / IN/
011341  040  104  125                .ASCII  / DU/
011344  120  040  120                .ASCII  /P P/
011347  113  124  040                .ASCII  /KT /
011352  127  110  105                .ASCII  /WHE/
011355  116  040  125                .ASCII  /N U/
011360  123  111  116                .ASCII  /SIN/
011363  107  040  103                .ASCII  /G C/
011366  124  114  122                .ASCII  /TLR/
011371  040  114  103                .ASCII  / LC/
011374  040  120  122                .ASCII  / PR/
011377  117  107  000                .ASCII  /OG/<00>
011402  045  101  052       P.AFB:   .ASCII  /%A*/
011405  040  104  111                .ASCII  / DI/
011410  123  113  045                .ASCII  /SK%/
011413  104  062  000                .ASCII  /D2/<00>
011416  045  101  111       P.AFD:   .ASCII  /%AI/
011421  116  126  101                .ASCII  /NVA/
011424  114  111  104                .ASCII  /LID/
011427  040  103  117                .ASCII  / CO/
011432  115  115  101                .ASCII  /MMA/
011435  116  104  000                .ASCII  /ND/<00>
011440  045  101  103       P.AFE:   .ASCII  /%AC/
011443  117  115  115                .ASCII  /OMM/
011446  101  116  104                .ASCII  /AND/
011451  040  101  102                .ASCII  / AB/
011454  117  122  124                .ASCII  /ORT/
011457  105  104  000                .ASCII  /ED/<00>
011462  045  101  125       P.AFF:   .ASCII  /%AU/
011465  116  111  124                .ASCII  /NIT/
011470  040  117  106                .ASCII  / OF/
011473  106  114  111                .ASCII  /FLI/
011476  116  105  000                .ASCII  /NE/<00>
011501  000                         .ASCII  <00>
011502  045  101  124       P.AFG:   .ASCII  /%AT/
011505  122  101  116                .ASCII  /RAN/
011510  123  111  124                .ASCII  /SIT/
011513  111  117  116                .ASCII  /ION/
011516  040  124  117                .ASCII  / TO/
011521  040  101  126                .ASCII  / AV/
011524  101  111  114                .ASCII  /AIL/
011527  101  102  114                .ASCII  /ABL/
011532  105  040  123                .ASCII  /E S/
011535  124  101  124                .ASCII  /TAT/
011540  105  000                     .ASCII  /E/<00>
011542  045  101  115       P.AFH:   .ASCII  /%AM/
011545  105  104  111                .ASCII  /EDI/
011550  101  040  106                .ASCII  /A F/
```

```
011553  117  122  115                .ASCII  /ORM/
011556  101  124  040                .ASCII  /AT /
011561  105  122  122                .ASCII  /ERR/
011564  117  122  000                .ASCII  /OR/<00>
011567  000                         .ASCII  <00>
011570  045  101  127     P.AFI:     .ASCII  /%AW/
011573  122  111  124                .ASCII  /RIT/
011576  105  055  120                .ASCII  /E-P/
011601  122  117  124                .ASCII  /ROT/
011604  105  103  124                .ASCII  /ECT/
011607  105  104  000                .ASCII  /ED/<00>
011612  045  101  104     P.AFJ:     .ASCII  /%AD/
011615  105  126  111                .ASCII  /EVI/
011620  103  105  040                .ASCII  /CE /
011623  103  117  115                .ASCII  /COM/
011626  120  101  122                .ASCII  /PAR/
011631  105  040  105                .ASCII  /E E/
011634  122  122  117                .ASCII  /RRO/
011637  122  000  000                .ASCII  /R/<00><00>
011642  045  101  104     P.AFK:     .ASCII  /%AD/
011645  101  124  101                .ASCII  /ATA/
011650  040  105  122                .ASCII  / ER/
011653  122  117  122                .ASCII  /ROR/
011656  000  000                     .ASCII  <00><00>
011660  045  101  110     P.AFL:     .ASCII  /%AH/
011663  117  123  124                .ASCII  /OST/
011666  040  102  125                .ASCII  / BU/
011671  106  106  105                .ASCII  /FFE/
011674  122  040  101                .ASCII  /R A/
011677  103  103  105                .ASCII  /CCE/
011702  123  123  040                .ASCII  /SS /
011705  105  122  122                .ASCII  /ERR/
011710  117  122  000                .ASCII  /OR/<00>
011713  000                         .ASCII  <00>
011714  045  101  103     P.AFM:     .ASCII  /%AC/
011717  117  116  124                .ASCII  /ONT/
011722  122  117  114                .ASCII  /ROL/
011725  114  105  122                .ASCII  /LER/
011730  040  105  122                .ASCII  / ER/
011733  122  117  122                .ASCII  /ROR/
011736  000  000                     .ASCII  <00><00>
011740  045  101  104     P.AFN:     .ASCII  /%AD/
011743  122  111  126                .ASCII  /RIV/
011746  105  040  105                .ASCII  /E E/
011751  122  122  117                .ASCII  /RRO/
011754  122  000                     .ASCII  /R/<00>
011756  045  101  115     P.AFO:     .ASCII  /%AM/
011761  105  123  123                .ASCII  /ESS/
011764  101  107  105                .ASCII  /AGE/
011767  040  106  122                .ASCII  / FR/
011772  117  115  040                .ASCII  /OM /
011775  111  116  124                .ASCII  /INT/
012000  105  122  116                .ASCII  /ERN/
```

```
012003      101      114      040                .ASCII  /AL /
012006      104      111      101                .ASCII  /DIA/
012011      107      116      117                .ASCII  /GNO/
012014      123      124      111                .ASCII  /STI/
012017      103      123      000                .ASCII  /CS/<00>
012022      045      101      110        P.AFP:  .ASCII  /%AH/
012025      117      123      124                .ASCII  /OST/
012030      040      103      117                .ASCII  / CO/
012033      115      120      101                .ASCII  /MPA/
012036      122      105      040                .ASCII  /RE /
012041      105      122      122                .ASCII  /ERR/
012044      117      122      000                .ASCII  /OR/<00>
012047      000                                  .ASCII  <00>
012050      045      101      103        P.AFQ:  .ASCII  /%AC/
012053      117      115      115                .ASCII  /OMM/
012056      101      116      104                .ASCII  /AND/
012061      040      124      111                .ASCII  / TI/
012064      115      105      117                .ASCII  /MEO/
012067      125      124      000                .ASCII  /UT/<00>
012072   011416'                        P.AFC:  .WORD   P.AFD
012074   011440'                                .WORD   P.AFE
012076   011462'                                .WORD   P.AFF
012100   011502'                                .WORD   P.AFG
012102   011542'                                .WORD   P.AFH
012104   011570'                                .WORD   P.AFI
012106   011612'                                .WORD   P.AFJ
012110   011642'                                .WORD   P.AFK
012112   011660'                                .WORD   P.AFL
012114   011714'                                .WORD   P.AFM
012116   011740'                                .WORD   P.AFN
012120   011756'                                .WORD   P.AFO
012122   012022'                                .WORD   P.AFP
012124   012050'                                .WORD   P.AFQ
012126      045      101      105        P.AFR:  .ASCII  /%AE/
012131      122      122      117                .ASCII  /RRO/
012134      122      040      114                .ASCII  /R L/
012137      117      107      040                .ASCII  /OG /
012142      115      105      123                .ASCII  /MES/
012145      123      101      107                .ASCII  /SAG/
012150      105      040      122                .ASCII  /E R/
012153      105      103      105                .ASCII  /ECE/
012156      111      126      105                .ASCII  /IVE/
012161      104      072      045                .ASCII  /D:%/
012164      116      000                         .ASCII  /N/<00>
012166      045      101      052        P.AFT:  .ASCII  /%A*/
012171      040      103      117                .ASCII  / CO/
012174      116      124      122                .ASCII  /NTR/
012177      117      114      114                .ASCII  /OLL/
012202      105      122      040                .ASCII  /ER /
012205      105      122      122                .ASCII  /ERR/
012210      117      122      045                .ASCII  /OR%/
012213      116      000      000                .ASCII  /N/<00><00>
012216      045      101      052        P.AFU:  .ASCII  /%A*/
```

```
012221  040  110  117          .ASCII  / HO/
012224  123  124  040          .ASCII  /ST /
012227  115  105  115          .ASCII  /MEM/
012232  117  122  131          .ASCII  /ORY/
012235  040  101  103          .ASCII  / AC/
012240  103  105  123          .ASCII  /CES/
012243  123  040  105          .ASCII  /S E/
012246  122  122  117          .ASCII  /RRO/
012251  122  045  116          .ASCII  /R%N/
012254  000  000               .ASCII  <00><00>
012256  045  101  052  P.AFV:   .ASCII  /%A*/
012261  040  104  111          .ASCII  / DI/
012264  123  113  045          .ASCII  /SK%/
012267  104  062  045          .ASCII  /D2%/
012272  101  040  055          .ASCII  /A -/
012275  040  104  111          .ASCII  / DI/
012300  123  113  040          .ASCII  /SK /
012303  124  122  101          .ASCII  /TRA/
012306  116  123  106          .ASCII  /NSF/
012311  105  122  040          .ASCII  /ER /
012314  105  122  122          .ASCII  /ERR/
012317  117  122  045          .ASCII  /OR%/
012322  116  000               .ASCII  /N/<00>
012324  045  101  052  P.AFW:   .ASCII  /%A*/
012327  040  104  111          .ASCII  / DI/
012332  123  113  045          .ASCII  /SK%/
012335  104  062  045          .ASCII  /D2%/
012340  101  040  055          .ASCII  /A -/
012343  040  042  123          .ASCII  / "S/
012346  124  101  116          .ASCII  /TAN/
012351  104  101  122          .ASCII  /DAR/
012354  104  040  104          .ASCII  /D D/
012357  111  123  113          .ASCII  /ISK/
012362  040  111  116          .ASCII  / IN/
012365  124  105  122          .ASCII  /TER/
012370  103  117  116          .ASCII  /CON/
012373  116  105  103          .ASCII  /NEC/
012376  124  042  040          .ASCII  /T" /
012401  105  122  122          .ASCII  /ERR/
012404  117  122  045          .ASCII  /OR%/
012407  116  000  000          .ASCII  /N/<00><00>
012412  045  101  052  P.AFX:   .ASCII  /%A*/
012415  040  104  111          .ASCII  / DI/
012420  123  113  045          .ASCII  /SK%/
012423  104  062  045          .ASCII  /D2%/
012426  101  040  055          .ASCII  /A -/
012431  040  042  123          .ASCII  / "S/
012434  115  101  114          .ASCII  /MAL/
012437  114  040  104          .ASCII  /L D/
012442  111  123  113          .ASCII  /ISK/
012445  042  040  105          .ASCII  /" E/
012450  122  122  117          .ASCII  /RRO/
012453  122  045  116          .ASCII  /R%N/
```

```
012456      000     000                     .ASCII  <00><00>
012460   012166'                    P.AFS:  .WORD   P.AFT
012462   012216'                            .WORD   P.AFU
012464   012256'                            .WORD   P.AFV
012466   012324'                            .WORD   P.AFW
012470   012412'                            .WORD   P.AFX
012472      045     116     045     P.AFY:  .ASCII  /%N%/
012475      101     052     040             .ASCII  /A* /
012500      123     101     072             .ASCII  /SA:/
012503      040     045     117             .ASCII  / %O/
012506      066     000                     .ASCII  /6/<00>
012510      045     116     045     P.AFZ:  .ASCII  /%N%/
012513      101     052     040             .ASCII  /A* /
012516      123     124     101             .ASCII  /STA/
012521      124     125     123             .ASCII  /TUS/
012524      040     103     117             .ASCII  / CO/
012527      104     105     072             .ASCII  /DE:/
012532      040     045     117             .ASCII  / %O/
012535      062     000     000             .ASCII  /2/<00><00>
012540      045     117     064     P.AGA:  .ASCII  /%04/
012543      000                             .ASCII  <00>
012544      045     116     045     P.AGB:  .ASCII  /%N%/
012547      101     052     040             .ASCII  /A* /
012552      123     125     102             .ASCII  /SUB/
012555      137     103     117             .ASCII  /_CO/
012560      104     105     072             .ASCII  /DE:/
012563      040     000     000             .ASCII  / /<00><00>
012566      045     116     045     P.AGC:  .ASCII  /%N%/
012571      101     052     040             .ASCII  /A* /
012574      103     117     115             .ASCII  /COM/
012577      115     101     116             .ASCII  /MAN/
012602      104     072     040             .ASCII  /D: /
012605      000                             .ASCII  <00>
012606      045     101     122     P.AGD:  .ASCII  /%AR/
012611      105     101     104             .ASCII  /EAD/
012614      000     000                     .ASCII  <00><00>
012616      045     101     127     P.AGE:  .ASCII  /%AW/
012621      122     111     124             .ASCII  /RIT/
012624      105     000                     .ASCII  /E/<00>
012626      045     101     055     P.AGF:  .ASCII  /%A-/
012631      103     117     115             .ASCII  /COM/
012634      120     101     122             .ASCII  /PAR/
012637      105     000     000             .ASCII  /E/<00><00>
012642      045     101     117     P.AGG:  .ASCII  /%AO/
012645      116     114     111             .ASCII  /NLI/
012650      116     105     000             .ASCII  /NE/<00>
012653      000                             .ASCII  <00>
012654      045     101     101     P.AGH:  .ASCII  /%AA/
012657      103     103     105             .ASCII  /CCE/
012662      123     123     000             .ASCII  /SS/<00>
012665      000                             .ASCII  <00>
012666      045     117     063     P.AGI:  .ASCII  /%03/
012671      000                             .ASCII  <00>
```

```
012672  045  116  045  P.AGJ:  .ASCII  /%N%/
012675  101  052  040         .ASCII  /A* /
012700  102  101  104         .ASCII  /BAD/
012703  040  102  114         .ASCII  / BL/
012706  117  103  113         .ASCII  /OCK/
012711  040  050  110         .ASCII  / (H/
012714  157  163  164         .ASCII  /ost/
012717  040  162  145         .ASCII  / re/
012722  160  154  141         .ASCII  /pla/
012725  143  145  141         .ASCII  /cea/
012730  142  154  145         .ASCII  /ble/
012733  051  072  040         .ASCII  /): /
012736  045  104  065         .ASCII  /%D5/
012741  045  101  056         .ASCII  /%A./
012744  040  050  117         .ASCII  / (O/
012747  103  124  040         .ASCII  /CT /
012752  045  117  066         .ASCII  /%O6/
012755  045  101  051         .ASCII  /%A)/
012760  000  000             .ASCII  <00><00>
012762  045  116  045  P.AGK:  .ASCII  /%N%/
012765  101  052  040         .ASCII  /A* /
012770  061  163  164         .ASCII  /1st/
012773  040  102  101         .ASCII  / BA/
012776  104  040  102         .ASCII  /D B/
013001  114  117  103         .ASCII  /LOC/
013004  113  040  050         .ASCII  /K (/
013007  110  157  163         .ASCII  /Hos/
013012  164  040  162         .ASCII  /t r/
013015  145  160  154         .ASCII  /epl/
013020  141  143  145         .ASCII  /ace/
013023  141  142  154         .ASCII  /abl/
013026  145  051  072         .ASCII  /e):/
013031  040  045  104         .ASCII  / %D/
013034  065  045  101         .ASCII  /5%A/
013037  056  040  050         .ASCII  /. (/
013042  117  103  124         .ASCII  /OCT/
013045  040  045  117         .ASCII  / %O/
013050  066  045  101         .ASCII  /6%A/
013053  051  000  000         .ASCII  /)/<00><00>
013056  045  116  045  P.AGL:  .ASCII  /%N%/
013061  101  052  040         .ASCII  /A* /
013064  102  101  104         .ASCII  /BAD/
013067  040  102  114         .ASCII  / BL/
013072  117  103  113         .ASCII  /OCK/
013075  040  122  105         .ASCII  / RE/
013100  120  117  122         .ASCII  /POR/
013103  124  105  104         .ASCII  /TED/
013106  040  050  122         .ASCII  / (R/
013111  145  160  154         .ASCII  /epl/
013114  141  143  145         .ASCII  /ace/
013117  144  051  072         .ASCII  /d):/
013122  040  045  104         .ASCII  / %D/
013125  045  101  056         .ASCII  /%A./
```

```
013130  040  050  117            .ASCII  / (O/
013133  103  124  040            .ASCII  /CT /
013136  045  117  066            .ASCII  /%O6/
013141  045  101  051            .ASCII  /%A)/
013144  000  000                 .ASCII  <00><00>
013146  045  116  045  P.AGM:     .ASCII  /%N%/
013151  101  052  040            .ASCII  /A* /
013154  114  102  116            .ASCII  /LBN/
013157  072  040  045            .ASCII  /: %/
013162  104  065  045            .ASCII  /D5%/
013165  101  056  040            .ASCII  /A. /
013170  050  117  103            .ASCII  /(OC/
013173  124  040  045            .ASCII  /T %/
013176  117  066  045            .ASCII  /O6%/
013201  101  051  000            .ASCII  /A)/<00>
013204  045  116  045  P.AGN:     .ASCII  /%N%/
013207  101  052  040            .ASCII  /A* /
013212  120  102  116            .ASCII  /PBN/
013215  072  040  045            .ASCII  /: %/
013220  104  065  045            .ASCII  /D5%/
013223  101  056  040            .ASCII  /A. /
013226  050  117  103            .ASCII  /(OC/
013231  124  040  045            .ASCII  /T %/
013234  117  066  045            .ASCII  /O6%/
013237  101  051  000            .ASCII  /A)/<00>
013242  045  116  045  P.AGO:     .ASCII  /%N%/
013245  101  052  040            .ASCII  /A* /
013250  114  102  116            .ASCII  /LBN/
013253  072  040  050            .ASCII  /: (/
013256  122  105  101            .ASCII  /REA/
013261  104  051  040            .ASCII  /D) /
013264  045  104  065            .ASCII  /%D5/
013267  045  101  056            .ASCII  /%A./
013272  040  050  117            .ASCII  / (O/
013275  103  124  040            .ASCII  /CT /
013300  045  117  066            .ASCII  /%O6/
013303  045  101  051            .ASCII  /%A)/
013306  000  000                 .ASCII  <00><00>
013310  045  116  045  P.AGP:     .ASCII  /%N%/
013313  101  052  040            .ASCII  /A* /
013316  114  102  116            .ASCII  /LBN/
013321  072  040  050            .ASCII  /: (/
013324  127  122  111            .ASCII  /WRI/
013327  124  105  051            .ASCII  /TE)/
013332  040  045  104            .ASCII  / %D/
013335  065  045  101            .ASCII  /5%A/
013340  056  040  050            .ASCII  /. (/
013343  117  103  124            .ASCII  /OCT/
013346  040  045  117            .ASCII  / %O/
013351  066  045  101            .ASCII  /6%A/
013354  051  000                 .ASCII  /)/<00>
013356  045  116  045  P.AGQ:     .ASCII  /%N%/
013361  101  052  040            .ASCII  /A* /
```

```
013364  122  105  120                .ASCII  /REP/
013367  114  101  103                .ASCII  /LAC/
013372  105  115  105                .ASCII  /EME/
013375  116  124  040                .ASCII  /NT /
013400  102  114  117                .ASCII  /BLO/
013403  103  113  040                .ASCII  /CK /
013406  116  117  056                .ASCII  /NO./
013411  040  045  104                .ASCII  / %D/
013414  065  045  101                .ASCII  /5%A/
013417  056  040  050                .ASCII  /. (/
013422  117  103  124                .ASCII  /OCT/
013425  040  045  117                .ASCII  / %O/
013430  066  045  101                .ASCII  /6%A/
013433  051  000  000                .ASCII  /)/<00><00>
013436  045  116  045      P.AGR:    .ASCII  /%N%/
013441  101  052  040                .ASCII  /A* /
013444  102  131  124                .ASCII  /BYT/
013447  105  040  103                .ASCII  /E C/
013452  117  125  116                .ASCII  /OUN/
013455  124  040  111                .ASCII  /T I/
013460  116  040  103                .ASCII  /N C/
013463  117  115  115                .ASCII  /OMM/
013466  101  116  104                .ASCII  /AND/
013471  072  040  045                .ASCII  /: %/
013474  104  065  045                .ASCII  /D5%/
013477  101  056  000                .ASCII  /A./<00>
013502  045  116  045      P.AGS:    .ASCII  /%N%/
013505  101  052  040                .ASCII  /A* /
013510  102  131  124                .ASCII  /BYT/
013513  105  040  103                .ASCII  /E C/
013516  117  125  116                .ASCII  /OUN/
013521  124  040  111                .ASCII  /T I/
013524  116  040  122                .ASCII  /N R/
013527  105  101  104                .ASCII  /EAD/
013532  040  103  117                .ASCII  / CO/
013535  115  115  101                .ASCII  /MMA/
013540  116  104  072                .ASCII  /ND:/
013543  040  045  104                .ASCII  / %D/
013546  065  045  101                .ASCII  /5%A/
013551  056  000  000                .ASCII  /./<00><00>
013554  045  116  045      P.AGT:    .ASCII  /%N%/
013557  101  052  040                .ASCII  /A* /
013562  102  131  124                .ASCII  /BYT/
013565  105  040  103                .ASCII  /E C/
013570  117  125  116                .ASCII  /OUN/
013573  124  040  111                .ASCII  /T I/
013576  116  040  127                .ASCII  /N W/
013601  122  111  124                .ASCII  /RIT/
013604  105  040  103                .ASCII  /E C/
013607  117  115  115                .ASCII  /OMM/
013612  101  116  104                .ASCII  /AND/
013615  072  040  045                .ASCII  /: %/
013620  104  065  045                .ASCII  /D5%/
```

```
013623   101   056   000                .ASCII  /A./<00>
013626   045   116   045        P.AGU:  .ASCII  /%N%/
013631   101   052   040                .ASCII  /A* /
013634   101   103   124                .ASCII  /ACT/
013637   125   101   114                .ASCII  /UAL/
013642   040   043   040                .ASCII  / # /
013645   117   106   040                .ASCII  /OF /
013650   102   131   124                .ASCII  /BYT/
013653   105   123   040                .ASCII  /ES /
013656   124   122   101                .ASCII  /TRA/
013661   116   123   106                .ASCII  /NSF/
013664   105   122   122                .ASCII  /ERR/
013667   105   104   072                .ASCII  /ED:/
013672   040   045   104                .ASCII  / %D/
013675   065   045   101                .ASCII  /5%A/
013700   056   000                      .ASCII  /./<00>
013702   045   116   045        P.AGV:  .ASCII  /%N%/
013705   101   052   040                .ASCII  /A* /
013710   111   057   117                .ASCII  /I/<57>/O/
013713   040   102   125                .ASCII  / BU/
013716   106   106   105                .ASCII  /FFE/
013721   122   040   101                .ASCII  /R A/
013724   104   104   122                .ASCII  /DDR/
013727   105   123   123                .ASCII  /ESS/
013732   040   050   063                .ASCII  / (3/
013735   062   040   142                .ASCII  /2 b/
013740   151   164   163                .ASCII  /its/
013743   051   072   040                .ASCII  /): /
013746   045   117   066                .ASCII  /%O6/
013751   045   101   040                .ASCII  /%A /
013754   045   117   066                .ASCII  /%O6/
013757   000                            .ASCII  <00>
013760   045   116   045        P.AGW:  .ASCII  /%N%/
013763   101   052   040                .ASCII  /A* /
013766   111   057   117                .ASCII  /I/<57>/O/
013771   040   102   125                .ASCII  / BU/
013774   106   106   105                .ASCII  /FFE/
013777   122   040   101                .ASCII  /R A/
014002   104   104   122                .ASCII  /DDR/
014005   105   123   123                .ASCII  /ESS/
014010   040   106   117                .ASCII  / FO/
014013   122   040   122                .ASCII  /R R/
014016   105   101   104                .ASCII  /EAD/
014021   040   050   063                .ASCII  / (3/
014024   062   040   142                .ASCII  /2 b/
014027   151   164   163                .ASCII  /its/
014032   051   072   040                .ASCII  /): /
014035   045   117   066                .ASCII  /%O6/
014040   045   101   040                .ASCII  /%A /
014043   045   117   066                .ASCII  /%O6/
014046   000   000                      .ASCII  <00><00>
014050   045   116   045        P.AGX:  .ASCII  /%N%/
014053   101   052   040                .ASCII  /A* /
```

```
014056  111  057  117                .ASCII  /I/<57>/O/
014061  040  102  125                .ASCII  / BU/
014064  106  106  105                .ASCII  /FFE/
014067  122  040  101                .ASCII  /R A/
014072  104  104  122                .ASCII  /DDR/
014075  105  123  123                .ASCII  /ESS/
014100  040  106  117                .ASCII  / FO/
014103  122  040  127                .ASCII  /R W/
014106  122  111  124                .ASCII  /RIT/
014111  105  040  050                .ASCII  /E (/
014114  063  062  040                .ASCII  /32 /
014117  142  151  164                .ASCII  /bit/
014122  163  051  072                .ASCII  /s):/
014125  040  045  117                .ASCII  / %O/
014130  066  045  101                .ASCII  /6%A/
014133  040  045  117                .ASCII  / %O/
014136  066  000                    .ASCII  /6/<00>
014140  045  116  045       P.AGY:   .ASCII  /%N%/
014143  101  103  117                .ASCII  /ACO/
014146  116  124  105                .ASCII  /NTE/
014151  116  124  123                .ASCII  /NTS/
014154  040  117  106                .ASCII  / OF/
014157  040  103  117                .ASCII  / CO/
014162  115  115  101                .ASCII  /MMA/
014165  116  104  057                .ASCII  /ND/<57>
014170  122  105  123                .ASCII  /RES/
014173  120  117  116                .ASCII  /PON/
014176  123  105  040                .ASCII  /SE /
014201  120  101  103                .ASCII  /PAC/
014204  113  105  124                .ASCII  /KET/
014207  040  123  101                .ASCII  / SA/
014212  126  105  040                .ASCII  /VE /
014215  101  122  105                .ASCII  /ARE/
014220  101  072  045                .ASCII  /A:%/
014223  116  000  000                .ASCII  /N/<00><00>
014226  045  101  040       P.AGZ:   .ASCII  /%A /
014231  045  117  066                .ASCII  /%06/
014234  000  000                    .ASCII  <00><00>
014236  045  116  045       P.AHA:   .ASCII  /%N%/
014241  101  052  040                .ASCII  /A* /
014244  124  111  115                .ASCII  /TIM/
014247  105  072  040                .ASCII  /E: /
014252  045  132  062                .ASCII  /%Z2/
014255  045  101  072                .ASCII  /%A:/
014260  045  132  062                .ASCII  /%Z2/
014263  045  101  040                .ASCII  /%A /
014266  110  117  125                .ASCII  /HOU/
014271  122  123  045                .ASCII  /RS%/
014274  116  000                    .ASCII  /N/<00>
014276  045  116  045       P.AHB:   .ASCII  /%N%/
014301  101  040  052                .ASCII  /A *
014304  040  104  111                .ASCII  / DI/
014307  123  113  040                .ASCII  /SK /
```

```
014312  072     040     045             .ASCII  /: %/
014315  104     062     000             .ASCII  /D2/<00>
014320  045     116     045     P.AHC:  .ASCII  /%N%/
014323  101     104     102             .ASCII  /ADB/
014326  116     072     040             .ASCII  /N: /
014331  045     104     065             .ASCII  /%D5/
014334  045     101     056             .ASCII  /%A./
014337  040     050     117             .ASCII  / (O/
014342  103     124     040             .ASCII  /CT /
014345  045     117     066             .ASCII  /%O6/
014350  045     101     051             .ASCII  /%A)/
014353  000                             .ASCII  <00>
014354  045     116     045     P.AHD:  .ASCII  /%N%/
014357  101     102     131             .ASCII  /ABY/
014362  124     105     040             .ASCII  /TE /
014365  116     125     115             .ASCII  /NUM/
014370  102     105     122             .ASCII  /BER/
014373  072     040     045             .ASCII  /: %/
014376  104     063     000             .ASCII  /D3/<00>
014401  000                             .ASCII  <00>
014402  045     116     045     P.AHE:  .ASCII  /%N%/
014405  101     122     101             .ASCII  /ARA/
014410  116     104     117             .ASCII  /NDO/
014413  115     040     127             .ASCII  /M W/
014416  122     111     124             .ASCII  /RIT/
014421  124     105     116             .ASCII  /TEN/
014424  040     127     117             .ASCII  / WO/
014427  122     104     040             .ASCII  /RD /
014432  072     045     102             .ASCII  /:%B/
014435  061     066     000             .ASCII  /16/<00>
014440  045     116     045     P.AHF:  .ASCII  /%N%/
014443  101     122     101             .ASCII  /ARA/
014446  116     104     117             .ASCII  /NDO/
014451  115     040     122             .ASCII  /M R/
014454  105     101     104             .ASCII  /EAD/
014457  040     127     117             .ASCII  / WO/
014462  122     104     040             .ASCII  /RD /
014465  142     151     156             .ASCII  /bin/
014470  072     045     102             .ASCII  /:%B/
014473  061     066     045             .ASCII  /16%/
014476  101     040     157             .ASCII  /A o/
014501  143     164     072             .ASCII  /ct:/
014504  045     117     066             .ASCII  /%O6/
014507  000                             .ASCII  <00>
014510  045     116     045     P.AHG:  .ASCII  /%N%/
014513  101     104     125             .ASCII  /ADU/
014516  120     114     111             .ASCII  /PLI/
014521  103     101     124             .ASCII  /CAT/
014524  105     040     125             .ASCII  /E U/
014527  116     111     124             .ASCII  /NIT/
014532  072     045     104             .ASCII  /:%D/
014535  062     045     101             .ASCII  /2%A/
014540  040     101     124             .ASCII  / AT/
```

```
014543  040  111  120                .ASCII  / IP/
014546  072  040  045                .ASCII  /: %/
014551  117  066  000                .ASCII  /06/<00>
014554  045  116  045  P.AHH:        .ASCII  /%N%/
014557  101  115  117                .ASCII  /AMO/
014562  122  105  040                .ASCII  /RE /
014565  124  110  101                .ASCII  /THA/
014570  116  040  045                .ASCII  /N %/
014573  104  061  045                .ASCII  /D1%/
014576  101  040  104                .ASCII  /A D/
014601  111  106  106                .ASCII  /IFF/
014604  105  122  105                .ASCII  /ERE/
014607  116  124  040                .ASCII  /NT /
014612  111  120  040                .ASCII  /IP /
014615  101  104  104                .ASCII  /ADD/
014620  122  105  123                .ASCII  /RES/
014623  123  105  123                .ASCII  /SES/
014626  000  000                     .ASCII  <00><00>
014630  045  101  123  P.AHI:        .ASCII  /%AS/
014633  120  111  116                .ASCII  /PIN/
014636  055  104  117                .ASCII  /-DO/
014641  127  116  040                .ASCII  /WN /
014644  111  107  116                .ASCII  /IGN/
014647  117  122  105                .ASCII  /ORE/
014652  104  000                     .ASCII  /D/<00>
014654  045  101  123  P.AHJ:        .ASCII  /%AS/
014657  124  111  114                .ASCII  /TIL/
014662  114  040  103                .ASCII  /L C/
014665  117  116  116                .ASCII  /ONN/
014670  105  103  124                .ASCII  /ECT/
014673  105  104  000                .ASCII  /ED/<00>
014676  045  101  104  P.AHK:        .ASCII  /%AD/
014701  125  120  114                .ASCII  /UPL/
014704  111  103  101                .ASCII  /ICA/
014707  124  105  040                .ASCII  /TE /
014712  125  116  111                .ASCII  /UNI/
014715  124  040  116                .ASCII  /T N/
014720  125  115  102                .ASCII  /UMB/
014723  105  122  000                .ASCII  /ER/<00>
014726  045  101  101  P.AHL:        .ASCII  /%AA/
014731  114  122  105                .ASCII  /LRE/
014734  101  104  131                .ASCII  /ADY/
014737  040  117  116                .ASCII  / ON/
014742  114  111  116                .ASCII  /LIN/
014745  105  000  000                .ASCII  /E/<00><00>
014750  045  101  123  P.AHM:        .ASCII  /%AS/
014753  124  111  114                .ASCII  /TIL/
014756  114  040  117                .ASCII  /L O/
014761  116  114  111                .ASCII  /NLI/
014764  116  105  000                .ASCII  /NE/<00>
014767  000                         .ASCII  <00>
014770  045  101  125  P.AHN:        .ASCII  /%AU/
014773  116  111  124                .ASCII  /NIT/
```

```
014776  040     125     116             .ASCII  / UN/
015001  113     116     117             .ASCII  /KNO/
015004  127     116     040             .ASCII  /WN /
015007  117     122     040             .ASCII  /OR /
015012  117     116     114             .ASCII  /ONL/
015015  111     116     105             .ASCII  /INE/
015020  040     124     117             .ASCII  / TO/
015023  040     101     116             .ASCII  / AN/
015026  117     124     110             .ASCII  /OTH/
015031  105     122     040             .ASCII  /ER /
015034  103     117     116             .ASCII  /CON/
015037  124     122     117             .ASCII  /TRO/
015042  114     114     105             .ASCII  /LLE/
015045  122     000     000             .ASCII  /R/<00><00>
015050  045     101     116     P.AHO:  .ASCII  /%AN/
015053  117     040     126             .ASCII  /O V/
015056  117     114     125             .ASCII  /OLU/
015061  115     105     040             .ASCII  /ME /
015064  115     117     125             .ASCII  /MOU/
015067  116     124     105             .ASCII  /NTE/
015072  104     040     117             .ASCII  /D O/
015075  122     040     104             .ASCII  /R D/
015100  122     111     126             .ASCII  /RIV/
015103  105     040     104             .ASCII  /E D/
015106  111     123     101             .ASCII  /ISA/
015111  102     114     105             .ASCII  /BLE/
015114  104     040     102             .ASCII  /D B/
015117  131     040     123             .ASCII  /Y S/
015122  127     111     124             .ASCII  /WIT/
015125  103     110     000             .ASCII  /CH/<00>
015130  045     101     125     P.AHP:  .ASCII  /%AU/
015133  116     111     124             .ASCII  /NIT/
015136  040     111     116             .ASCII  / IN/
015141  117     120     105             .ASCII  /OPE/
015144  122     101     124             .ASCII  /RAT/
015147  111     126     105             .ASCII  /IVE/
015152  040     050     122             .ASCII  / (R/
015155  104     065     061             .ASCII  /D51/
015160  057     065     062             .ASCII  <57>/52/
015163  040     167     162             .ASCII  / wr/
015166  151     164     145             .ASCII  /ite/
015171  040     146     141             .ASCII  / fa/
015174  165     154     164             .ASCII  /ult/
015177  051     000     000             .ASCII  /)/<00><00>
015202  045     101     125     P.AHQ:  .ASCII  /%AU/
015205  116     111     124             .ASCII  /NIT/
015210  040     104     111             .ASCII  / DI/
015213  123     101     102             .ASCII  /SAB/
015216  114     105     104             .ASCII  /LED/
015221  040     102     131             .ASCII  / BY/
015224  040     106     111             .ASCII  / FI/
015227  105     114     104             .ASCII  /ELD/
015232  040     123     105             .ASCII  / SE/
```

```
015235  122  126  111          .ASCII  /RVI/
015240  103  105  040          .ASCII  /CE /
015243  117  122  040          .ASCII  /OR /
015246  111  116  124          .ASCII  /INT/
015251  105  122  116          .ASCII  /ERN/
015254  101  114  040          .ASCII  /AL /
015257  104  111  101          .ASCII  /DIA/
015262  107  116  117          .ASCII  /GNO/
015265  123  124  111          .ASCII  /STI/
015270  103  123  000          .ASCII  /CS/<00>
015273  000                   .ASCII  <00>
015274  045  101  042  P.AHR:  .ASCII  /%A"/
015277  106  117  122          .ASCII  /FOR/
015302  103  105  104          .ASCII  /CED/
015305  040  105  122          .ASCII  / ER/
015310  122  117  122          .ASCII  /ROR/
015313  042  040  104          .ASCII  /" D/
015316  105  124  105          .ASCII  /ETE/
015321  103  124  105          .ASCII  /CTE/
015324  104  040  127          .ASCII  /D W/
015327  110  111  114          .ASCII  /HIL/
015332  105  040  101          .ASCII  /E A/
015335  103  103  105          .ASCII  /CCE/
015340  123  123  111          .ASCII  /SSI/
015343  116  107  040          .ASCII  /NG /
015346  106  103  124          .ASCII  /FCT/
015351  040  117  122          .ASCII  / OR/
015354  040  122  103          .ASCII  / RC/
015357  124  000  000          .ASCII  /T/<00><00>
015362  045  101  123  P.AHS:  .ASCII  /%AS/
015365  105  103  124          .ASCII  /ECT/
015370  117  122  040          .ASCII  /OR /
015373  110  101  104          .ASCII  /HAD/
015376  040  102  105          .ASCII  / BE/
015401  105  116  040          .ASCII  /EN /
015404  127  122  111          .ASCII  /WRI/
015407  124  124  105          .ASCII  /TTE/
015412  116  040  127          .ASCII  /N W/
015415  111  124  110          .ASCII  /ITH/
015420  040  042  106          .ASCII  / "F/
015423  117  122  103          .ASCII  /ORC/
015426  105  104  040          .ASCII  /ED /
015431  105  122  122          .ASCII  /ERR/
015434  117  122  042          .ASCII  /OR"/
015437  040  115  117          .ASCII  / MO/
015442  104  111  106          .ASCII  /DIF/
015445  111  105  122          .ASCII  /IER/
015450  000  000               .ASCII  <00><00>
015452  045  101  106  P.AHT:  .ASCII  /%AF/
015455  103  124  040          .ASCII  /CT /
015460  117  122  040          .ASCII  /OR /
015463  122  103  124          .ASCII  /RCT/
015466  040  125  116          .ASCII  / UN/
```

```
015471  122  105  101            .ASCII  /REA/
015474  104  101  102            .ASCII  /DAB/
015477  114  105  040            .ASCII  /LE /
015502  055  040  111            .ASCII  /- I/
015505  116  126  101            .ASCII  /NVA/
015510  114  111  104            .ASCII  /LID/
015513  040  123  105            .ASCII  / SE/
015516  103  124  117            .ASCII  /CTO/
015521  122  040  110            .ASCII  /R H/
015524  105  101  104            .ASCII  /EAD/
015527  105  122  000            .ASCII  /ER/<00>
015532  045  101  110  P.AHU:    .ASCII  /%AH/
015535  105  101  104            .ASCII  /EAD/
015540  105  122  040            .ASCII  /ER /
015543  103  117  115            .ASCII  /COM/
015546  120  101  122            .ASCII  /PAR/
015551  105  040  105            .ASCII  /E E/
015554  122  122  117            .ASCII  /RRO/
015557  122  040  050            .ASCII  /R (/
015562  126  141  154            .ASCII  /Val/
015565  151  144  040            .ASCII  /id /
015570  150  145  141            .ASCII  /hea/
015573  144  145  162            .ASCII  /der/
015576  040  156  157            .ASCII  / no/
015601  164  040  146            .ASCII  /t f/
015604  157  165  156            .ASCII  /oun/
015607  144  051  000            .ASCII  /d)/<00>
015612  045  101  106  P.AHV:    .ASCII  /%AF/
015615  103  124  040            .ASCII  /CT /
015620  117  122  040            .ASCII  /OR /
015623  122  103  124            .ASCII  /RCT/
015626  040  125  116            .ASCII  / UN/
015631  122  105  101            .ASCII  /REA/
015634  104  101  102            .ASCII  /DAB/
015637  114  105  040            .ASCII  /LE /
015642  055  040  104            .ASCII  /- D/
015645  101  124  101            .ASCII  /ATA/
015650  040  123  131            .ASCII  / SY/
015653  116  103  040            .ASCII  /NC /
015656  124  111  115            .ASCII  /TIM/
015661  105  117  125            .ASCII  /EOU/
015664  124  000                 .ASCII  /T/<00>
015666  045  101  104  P.AHW:    .ASCII  /%AD/
015671  101  124  101            .ASCII  /ATA/
015674  040  123  131            .ASCII  / SY/
015677  116  103  040            .ASCII  /NC /
015702  116  117  124            .ASCII  /NOT/
015705  040  106  117            .ASCII  / FO/
015710  125  116  104            .ASCII  /UND/
015713  040  050  104            .ASCII  / (D/
015716  141  164  141            .ASCII  /ata/
015721  040  163  171            .ASCII  / sy/
015724  156  143  040            .ASCII  /nc /
```

```
015727  164     151     155             .ASCII  /tim/
015732  145     157     165             .ASCII  /eou/
015735  164     051     000             .ASCII  /t)/<00>
015740  045     101     106     P.AHX:  .ASCII  /%AF/
015743  103     124     040             .ASCII  /CT /
015746  117     122     040             .ASCII  /OR /
015751  122     103     124             .ASCII  /RCT/
015754  040     125     116             .ASCII  / UN/
015757  122     105     101             .ASCII  /REA/
015762  104     101     102             .ASCII  /DAB/
015765  114     105     040             .ASCII  /LE /
015770  055     040     125             .ASCII  /- U/
015773  116     103     117             .ASCII  /NCO/
015776  122     122     105             .ASCII  /RRE/
016001  103     124     101             .ASCII  /CTA/
016004  102     114     105             .ASCII  /BLE/
016007  040     105     103             .ASCII  / EC/
016012  103     040     105             .ASCII  /C E/
016015  122     122     117             .ASCII  /RRO/
016020  122     000                     .ASCII  /R/<00>
016022  045     101     125     P.AHY:  .ASCII  /%AU/
016025  116     103     117             .ASCII  /NCO/
016030  122     122     105             .ASCII  /RRE/
016033  103     124     101             .ASCII  /CTA/
016036  102     114     105             .ASCII  /BLE/
016041  040     105     103             .ASCII  / EC/
016044  103     040     105             .ASCII  /C E/
016047  122     122     117             .ASCII  /RRO/
016052  122     000                     .ASCII  /R/<00>
016054  045     101     122     P.AHZ:  .ASCII  /%AR/
016057  103     124     040             .ASCII  /CT /
016062  103     117     122             .ASCII  /COR/
016065  122     125     120             .ASCII  /RUP/
016070  124     105     104             .ASCII  /TED/
016073  000                             .ASCII  <00>
016074  045     101     116     P.AIA:  .ASCII  /%AN/
016077  117     040     122             .ASCII  /O R/
016102  105     120     114             .ASCII  /EPL/
016105  101     103     105             .ASCII  /ACE/
016110  115     105     116             .ASCII  /MEN/
016113  124     040     102             .ASCII  /T B/
016116  114     117     103             .ASCII  /LOC/
016121  113     040     101             .ASCII  /K A/
016124  126     101     111             .ASCII  /VAI/
016127  114     101     102             .ASCII  /LAB/
016132  114     105     040             .ASCII  /LE /
016135  050     122     103             .ASCII  /(RC/
016140  124     040     146             .ASCII  /T f/
016143  165     154     154             .ASCII  /ull/
016146  051     000                     .ASCII  /)/<00>
016150  045     101     104     P.AIB:  .ASCII  /%AD/
016153  111     123     113             .ASCII  /ISK/
016156  040     116     117             .ASCII  / NO/
```

```
016161  124  040  106            .ASCII  /T F/
016164  117  122  115            .ASCII  /ORM/
016167  101  124  124            .ASCII  /ATT/
016172  105  104  040            .ASCII  /ED /
016175  127  111  124            .ASCII  /WIT/
016200  110  040  065            .ASCII  /H 5/
016203  061  062  040            .ASCII  /12 /
016206  102  131  124            .ASCII  /BYT/
016211  105  040  123            .ASCII  /E S/
016214  105  103  124            .ASCII  /ECT/
016217  117  122  123            .ASCII  /ORS/
016222  000  000                 .ASCII  <00><00>
016224  045  101  104       P.AIC: .ASCII  /%AD/
016227  111  123  113            .ASCII  /ISK/
016232  040  116  117            .ASCII  / NO/
016235  124  040  106            .ASCII  /T F/
016240  117  122  115            .ASCII  /ORM/
016243  101  124  124            .ASCII  /ATT/
016246  105  104  040            .ASCII  /ED /
016251  117  122  040            .ASCII  /OR /
016254  106  103  124            .ASCII  /FCT/
016257  040  103  117            .ASCII  / CO/
016262  122  122  125            .ASCII  /RRU/
016265  120  124  105            .ASCII  /PTE/
016270  104  000                 .ASCII  /D/<00>
016272  045  101  117       P.AID: .ASCII  /%AO/
016275  116  105  040            .ASCII  /NE /
016300  123  131  115            .ASCII  /SYM/
016303  102  117  114            .ASCII  /BOL/
016306  040  105  103            .ASCII  / EC/
016311  103  040  105            .ASCII  /C E/
016314  122  122  117            .ASCII  /RRO/
016317  122  000  000            .ASCII  /R/<00><00>
016322  045  101  124       P.AIE: .ASCII  /%AT/
016325  127  117  040            .ASCII  /WO /
016330  123  131  115            .ASCII  /SYM/
016333  102  117  114            .ASCII  /BOL/
016336  040  105  103            .ASCII  / EC/
016341  103  040  105            .ASCII  /C E/
016344  122  122  117            .ASCII  /RRO/
016347  122  000  000            .ASCII  /R/<00><00>
016352  045  101  124       P.AIF: .ASCII  /%AT/
016355  110  122  105            .ASCII  /HRE/
016360  105  040  123            .ASCII  /E S/
016363  131  115  102            .ASCII  /YMB/
016366  117  114  040            .ASCII  /OL /
016371  105  103  103            .ASCII  /ECC/
016374  040  105  122            .ASCII  / ER/
016377  122  117  122            .ASCII  /ROR/
016402  000  000                 .ASCII  <00><00>
016404  045  101  106       P.AIG: .ASCII  /%AF/
016407  117  125  122            .ASCII  /OUR/
016412  040  123  131            .ASCII  / SY/
```

```
016415  115     102     117             .ASCII  /MBO/
016420  114     040     105             .ASCII  /L E/
016423  103     103     040             .ASCII  /CC /
016426  105     122     122             .ASCII  /ERR/
016431  117     122     000             .ASCII  /OR/<00>
016434  045     101     106     P.AIH:  .ASCII  /%AF/
016437  111     126     105             .ASCII  /IVE/
016442  040     123     131             .ASCII  / SY/
016445  115     102     117             .ASCII  /MBO/
016450  114     040     105             .ASCII  /L E/
016453  103     103     040             .ASCII  /CC /
016456  105     122     122             .ASCII  /ERR/
016461  117     122     000             .ASCII  /OR/<00>
016464  045     101     123     P.AII:  .ASCII  /%AS/
016467  111     130     040             .ASCII  /IX /
016472  123     131     115             .ASCII  /SYM/
016475  102     117     114             .ASCII  /BOL/
016500  040     105     103             .ASCII  / EC/
016503  103     040     105             .ASCII  /C E/
016506  122     122     117             .ASCII  /RRO/
016511  122     000     000             .ASCII  /R/<00><00>
016514  045     101     123     P.AIJ:  .ASCII  /%AS/
016517  105     126     105             .ASCII  /EVE/
016522  116     040     123             .ASCII  /N S/
016525  131     115     102             .ASCII  /YMB/
016530  117     114     040             .ASCII  /OL /
016533  105     103     103             .ASCII  /ECC/
016536  040     105     122             .ASCII  / ER/
016541  122     117     122             .ASCII  /ROR/
016544  000     000                     .ASCII  <00><00>
016546  045     101     105     P.AIK:  .ASCII  /%AE/
016551  111     107     110             .ASCII  /IGH/
016554  124     040     123             .ASCII  /T S/
016557  131     115     102             .ASCII  /YMB/
016562  117     114     040             .ASCII  /OL /
016565  105     103     103             .ASCII  /ECC/
016570  040     105     122             .ASCII  / ER/
016573  122     117     122             .ASCII  /ROR/
016576  000     000                     .ASCII  <00><00>
016600  045     101     103     P.AIL:  .ASCII  /%AC/
016603  117     122     122             .ASCII  /ORR/
016606  105     103     124             .ASCII  /ECT/
016611  101     102     114             .ASCII  /ABL/
016614  105     040     105             .ASCII  /E E/
016617  122     122     117             .ASCII  /RRO/
016622  122     040     111             .ASCII  /R I/
016625  116     040     105             .ASCII  /N E/
016630  103     103     040             .ASCII  /CC /
016633  106     111     105             .ASCII  /FIE/
016636  114     104     000             .ASCII  /LD/<00>
016641  000                             .ASCII  <00>
016642  045     101     125     P.AIM:  .ASCII  /%AU/
016645  116     111     124             .ASCII  /NIT/
```

```
016650  040  123  117          .ASCII  / SO/
016653  106  124  127          .ASCII  /FTW/
016656  101  122  105          .ASCII  /ARE/
016661  040  127  122          .ASCII  / WR/
016664  111  124  105          .ASCII  /ITE/
016667  040  120  122          .ASCII  / PR/
016672  117  124  105          .ASCII  /OTE/
016675  103  124  105          .ASCII  /CTE/
016700  104  000               .ASCII  /D/<00>
016702  045  101  125  P.AIN:   .ASCII  /%AU/
016705  116  111  124          .ASCII  /NIT/
016710  040  110  101          .ASCII  / HA/
016713  122  104  127          .ASCII  /RDW/
016716  101  122  105          .ASCII  /ARE/
016721  040  127  122          .ASCII  / WR/
016724  111  124  105          .ASCII  /ITE/
016727  040  120  122          .ASCII  / PR/
016732  117  124  105          .ASCII  /OTE/
016735  103  124  105          .ASCII  /CTE/
016740  104  000               .ASCII  /D/<00>
016742  045  101  117  P.AIO:   .ASCII  /%AO/
016745  104  104  040          .ASCII  /DD /
016750  124  122  101          .ASCII  /TRA/
016753  116  123  106          .ASCII  /NSF/
016756  105  122  040          .ASCII  /ER /
016761  101  104  104          .ASCII  /ADD/
016764  122  105  123          .ASCII  /RES/
016767  123  000  000          .ASCII  /S/<00><00>
016772  045  101  117  P.AIP:   .ASCII  /%AO/
016775  104  104  040          .ASCII  /DD /
017000  102  131  124          .ASCII  /BYT/
017003  105  040  103          .ASCII  /E C/
017006  117  125  116          .ASCII  /OUN/
017011  124  000  000          .ASCII  /T/<00><00>
017014  045  101  116  P.AIQ:   .ASCII  /%AN/
017017  117  116  055          .ASCII  /ON-/
017022  105  130  111          .ASCII  /EXI/
017025  123  124  105          .ASCII  /STE/
017030  116  124  040          .ASCII  /NT /
017033  110  117  123          .ASCII  /HOS/
017036  124  040  115          .ASCII  /T M/
017041  105  115  117          .ASCII  /EMO/
017044  122  131  000          .ASCII  /RY/<00>
017047  000                    .ASCII  <00>
017050  045  101  110  P.AIR:   .ASCII  /%AH/
017053  117  123  124          .ASCII  /OST/
017056  040  115  105          .ASCII  / ME/
017061  115  117  122          .ASCII  /MOR/
017064  131  040  120          .ASCII  /Y P/
017067  101  122  111          .ASCII  /ARI/
017072  124  131  040          .ASCII  /TY /
017075  105  122  122          .ASCII  /ERR/
017100  117  122  000          .ASCII  /OR/<00>
```

```
017103  000                             .ASCII  <00>
017104  045     101     103     P.AIS:  .ASCII  /≠AC/
017107  117     115     115             .ASCII  /OMM/
017112  101     116     104             .ASCII  /AND/
017115  040     124     111             .ASCII  / TI/
017120  115     117     125             .ASCII  /MOU/
017123  124     040     117             .ASCII  /T O/
017126  122     040     122             .ASCII  /R R/
017131  105     124     122             .ASCII  /ETR/
017134  131     040     114             .ASCII  /Y L/
017137  111     115     111             .ASCII  /IMI/
017142  124     040     105             .ASCII  /T E/
017145  130     103     105             .ASCII  /XCE/
017150  105     104     105             .ASCII  /EDE/
017153  104     000     000             .ASCII  /D/<00><00>
017156  045     101     123     P.AIT:  .ASCII  /≠AS/
017161  105     122     111             .ASCII  /ERI/
017164  101     114     111             .ASCII  /ALI/
017167  132     105     122             .ASCII  /ZER/
017172  057     104     105             .ASCII  <57>/DE/
017175  123     105     122             .ASCII  /SER/
017200  111     101     114             .ASCII  /IAL/
017203  111     132     105             .ASCII  /IZE/
017206  122     040     117             .ASCII  /R O/
017211  126     105     122             .ASCII  /VER/
017214  122     125     116             .ASCII  /RUN/
017217  040     117     122             .ASCII  / OR/
017222  040     125     116             .ASCII  / UN/
017225  104     105     122             .ASCII  /DER/
017230  122     125     116             .ASCII  /RUN/
017233  000                             .ASCII  <00>
017234  045     101     042     P.AIU:  .ASCII  /≠A"/
017237  105     122     122             .ASCII  /ERR/
017242  117     122     040             .ASCII  /OR /
017245  104     105     124             .ASCII  /DET/
017250  105     103     124             .ASCII  /ECT/
017253  111     117     116             .ASCII  /ION/
017256  040     103     117             .ASCII  / CO/
017261  104     105     042             .ASCII  /DE"/
017264  040     105     122             .ASCII  / ER/
017267  122     117     122             .ASCII  /ROR/
017272  000     000                     .ASCII  <00><00>
017274  045     101     111     P.AIV:  .ASCII  /≠AI/
017277  116     103     117             .ASCII  /NCO/
017302  116     123     111             .ASCII  /NSI/
017305  123     124     105             .ASCII  /STE/
017310  116     124     040             .ASCII  /NT /
017313  111     116     124             .ASCII  /INT/
017316  105     122     116             .ASCII  /ERN/
017321  101     114     040             .ASCII  /AL /
017324  104     101     124             .ASCII  /DAT/
017327  101     040     123             .ASCII  /A S/
017332  124     122     125             .ASCII  /TRU/
```

```
017335  103  124  125                .ASCII  /CTU/
017340  122  105  000                .ASCII  /RE/<00>
017343  000                         .ASCII  <00>
017344  045  101  104       P.AIW:   .ASCII  /%AD/
017347  122  111  126                .ASCII  /RIV/
017352  105  040  103                .ASCII  /E C/
017355  117  115  115                .ASCII  /OMM/
017360  101  116  104                .ASCII  /AND/
017363  040  124  111                .ASCII  / TI/
017366  115  105  117                .ASCII  /MEO/
017371  125  124  040                .ASCII  /UT /
017374  050  116  157                .ASCII  /(No/
017377  040  162  145                .ASCII  / re/
017402  163  160  157                .ASCII  /spo/
017405  156  163  145                .ASCII  /nse/
017410  040  157  162                .ASCII  / or/
017413  040  163  145                .ASCII  / se/
017416  145  153  040                .ASCII  /ek /
017421  151  156  143                .ASCII  /inc/
017424  157  155  160                .ASCII  /omp/
017427  154  145  164                .ASCII  /let/
017432  145  051  000                .ASCII  /e)/<00>
017435  000                         .ASCII  <00>
017436  045  101  103       P.AIX:   .ASCII  /%AC/
017441  117  116  124                .ASCII  /ONT/
017444  122  117  114                .ASCII  /ROL/
017447  114  105  122                .ASCII  /LER/
017452  040  104  105                .ASCII  / DE/
017455  124  105  103                .ASCII  /TEC/
017460  124  105  104                .ASCII  /TED/
017463  040  124  122                .ASCII  / TR/
017466  101  116  123                .ASCII  /ANS/
017471  115  111  123                .ASCII  /MIS/
017474  123  111  117                .ASCII  /SIO/
017477  116  040  117                .ASCII  /N O/
017502  122  040  120                .ASCII  /R P/
017505  122  117  124                .ASCII  /ROT/
017510  117  103  117                .ASCII  /OCO/
017513  114  040  105                .ASCII  /L E/
017516  122  122  117                .ASCII  /RRO/
017521  122  000  000                .ASCII  /R/<00><00>
017524  045  101  120       P.AIY:   .ASCII  /%AP/
017527  117  123  111                .ASCII  /OSI/
017532  124  111  117                .ASCII  /TIO/
017535  116  040  105                .ASCII  /N E/
017540  122  122  117                .ASCII  /RRO/
017543  122  040  050                .ASCII  /R (/
017546  115  151  163                .ASCII  /Mis/
017551  055  163  145                .ASCII  /-se/
017554  145  153  051                .ASCII  /ek)/
017557  000                         .ASCII  <00>
017560  045  101  114       P.AIZ:   .ASCII  /%AL/
017563  117  123  124                .ASCII  /OST/
```

```
017566  040     122     105             .ASCII  / RE/
017571  101     104     057             .ASCII  /AD/<57>
017574  127     122     111             .ASCII  /WRI/
017577  124     105     040             .ASCII  /TE /
017602  122     105     101             .ASCII  /REA/
017605  104     131     040             .ASCII  /DY /
017610  104     125     122             .ASCII  /DUR/
017613  111     116     107             .ASCII  /ING/
017616  057     102     105             .ASCII  <57>/BE/
017621  124     127     105             .ASCII  /TWE/
017624  105     116     040             .ASCII  /EN /
017627  124     122     101             .ASCII  /TRA/
017632  116     123     106             .ASCII  /NSF/
017635  105     122     123             .ASCII  /ERS/
017640  000     000                     .ASCII  <00><00>
017642  045     101     104     P.AJA:  .ASCII  /%AD/
017645  122     111     126             .ASCII  /RIV/
017650  105     040     103             .ASCII  /E C/
017653  114     117     103             .ASCII  /LOC/
017656  113     040     104             .ASCII  /K D/
017661  122     117     120             .ASCII  /ROP/
017664  117     125     124             .ASCII  /OUT/
017667  000                             .ASCII  <00>
017670  045     101     114     P.AJB:  .ASCII  /%AL/
017673  117     123     124             .ASCII  /OST/
017676  040     122     105             .ASCII  / RE/
017701  103     105     111             .ASCII  /CEI/
017704  126     105     122             .ASCII  /VER/
017707  040     122     105             .ASCII  / RE/
017712  101     104     131             .ASCII  /ADY/
017715  040     102     105             .ASCII  / BE/
017720  124     127     105             .ASCII  /TWE/
017723  105     116     040             .ASCII  /EN /
017726  123     105     103             .ASCII  /SEC/
017731  124     117     122             .ASCII  /TOR/
017734  123     000                     .ASCII  /S/<00>
017736  045     101     104     P.AJC:  .ASCII  /%AD/
017741  122     111     126             .ASCII  /RIV/
017744  105     040     104             .ASCII  /E D/
017747  105     124     105             .ASCII  /ETE/
017752  103     124     105             .ASCII  /CTE/
017755  104     040     105             .ASCII  /D E/
017760  122     122     117             .ASCII  /RRO/
017763  122     000     000             .ASCII  /R/<00><00>
017766  045     101     103     P.AJD:  .ASCII  /%AC/
017771  117     116     124             .ASCII  /ONT/
017774  122     117     114             .ASCII  /ROL/
017777  114     105     122             .ASCII  /LER/
020002  040     104     105             .ASCII  / DE/
020005  124     105     103             .ASCII  /TEC/
020010  124     105     104             .ASCII  /TED/
020013  040     120     125             .ASCII  / PU/
020016  114     123     105             .ASCII  /LSE/
```

```
020021  040  117  122            .ASCII  / OR/
020024  040  123  124            .ASCII  / ST/
020027  101  124  105            .ASCII  /ATE/
020032  040  120  101            .ASCII  / PA/
020035  122  111  124            .ASCII  /RIT/
020040  131  040  105            .ASCII  /Y E/
020043  122  122  117            .ASCII  /RRO/
020046  122  000                 .ASCII  /R/<00>
020050  045  101  103  P.AJF:     .ASCII  /%AC/
020053  117  116  124            .ASCII  /ONT/
020056  122  117  114            .ASCII  /ROL/
020061  114  105  122            .ASCII  /LER/
020064  040  124  111            .ASCII  / TI/
020067  115  105  117            .ASCII  /MEO/
020072  125  124  000            .ASCII  /UT/<00>
020075  000                      .ASCII  <00>
020076  045  101  105  P.AJG:     .ASCII  /%AE/
020101  116  126  105            .ASCII  /NVE/
020104  114  117  120            .ASCII  /LOP/
020107  105  057  120            .ASCII  /E/<57>/P/
020112  101  103  113            .ASCII  /ACK/
020115  105  124  040            .ASCII  /ET /
020120  122  105  101            .ASCII  /REA/
020123  104  040  105            .ASCII  /D E/
020126  122  122  117            .ASCII  /RRO/
020131  122  040  050            .ASCII  /R (/
020134  120  141  162            .ASCII  /Par/
020137  151  164  171            .ASCII  /ity/
020142  040  157  162            .ASCII  / or/
020145  040  164  151            .ASCII  / ti/
020150  155  145  157            .ASCII  /meo/
020153  165  164  051            .ASCII  /ut)/
020156  000  000                 .ASCII  <00><00>
020160  045  101  105  P.AJH:     .ASCII  /%AE/
020163  116  126  105            .ASCII  /NVE/
020166  114  117  120            .ASCII  /LOP/
020171  105  057  120            .ASCII  /E/<57>/P/
020174  101  103  113            .ASCII  /ACK/
020177  105  124  040            .ASCII  /ET /
020202  127  122  111            .ASCII  /WRI/
020205  124  105  040            .ASCII  /TE /
020210  105  122  122            .ASCII  /ERR/
020213  117  122  040            .ASCII  /OR /
020216  050  120  141            .ASCII  /(Pa/
020221  162  151  164            .ASCII  /rit/
020224  171  040  157            .ASCII  /y o/
020227  162  040  164            .ASCII  /r t/
020232  151  155  145            .ASCII  /ime/
020235  157  165  164            .ASCII  /out/
020240  051  000                 .ASCII  /)/<00>
020242  045  101  103  P.AJI:     .ASCII  /%AC/
020245  117  116  124            .ASCII  /ONT/
020250  122  117  114            .ASCII  /ROL/
```

```
020253  114  105  122            .ASCII  /LER/
020256  040  122  117            .ASCII  / RO/
020261  115  040  101            .ASCII  /M A/
020264  116  104  040            .ASCII  /ND /
020267  122  101  115            .ASCII  /RAM/
020272  040  120  101            .ASCII  / PA/
020275  122  111  124            .ASCII  /RIT/
020300  131  040  105            .ASCII  /Y E/
020303  122  122  117            .ASCII  /RRO/
020306  122  000                 .ASCII  /R/<00>
020310  045  101  103  P.AJJ:    .ASCII  /%AC/
020313  117  116  124            .ASCII  /ONT/
020316  122  117  114            .ASCII  /ROL/
020321  114  105  122            .ASCII  /LER/
020324  040  122  101            .ASCII  / RA/
020327  115  040  120            .ASCII  /M P/
020332  101  122  111            .ASCII  /ARI/
020335  124  131  040            .ASCII  /TY /
020340  105  122  122            .ASCII  /ERR/
020343  117  122  000            .ASCII  /OR/<00>
020346  045  101  103  P.AJK:    .ASCII  /%AC/
020351  117  116  124            .ASCII  /ONT/
020354  122  117  114            .ASCII  /ROL/
020357  114  105  122            .ASCII  /LER/
020362  040  122  117            .ASCII  / RO/
020365  115  040  120            .ASCII  /M P/
020370  101  122  111            .ASCII  /ARI/
020373  124  131  040            .ASCII  /TY /
020376  105  122  122            .ASCII  /ERR/
020401  117  122  000            .ASCII  /OR/<00>
020404  045  101  122  P.AJL:    .ASCII  /%AR/
020407  111  116  107            .ASCII  /ING/
020412  040  122  105            .ASCII  / RE/
020415  101  104  040            .ASCII  /AD /
020420  105  122  122            .ASCII  /ERR/
020423  117  122  040            .ASCII  /OR /
020426  050  120  141            .ASCII  /(Pa/
020431  162  151  164            .ASCII  /rit/
020434  171  040  157            .ASCII  /y o/
020437  162  040  164            .ASCII  /r t/
020442  151  155  145            .ASCII  /ime/
020445  157  165  164            .ASCII  /out/
020450  051  000                 .ASCII  /)/<00>
020452  045  101  122  P.AJM:    .ASCII  /%AR/
020455  111  116  107            .ASCII  /ING/
020460  040  127  122            .ASCII  / WR/
020463  111  124  105            .ASCII  /ITE/
020466  040  105  122            .ASCII  / ER/
020471  122  117  122            .ASCII  /ROR/
020474  040  050  120            .ASCII  / (P/
020477  141  162  151            .ASCII  /ari/
020502  164  171  040            .ASCII  /ty /
020505  157  162  040            .ASCII  /or /
```

```
020510  164     151     155             .ASCII  /tim/
020513  145     157     165             .ASCII  /eou/
020516  164     051     000             .ASCII  /t)/<00>
020521  000                             .ASCII  <00>
020522  111     116     124     P.AJN:  .ASCII  /INT/
020525  105     122     122             .ASCII  /ERR/
020530  125     120     124             .ASCII  /UPT/
020533  040     115     101             .ASCII  / MA/
020536  123     124     105             .ASCII  /STE/
020541  122     040     106             .ASCII  /R F/
020544  101     111     114             .ASCII  /AIL/
020547  125     122     105             .ASCII  /URE/
020552  000     000                     .ASCII  <00><00>
020554  045     101     110     P.AJO:  .ASCII  /%AH/
020557  117     123     124             .ASCII  /OST/
020562  040     101     103             .ASCII  / AC/
020565  103     105     123             .ASCII  /CES/
020570  123     040     124             .ASCII  /S T/
020573  111     115     105             .ASCII  /IME/
020576  117     125     124             .ASCII  /OUT/
020601  040     050     110             .ASCII  / (H/
020604  151     147     150             .ASCII  /igh/
020607  145     162     040             .ASCII  /er /
020612  154     145     166             .ASCII  /lev/
020615  145     154     040             .ASCII  /el /
020620  160     162     157             .ASCII  /pro/
020623  164     157     143             .ASCII  /toc/
020626  157     154     040             .ASCII  /ol /
020631  144     145     160             .ASCII  /dep/
020634  145     156     144             .ASCII  /end/
020637  145     156     164             .ASCII  /ent/
020642  051     000                     .ASCII  /)/<00>
020644  045     101     103     P.AJP:  .ASCII  /%AC/
020647  122     105     104             .ASCII  /RED/
020652  111     124     040             .ASCII  /IT /
020655  114     111     115             .ASCII  /LIM/
020660  111     124     040             .ASCII  /IT /
020663  105     130     103             .ASCII  /EXC/
020666  105     105     104             .ASCII  /EED/
020671  105     104     000             .ASCII  /ED/<00>
020674  045     101     121     P.AJQ:  .ASCII  /%AQ/
020677  055     102     125             .ASCII  /-BU/
020702  123     040     115             .ASCII  /S M/
020705  101     123     124             .ASCII  /AST/
020710  105     122     040             .ASCII  /ER /
020713  105     122     122             .ASCII  /ERR/
020716  117     122     000             .ASCII  /OR/<00>
020721  000                             .ASCII  <00>
020722  045     101     103     P.AJR:  .ASCII  /%AC/
020725  117     116     124             .ASCII  /ONT/
020730  122     117     114             .ASCII  /ROL/
020733  114     105     122             .ASCII  /LER/
020736  040     106     101             .ASCII  / FA/
```

```
020741  124  101  114            .ASCII  /TAL/
020744  040  105  122            .ASCII  / ER/
020747  122  117  122            .ASCII  /ROR/
020752  000  000                 .ASCII  <00><00>
020754  045  101  111  P.AJS:     .ASCII  /%AI/
020757  116  123  124            .ASCII  /NST/
020762  122  125  103            .ASCII  /RUC/
020765  124  111  117            .ASCII  /TIO/
020770  116  040  114            .ASCII  /N L/
020773  117  117  120            .ASCII  /OOP/
020776  040  124  111            .ASCII  / TI/
021001  115  105  117            .ASCII  /MEO/
021004  125  124  000            .ASCII  /UT/<00>
021007  000                      .ASCII  <00>
021010  045  101  111  P.AJT:     .ASCII  /%AI/
021013  114  114  105            .ASCII  /LLE/
021016  107  101  114            .ASCII  /GAL/
021021  040  126  111            .ASCII  / VI/
021024  122  124  125            .ASCII  /RTU/
021027  101  114  040            .ASCII  /AL /
021032  103  111  122            .ASCII  /CIR/
021035  103  125  111            .ASCII  /CUI/
021040  124  040  111            .ASCII  /T I/
021043  104  000  000            .ASCII  /D/<00><00>
021046  045  101  111  P.AJU:     .ASCII  /%AI/
021051  116  124  105            .ASCII  /NTE/
021054  122  122  125            .ASCII  /RRU/
021057  120  124  040            .ASCII  /PT /
021062  126  105  103            .ASCII  /VEC/
021065  124  117  122            .ASCII  /TOR/
021070  040  111  114            .ASCII  / IL/
021073  114  105  107            .ASCII  /LEG/
021076  101  114  000            .ASCII  /AL/<00>
021101  000                      .ASCII  <00>
021102  045  101  115  P.AJV:     .ASCII  /%AM/
021105  101  111  116            .ASCII  /AIN/
021110  124  105  116            .ASCII  /TEN/
021113  101  116  103            .ASCII  /ANC/
021116  105  040  122            .ASCII  /E R/
021121  105  101  104            .ASCII  /EAD/
021124  057  127  122            .ASCII  <57>/WR/
021127  111  124  105            .ASCII  /ITE/
021132  040  111  116            .ASCII  / IN/
021135  126  101  114            .ASCII  /VAL/
021140  111  104  040            .ASCII  /ID /
021143  122  105  107            .ASCII  /REG/
021146  111  117  116            .ASCII  /ION/
021151  040  111  104            .ASCII  / ID/
021154  105  116  124            .ASCII  /ENT/
021157  111  106  111            .ASCII  /IFI/
021162  105  122  000            .ASCII  /ER/<00>
021165  000                      .ASCII  <00>
021166  045  101  115  P.AJW:     .ASCII  /%AM/
```

```
021171  101  111  116                .ASCII  /AIN/
021174  124  105  116                .ASCII  /TEN/
021177  101  116  103                .ASCII  /ANC/
021202  105  040  127                .ASCII  /E W/
021205  122  111  124                .ASCII  /RIT/
021210  105  040  114                .ASCII  /E L/
021213  117  101  104                .ASCII  /OAD/
021216  040  124  117                .ASCII  / TO/
021221  040  116  117                .ASCII  / NO/
021224  116  055  114                .ASCII  /N-L/
021227  117  101  104                .ASCII  /OAD/
021232  101  102  114                .ASCII  /ABL/
021235  105  040  103                .ASCII  /E C/
021240  117  116  124                .ASCII  /ONT/
021243  122  117  114                .ASCII  /ROL/
021246  114  105  122                .ASCII  /LER/
021251  000                         .ASCII  <00>
021252  045  101  103       P.AJX:   .ASCII  /%AC/
021255  117  116  124                .ASCII  /ONT/
021260  122  117  114                .ASCII  /ROL/
021263  114  105  122                .ASCII  /LER/
021266  040  122  101                .ASCII  / RA/
021271  115  040  105                .ASCII  /M E/
021274  122  122  117                .ASCII  /RRO/
021277  122  040  050                .ASCII  /R (/
021302  116  157  156                .ASCII  /Non/
021305  055  160  141                .ASCII  /-pa/
021310  162  151  164                .ASCII  /rit/
021313  171  051  000                .ASCII  /y)/<00>
021316  045  101  111       P.AJY:   .ASCII  /%AI/
021321  116  111  124                .ASCII  /NIT/
021324  040  123  105                .ASCII  / SE/
021327  121  125  105                .ASCII  /QUE/
021332  116  103  105                .ASCII  /NCE/
021335  040  105  122                .ASCII  / ER/
021340  122  117  122                .ASCII  /ROR/
021343  000                         .ASCII  <00>
021344  045  101  110       P.AJZ:   .ASCII  /%AH/
021347  111  107  110                .ASCII  /IGH/
021352  105  122  040                .ASCII  /ER /
021355  114  105  126                .ASCII  /LEV/
021360  105  114  040                .ASCII  /EL /
021363  120  122  117                .ASCII  /PRO/
021366  124  117  103                .ASCII  /TOC/
021371  117  114  040                .ASCII  /OL /
021374  111  116  103                .ASCII  /INC/
021377  117  115  120                .ASCII  /OMP/
021402  101  124  111                .ASCII  /ATI/
021405  102  111  114                .ASCII  /BIL/
021410  111  124  131                .ASCII  /ITY/
021413  040  105  122                .ASCII  / ER/
021416  122  117  122                .ASCII  /ROR/
021421  000                         .ASCII  <00>
```

```
021422      045     101     120         P.AKA:  .ASCII  /%AP/
021425      125     122     107                 .ASCII  /URG/
021430      105     057     120                 .ASCII  /E/<57>/P/
021433      117     114     114                 .ASCII  /OLL/
021436      040     110     101                 .ASCII  / HA/
021441      122     104     127                 .ASCII  /RDW/
021444      101     122     105                 .ASCII  /ARE/
021447      040     106     101                 .ASCII  / FA/
021452      111     114     125                 .ASCII  /ILU/
021455      122     105     000                 .ASCII  /RE/<00>
021460      045     101     115         P.AKB:  .ASCII  /%AM/
021463      101     120     120                 .ASCII  /APP/
021466      111     116     107                 .ASCII  /ING/
021471      040     122     105                 .ASCII  / RE/
021474      107     111     123                 .ASCII  /GIS/
021477      124     105     122                 .ASCII  /TER/
021502      040     122     105                 .ASCII  / RE/
021505      101     104     040                 .ASCII  /AD /
021510      106     101     111                 .ASCII  /FAI/
021513      114     125     122                 .ASCII  /LUR/
021516      105     040     050                 .ASCII  /E (/
021521      120     141     162                 .ASCII  /Par/
021524      151     164     171                 .ASCII  /ity/
021527      040     157     162                 .ASCII  / or/
021532      040     164     151                 .ASCII  / ti/
021535      155     145     157                 .ASCII  /meo/
021540      165     164     051                 .ASCII  /ut)/
021543      000                                 .ASCII  <00>
021544   020050'                        P.AJE:  .WORD   P.AJF
021546   020076'                                .WORD   P.AJG
021550   020160'                                .WORD   P.AJH
021552   020242'                                .WORD   P.AJI
021554   020310'                                .WORD   P.AJJ
021556   020346'                                .WORD   P.AJK
021560   020404'                                .WORD   P.AJL
021562   020452'                                .WORD   P.AJM
021564   020522'                                .WORD   P.AJN
021566   020554'                                .WORD   P.AJO
021570   020644'                                .WORD   P.AJP
021572   020674'                                .WORD   P.AJQ
021574   020722'                                .WORD   P.AJR
021576   020754'                                .WORD   P.AJS
021600   021010'                                .WORD   P.AJT
021602   021046'                                .WORD   P.AJU
021604   021102'                                .WORD   P.AJV
021606   021166'                                .WORD   P.AJW
021610   021252'                                .WORD   P.AJX
021612   021316'                                .WORD   P.AJY
021614   021344'                                .WORD   P.AJZ
021616   021422'                                .WORD   P.AKA
021620   021460'                                .WORD   P.AKB
021622      045     101     124         P.AKD:  .ASCII  /%AT/
021625      061     061     040                 .ASCII  /11 /
```

```
021630  103  120  125                .ASCII  /CPU/
021633  040  106  101                .ASCII  / FA/
021636  111  114  125                .ASCII  /ILU/
021641  122  105  000                .ASCII  /RE/<00>
021644  045  101  116        P.AKE:  .ASCII  /%AN/
021647  117  116  055                .ASCII  /ON-/
021652  120  101  122                .ASCII  /PAR/
021655  111  124  131                .ASCII  /ITY/
021660  040  122  101                .ASCII  / RA/
021663  115  040  105                .ASCII  /M E/
021666  122  122  117                .ASCII  /RRO/
021671  122  000  000                .ASCII  /R/<00><00>
021674  045  101  123        P.AKF:  .ASCII  /%AS/
021677  124  101  124                .ASCII  /TAT/
021702  105  040  115                .ASCII  /E M/
021705  101  103  110                .ASCII  /ACH/
021710  111  116  105                .ASCII  /INE/
021713  040  106  101                .ASCII  / FA/
021716  111  114  125                .ASCII  /ILU/
021721  122  105  040                .ASCII  /RE /
021724  055  040  124                .ASCII  /- T/
021727  061  061  040                .ASCII  /11 /
021732  101  104  104                .ASCII  /ADD/
021735  122  105  123                .ASCII  /RES/
021740  123  040  122                .ASCII  /S R/
021743  105  107  111                .ASCII  /EGI/
021746  123  124  105                .ASCII  /STE/
021751  122  000  000                .ASCII  /R/<00><00>
021754  045  101  123        P.AKG:  .ASCII  /%AS/
021757  124  101  124                .ASCII  /TAT/
021762  105  040  115                .ASCII  /E M/
021765  101  103  110                .ASCII  /ACH/
021770  111  116  105                .ASCII  /INE/
021773  040  106  101                .ASCII  / FA/
021776  111  114  125                .ASCII  /ILU/
022001  122  105  040                .ASCII  /RE /
022004  055  040  121                .ASCII  /- Q/
022007  055  102  125                .ASCII  /-BU/
022012  123  040  101                .ASCII  /S A/
022015  104  104  122                .ASCII  /DDR/
022020  105  123  123                .ASCII  /ESS/
022023  040  122  105                .ASCII  / RE/
022026  107  111  123                .ASCII  /GIS/
022031  124  105  122                .ASCII  /TER/
022034  000  000                     .ASCII  <00><00>
022036  045  101  123        P.AKH:  .ASCII  /%AS/
022041  124  101  124                .ASCII  /TAT/
022044  105  040  115                .ASCII  /E M/
022047  101  103  110                .ASCII  /ACH/
022052  111  116  105                .ASCII  /INE/
022055  040  106  101                .ASCII  / FA/
022060  111  114  125                .ASCII  /ILU/
022063  122  105  040                .ASCII  /RE /
```

```
022066  055     040     103             .ASCII  /- C/
022071  122     103     040             .ASCII  /RC /
022074  122     105     107             .ASCII  /REG/
022077  111     123     124             .ASCII  /IST/
022102  105     122     000             .ASCII  /ER/<00>
022105  000                             .ASCII  <00>
022106  045     101     123     P.AKI:  .ASCII  /%AS/
022111  124     101     124             .ASCII  /TAT/
022114  105     040     115             .ASCII  /E M/
022117  101     103     110             .ASCII  /ACH/
022122  111     116     105             .ASCII  /INE/
022125  040     106     101             .ASCII  / FA/
022130  111     114     125             .ASCII  /ILU/
022133  122     105     040             .ASCII  /RE /
022136  055     040     123             .ASCII  /- S/
022141  105     122     111             .ASCII  /ERI/
022144  101     114     111             .ASCII  /ALI/
022147  132     105     122             .ASCII  /ZER/
022152  057     104     105             .ASCII  <57>/DE/
022155  123     105     122             .ASCII  /SER/
022160  111     101     114             .ASCII  /IAL/
022163  111     132     105             .ASCII  /IZE/
022166  122     040     122             .ASCII  /R R/
022171  105     107     111             .ASCII  /EGI/
022174  123     124     105             .ASCII  /STE/
022177  122     000     000             .ASCII  /R/<00><00>
022202  045     101     123     P.AKJ:  .ASCII  /%AS/
022205  124     101     124             .ASCII  /TAT/
022210  105     040     115             .ASCII  /E M/
022213  101     103     110             .ASCII  /ACH/
022216  111     116     105             .ASCII  /INE/
022221  040     106     101             .ASCII  / FA/
022224  111     114     125             .ASCII  /ILU/
022227  122     105     040             .ASCII  /RE /
022232  055     040     127             .ASCII  /- W/
022235  122     117     116             .ASCII  /RON/
022240  107     040     110             .ASCII  /G H/
022243  101     122     104             .ASCII  /ARD/
022246  127     101     122             .ASCII  /WAR/
022251  105     040     126             .ASCII  /E V/
022254  105     122     123             .ASCII  /ERS/
022257  111     117     116             .ASCII  /ION/
022262  000     000                     .ASCII  <00><00>
022264  021622'                 P.AKC:  .WORD   P.AKD
022266  021644'                         .WORD   P.AKE
022270  021674'                         .WORD   P.AKF
022272  021754'                         .WORD   P.AKG
022274  022036'                         .WORD   P.AKH
022276  022106'                         .WORD   P.AKI
022300  022202'                         .WORD   P.AKJ
022302  045     116     045     P.AKK:  .ASCII  /%N%/
022305  101     132     122             .ASCII  /AZR/
022310  121     101     040             .ASCII  /QA /
```

```
022313  104  105  126            .ASCII  /DEV/
022316  040  106  124            .ASCII  / FT/
022321  114  040  040            .ASCII  /L  /
022324  045  132  065            .ASCII  /%Z5/
022327  045  101  040            .ASCII  /%A /
022332  117  116  040            .ASCII  /ON /
022335  125  116  111            .ASCII  /UNI/
022340  124  040  045            .ASCII  /T %/
022343  132  062  045            .ASCII  /Z2%/
022346  101  040  124            .ASCII  /A T/
022351  123  124  040            .ASCII  /ST /
022354  060  060  061            .ASCII  /001/
022357  040  123  125            .ASCII  / SU/
022362  102  040  060            .ASCII  /B 0/
022365  060  060  040            .ASCII  /00 /
022370  120  103  072            .ASCII  /PC:/
022373  040  045  117            .ASCII  / %O/
022376  066  000                 .ASCII  /6/<00>
022400  045  116  045  P.AKL:     .ASCII  /%N%/
022403  101  132  122            .ASCII  /AZR/
022406  121  101  040            .ASCII  /QA /
022411  110  122  104            .ASCII  /HRD/
022414  040  105  122            .ASCII  / ER/
022417  122  040  040            .ASCII  /R  /
022422  045  132  065            .ASCII  /%Z5/
022425  045  101  040            .ASCII  /%A /
022430  117  116  040            .ASCII  /ON /
022433  125  116  111            .ASCII  /UNI/
022436  124  040  045            .ASCII  /T %/
022441  132  062  045            .ASCII  /Z2%/
022444  101  040  124            .ASCII  /A T/
022447  123  124  040            .ASCII  /ST /
022452  060  060  061            .ASCII  /001/
022455  040  123  125            .ASCII  / SU/
022460  102  040  060            .ASCII  /B 0/
022463  060  060  040            .ASCII  /00 /
022466  120  103  072            .ASCII  /PC:/
022471  040  045  117            .ASCII  / %O/
022474  066  000                 .ASCII  /6/<00>
022476  045  116  045  P.AKM:     .ASCII  /%N%/
022501  101  132  122            .ASCII  /AZR/
022504  121  101  040            .ASCII  /QA /
022507  123  106  124            .ASCII  /SFT/
022512  040  105  122            .ASCII  / ER/
022515  122  040  040            .ASCII  /R  /
022520  045  132  065            .ASCII  /%Z5/
022523  045  101  040            .ASCII  /%A /
022526  117  116  040            .ASCII  /ON /
022531  125  116  111            .ASCII  /UNI/
022534  124  040  045            .ASCII  /T %/
022537  132  062  045            .ASCII  /Z2%/
022542  101  040  124            .ASCII  /A T/
022545  123  124  040            .ASCII  /ST /
```

```
022550      060     060     061                 .ASCII  /001/
022553      040     123     125                 .ASCII  / SU/
022556      102     040     060                 .ASCII  /B 0/
022561      060     060     040                 .ASCII  /00 /
022564      120     103     072                 .ASCII  /PC:/
022567      040     045     117                 .ASCII  / %0/
022572      066     045     116                 .ASCII  /6%N/
022575      000                                 .ASCII  <00>
022576      045     116     045     P.AKN:      .ASCII  /%N%/
022601      101     111     057                 .ASCII  /AI/<57>
022604      117     040     122                 .ASCII  /O R/
022607      105     121     125                 .ASCII  /EQU/
022612      105     123     124                 .ASCII  /EST/
022615      040     106     101                 .ASCII  / FA/
022620      111     114     105                 .ASCII  /ILE/
022623      104     045     116                 .ASCII  /D%N/
022626      000     000                         .ASCII  <00><00>
022630      045     123     064     P.AKO:      .ASCII  /%S4/
022633      000                                 .ASCII  <00>
022634      045     116     000     P.AKP:      .ASCII  /%N/<00>
022637      000                                 .ASCII  <00>
022640      045     101     040     P.AKQ:      .ASCII  /%A /
022643      055     040     000                 .ASCII  /- /<00>
022646      045     101     052     P.AKR:      .ASCII  /%A*/
022651      040     000     000                 .ASCII  / /<00><00>
022654  000000C                     L$HWLEN::
                                                .WORD   <<L$NDHW-L$HWLEN>/2>
022656  172150                      HWPT.IP.ADDR::
                                                .WORD   -5630
022660  000154                      HWPT.VECTOR::
                                                .WORD   154
022662  000004                      HWPT.BR.LEVEL::
                                                .WORD   4
022664  000340                      HWPT.DISK::
                                                .WORD   340
022666  000000                      HWPTS0.LBN::
                                                .WORD   0
022670  000000                      HWPTS1.LBN::
                                                .WORD   0
022672  177777                      HWPTE0.LBN::
                                                .WORD   -1
022674  000000                      HWPTE1.LBN::
                                                .WORD   0
022676  020040                      NAME.LO::
                                                .WORD   20040
022700  020040                      NAME.HI::
                                                .WORD   20040
022702                              L$NDHW::.BLKW   1
022704  000000C                     L$SWLEN::
                                                .WORD   <<L$NDSW-L$SWLEN>/2>
022706  000040                      SWP.ERROR::
                                                .WORD   40
022710  000000                      SWP.XFER::
```

```
                                .WORD   0
022712  054046                  SWP.FLAGS::
                                .WORD   54046
022714  000000                  SWP.DPAT::
                                .WORD   0
022716  000143                  SWP.RAT::
                                .WORD   143
022720  000000                  SWP.TIME::
                                .WORD   0
022722  000013                  DUPROUND::
                                .WORD   13
022724  000020                  SWP.UCNT::
                                .WORD   20
022726                          SWP.UDPAT::
                                .BLKW   20
022766                          L$NDSW::.BLKW   1
022770  000000                  L$PROT::.WORD   0
022772  177777                          .WORD   -1
022774  000006                          .WORD   6


000000                          .PSECT  $FFF$,  D  ,  GBL
000000                  CST::   .BLKW   53
000126                  CST.ADDR::
                                .BLKW   1
000130                  DCT::   .BLKW   11
000152                  DCT.ADDR::
                                .BLKW   1
000154                  RDRX.ADDR::
                                .BLKW   1
000156                  IRDRX.ADDR::
                                .BLKW   1
000160                  BST::   .BLKW   10
000200                  TALLY:: .BLKW   154
000530                  T.ADDR::.BLKW   1
000532                  DUPPKT::.BLKW   401
001534                  TRK.SGN::
001534      001                 .BYTE   1
001535      001                 .BYTE   1
001536      001                 .BYTE   1
001537      001                 .BYTE   1
001540  000020                  RDM.CNT::
                                .WORD   20
001542                          RANDOM::.BLKW   20
001602                          C.ERR.TBL::
                                .BLKW   1
001604                          MSCP.PKT::
                                .BLKW   644
003314                          IPKT.ADDR::
                                .BLKW   1
003316                          PKT.USE::
                                .BLKW   6
```

```
003332                  RETPKT::.BLKW   260
004072                  RP.USE::.BLKW   4
004102                  RP.INDX::
                                .BLKW   1
004104                  RP.ADDR::
                                .BLKW   1
004106                  ELOG.PKT::
                                .BLKW   655
005640                  BUFF.ADDR::
                                .BLKW   10
005660                  BUFF.OWN::
                                .BLKW   4
005670                  IODQ::  .BLKW   4
005700                  IODQ.IN::
                                .BLKW   1
005702                  IODQ.OUT::
                                .BLKW   1
005704                  ENTRY.REASON::
                                .BLKB   1
005705                  EOP.FLAG::
                                .BLKB   1
005706                  DUP.FLAGS::
                                .BLKW   1
005710                  CCTLR:: .BLKW   1
005712                  CDISK:: .BLKW   1
005714                  CUOFF:: .BLKW   1
005716                  CTLR.CNT::
                                .BLKW   1
005720                  DUR::   .BLKW   2
005724                  QIO::   .BLKB   1
                                .EVEN
005726                  FREE.MEM.ADDR::
                                .BLKW   1
005730                  BYTS.PER.QIO::
                                .BLKW   1
005732                  ST.CODE::
                                .BLKW   1
005734                  SB.CODE::
                                .BLKW   1
005736                  STEP::  .BLKW   1
005740                  OF.RC:: .BLKW   1
005742                  SA.REG::.BLKW   1
005744                  CMD.TIME::
                                .BLKW   1
005746                  NEX::   .BLKW   1
005750                  CRN.LOW::
                                .BLKW   1
005752                  CRN.HIGH::
                                .BLKW   1
005754                  CREDIT.BAL::
                                .BLKW   1
005756                  NEXT.PKT.USE::
                                .BLKB   1
```

```
005757                          HOURS:: .BLKB   1
005760                          MINUTES::
                                        .BLKB   1
                                        .EVEN
005762                          CLK.TICKS::
                                        .BLKW   1
005764                          CLK.PRESENT::
                                        .BLKB   1
005765                          HOE.FLAG::
                                        .BLKB   1
005766                          S.PATTERN::
                                        .BLKW   1
005770                          S.DUPPKT::
                                        .BLKW   1
005772                          P.INDEX::
                                        .BLKW   1
005774                          FORCED.ERROR::
                                        .BLKB   1
                                        .EVEN
005776                          FER.LBN::
                                        .BLKW   1
006000                          FER.BC::.BLKW   1
006002                          INIT.OCCURED::
006002  000                             .BYTE   0
006003                          ADDR.VECT.OK::
006003  000                             .BYTE   0

                                .GLOBL  L$SOFT, T$PTHV, L$RPT, L$INIT
                                .GLOBL  L$CLEAN, L$LAST, L$HARD, L$DVTYP
                                .GLOBL  L$DESC, L$DU, L$AU, L$AUTO, T1


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=:          1000
000400                          BIT8=:          400
000200                          BIT7=:          200
000100                          BIT6=:          100
```

```
000040          BIT5==          40
000020          BIT4==          20
000010          BIT3==          10
000004          BIT2==          4
000002          BIT1==          2
000001          BIT0==          1
000040          EF.START==      40
000037          EF.RESTART==    37
000036          EF.CONTINUE==   36
000035          EF.NEW==        35
000034          EF.PWR==        34
000340          PRI07==         340
000300          PRI06==         300
000240          PRI05==         240
000200          PRI04==         200
000140          PRI03==         140
000100          PRI02==         100
000040          PRI01==         40
000000          PRI00==         0
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
000126'         L$ERRTBL==      ERRTYP
022706'         L$SW==          L$SWLEN+2
022656'         L$HW==          L$HWLEN+2
000011'         L$DEPO==        L$REV+1
000136'         HWQ1==          P.AAA
000152'         HWQ2==          P.AAB
000162'         HWQ3==          P.AAC
000226'         HWQ4==          P.AAD
000244'         HWQ5==          P.AAE
000314'         HWQ6A==         P.AAF
000366'         HWQ6B==         P.AAG
000442'         HWQ7A==         P.AAH
000512'         HWQ7B==         P.AAI
000562'         HWQ8==          P.AAJ
000634'         HWQ9==          P.AAK
000734'         HWQ10==         P.AAL
000764'         HWQ11==         P.AAM
001016'         SWQ1==          P.AAN
001040'         SWQ2==          P.AAO
001122'         SWQ4==          P.AAP
```

```
001144'  SWQ7==     P.AAQ
001216'  SWQ9==     P.AAR
001272'  SWQ10==    P.AAS
001336'  SWQ11==    P.AAT
001370'  SWQ12==    P.AAU
001466'  SWQ13==    P.AAV
001544'  SWQ14==    P.AAW
001616'  SWQ15==    P.AAX
001666'  SWQ17==    P.AAY
001764'  SWQ19==    P.AAZ
002054'  SWQ20==    P.ABA
002152'  SWQ21==    P.ABB
002254'  SWQ22==    P.ABC
002332'  SWQ23==    P.ABD
002424'  SWQ24==    P.ABE
002522'  SWQ25==    P.ABF
002622'  SWM1==     P.ABG
002712'  NULL==     P.ABH
002714'  DBM5==     P.ABI
002742'  DBM12==    P.ABJ
003016'  DBM15==    P.ABK
003046'  DBM18==    P.ABL
003120'  DBM19==    P.ABM
003204'  DBM20==    P.ABN
003260'  DBM21==    P.ABO
003342'  DBM22==    P.ABP
003406'  DBM23==    P.ABQ
003444'  DBM25==    P.ABR
003512'  DBM26==    P.ABS
003544'  DBM27==    P.ABT
003616'  DBM28A==   P.ABU
003656'  DBM28B==   P.ABV
003716'  DBM29==    P.ABW
003764'  DBM32==    P.ABX
004040'  DBM101==   P.ABY
004066'  DBM104==   P.ABZ
004130'  DBM105==   P.ACA
004166'  DBM107==   P.ACB
004224'  DBM108==   P.ACC
004314'  DBM109==   P.ACD
004374'  DBM111==   P.ACE
004474'  DBM112==   P.ACF
004576'  DBM120==   P.ACG
004670'  DBM121==   P.ACH
004760'  DU.MSG==   P.ACI
005500'  DU.RSN==   P.ACJ
005526'  MSG.01==   P.ACV
005560'  MSG.02==   P.ACW
005614'  MSG.03==   P.ACX
005646'  RPT1==     P.ACY
005732'  RPT2==     P.ACZ
005776'  RPT3==     P.ADA
006062'  RPT4==     P.ADB
```

```
006126'  RPT5==      P.ADC
006214'  RPT6==      P.ADD
006260'  RPT7==      P.ADE
006276'  RPT8==      P.ADF
006324'  RPT9==      P.ADG
006356'  RPT10==     P.ADH
006444'  RPT11==     P.ADI
006512'  RPT12==     P.ADJ
006560'  RPT13==     P.ADK
006660'  RPT14==     P.ADL
006756'  RPT15==     P.ADM
007056'  RPT16==     P.ADN
007140'  EGS.01==    P.ADO
007160'  EGS.02==    P.ADP
007252'  EGD.10==    P.ADQ
007312'  EGD.11==    P.ADR
007336'  EGD.12==    P.ADS
007364'  EGD.13==    P.ADT
007412'  EGD.14==    P.ADU
007442'  EGD.15==    P.ADV
007460'  EGD.16==    P.ADW
007510'  EGD.17==    P.ADX
007526'  EGD.18==    P.ADY
007546'  EGD.19==    P.ADZ
007606'  EGD.20==    P.AEA
007674'  EGD.21==    P.AEB
010006'  EGD.22==    P.AEC
010046'  EGD.23==    P.AED
010110'  EGD.24==    P.AEE
010154'  EGH.30==    P.AEF
010200'  EBS.01==    P.AEG
010242'  EBD.10==    P.AEH
010302'  EBD.12==    P.AEI
010350'  EBD.13==    P.AEJ
010402'  EBD.14==    P.AEK
010442'  EBD.18==    P.AEL
010476'  EBD.19==    P.AEM
010556'  EBD.24==    P.AEN
010632'  EH.0==      P.AEO
010670'  EH.1==      P.AEP
010726'  EH.2==      P.AEQ
010766'  EH.3==      P.AER
011024'  EH.4==      P.AES
011050'  EH.5==      P.AET
011100'  EH.6==      P.AEU
011130'  EH.7==      P.AEV
011160'  EH.8==      P.AEW
011214'  EH.9==      P.AEX
011244'  EH.10==     P.AEY
011274'  EH.12==     P.AEZ
011330'  EH.13==     P.AFA
011402'  ERR.00==    P.AFB
012072'  ERR.COD==   P.AFC
```

012126'	ELG.00==	P.AFR
012460'	ELG.FMT==	P.AFS
012472'	EX.SA==	P.AFY
012510'	EX.SC==	P.AFZ
012540'	EX.SBO==	P.AGA
012544'	EX.SB==	P.AGB
012566'	EX.CMD==	P.AGC
012606'	EX.RD==	P.AGD
012616'	EX.WRT==	P.AGE
012626'	EX.CMP==	P.AGF
012642'	EX.ONL==	P.AGG
012654'	EX.ACC==	P.AGH
012666'	EX.OP==	P.AGI
012672'	EX.BB==	P.AGJ
012762'	EX.BB1==	P.AGK
013056'	EX.BBU==	P.AGL
013146'	EX.LBN==	P.AGM
013204'	EX.PBN==	P.AGN
013242'	EX.LBR==	P.AGO
013310'	EX.LBW==	P.AGP
013356'	EX.RBN==	P.AGQ
013436'	EX.CBC==	P.AGR
013502'	EX.CBR==	P.AGS
013554'	EX.CBW==	P.AGT
013626'	EX.BC==	P.AGU
013702'	EX.BD==	P.AGV
013760'	EX.BDP==	P.AGW
014050'	EX.BDW==	P.AGX
014140'	EX.RP==	P.AGY
014226'	EX.WRD==	P.AGZ
014236'	EX.TIM==	P.AHA
014276'	XX13==	P.AHB
014320'	XX23==	P.AHC
014354'	XX32==	P.AHD
014402'	XX33==	P.AHE
014440'	XX34==	P.AHF
014510'	CER.01==	P.AHG
014554'	CER.02==	P.AHH
014630'	SC.SDI==	P.AHI
014654'	SC.CON==	P.AHJ
014676'	SC.DUP==	P.AHK
014726'	SC.ONL==	P.AHL
014750'	SC.SON==	P.AHM
014770'	SC.UNK==	P.AHN
015050'	SC.VOL==	P.AHO
015130'	SC.IOP==	P.AHP
015202'	SC.DIS==	P.AHQ
015274'	SC.FER==	P.AHR
015362'	SC.FE2==	P.AHS
015452'	SC.ISH==	P.AHT
015532'	SC.IS2==	P.AHU
015612'	SC.DST==	P.AHV
015666'	SC.DS2==	P.AHW

```
015740'                 SC.ECC==        P.AHX
016022'                 SC.ECD==        P.AHY
016054'                 SC.RCT==        P.AHZ
016074'                 SC.FUL==        P.AIA
016150'                 SC.576==        P.AIB
016224'                 SC.FCT==        P.AIC
016272'                 SC.EC1==        P.AID
016322'                 SC.EC2==        P.AIE
016352'                 SC.EC3==        P.AIF
016404'                 SC.EC4==        P.AIG
016434'                 SC.EC5==        P.AIH
016464'                 SC.EC6==        P.AII
016514'                 SC.EC7==        P.AIJ
016546'                 SC.EC8==        P.AIK
016600'                 SC.EC9==        P.AIL
016642'                 SC.SWP==        P.AIM
016702'                 SC.HWP==        P.AIN
016742'                 SC.ODA==        P.AIO
016772'                 SC.ODB==        P.AIP
017014'                 SC.NXM==        P.AIQ
017050'                 SC.PAR==        P.AIR
017104'                 SC.CTO==        P.AIS
017156'                 SC.SDS==        P.AIT
017234'                 SC.EDC==        P.AIU
017274'                 SC.IDS==        P.AIV
017344'                 SC.SRT==        P.AIW
017436'                 SC.SRI==        P.AIX
017524'                 SC.POE==        P.AIY
017560'                 SC.RDY==        P.AIZ
017642'                 SC.CLK==        P.AJA
017670'                 SC.RSP==        P.AJB
017736'                 SC.SUR==        P.AJC
017766'                 SC.PSP==        P.AJD
021544'                 CNTR.ERR==      P.AJE
022264'                 RDRX.ERR==      P.AJC
022302'                 DF.MSG==        P.AKK
022400'                 HRD.MSG==       P.AKL
022476'                 SFT.MSG==       P.AKM
022576'                 HRD.SUB==       P.AKN
022630'                 SPACE4==        P.AKO
022634'                 CRLF==          P.AKP
022640'                 DASH==          P.AKQ
022646'                 ASTERISK==      P.AKR
022656'                 DFPTBL==        L$HWLEN+2
022706'                 SFPTBL==        L$SWLEN+2


;                       PSECT SUMMARY
;
;       Psect Name      Words     Attributes
;         $CODE$         4863       RO  ,  1  ,  LCL,  REL,  CON
;         $FFF$          1538       RW  ,  D  ,  GBL,  REL,  CON
```

```
;                          Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;      File                              Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAE0.L16;18       405       181        44        21          00:00.1



;                                  COMMAND QUALIFIERS

;      BLISS/PDP11 ZRQAE0.BL1/LIST=ZRQAE0.LS1/OBJECT=ZRQAE0.OB1/SOURCE=PAGE:53
```

```
;    3273  0     module ZRQAM2 (
;    3274  0
;    3275  0     %title 'RD/RX EXERCISER'
;    3276  0                 ident = 'V01.8',
;    3277  0                 addressing_mode (absolute),
;    3278  0                 environment (noeis)
;    3279  0                 ) =
;    3280  0
;    3281  1     begin
;    3282  1
;    3283  1     %sbttl 'DECLARATIONS'
;    3284  1
;    3285  1     library 'ZRQAE0.L16';                          ! RDRX EXERCISER GLOBAL LIBRARY
;    3286  1
;    3287  1     require 'BLSMAC.REQ';                          ! DIAGNOSTIC SUPERVISOR LIBRARY
;    4778  1
;    4779  1     forward routine
;    4780  1         NEX_TRAP : L$ISR novalue,
;    4781  1         EMS_01 : novalue,
;    4782  1         EMS_TIM : novalue,
;    4783  1         EMS_DBN : NOVALUE,                                          !ZZZ
;    4784  1         EMS_BLK : NOVALUE,                                          !ZZZ
;    4785  1         SET_CPAR : novalue,
;    4786  1         SET_UPAR : novalue;
;    4787  1
;    4788  1     external
;    4789  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    4790  1                             ! RUN-TIME CONTROLLER STATUS TABLES
;    4791  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    4792  1                             ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    4793  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    4794  1                             ! DRIVER CONTROLLER TABLES
;    4795  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    4796  1                             ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    4797  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    4798  1                             ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    4799  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    4800  1                             ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    4801  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                     !ZZZ
;    4802  1                             !CONTAINS LBNS (HI + LO FIELDS) FOR SEQUENTIAL  !ZZZ
;    4803  1                             !I/O TRANSFER FOR EACH UNIT.                    !ZZZ
;    4804  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    4805  1                             ! STATISTICS TABLES
;    4806  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    4807  1                             ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    4808  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    4809  1                             !INFO FROM RECEIVE AND SEND CMDS                ZZZ
;    4810  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TRACK DIRECTION ZZZ
;    4811  1         RDM_CNT : WORD,                             !NO OF RANDOM NOS   \\KEEP      ZZZ
;    4812  1         RANDOM : VECTOR [RDM_LEN, WORD],            !RANDOM NO TABLE    //TOGETHER  ZZZ
;    4813  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    4814  1                             ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    4815  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
```

```
;  4816  1                            ! MSCP PACKET POOL
;  4817  1        IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  4818  1                            ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;  4819  1        PKT_USE : vector [PKT_CNT, byte, signed],
;  4820  1                            ! MSCP PACKET POOL ALLOCATION TABLE
;  4821  1        RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  4822  1                            ! RETURN PACKET POOL
;  4823  1        RP_USE : vector [RP_CNT, byte, signed],
;  4824  1                            ! RETURN PACKET POOL ALLOCATION TABLE
;  4825  1        RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;  4826  1        RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;  4827  1                            ! CURRENT RETURN PACKET ADDRESS
;  4828  1        ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;  4829  1                            ! ERROR-LOG PACKET SAVE AREA
;  4830  1        BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;  4831  1        BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;  4832  1        IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;  4833  1        IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;  4834  1        IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;  4835  1        ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;  4836  1        EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;  4837  1        DUP_FLAGS : WORD,                                    !DUP FLAGS       ZZZ
;  4838  1        CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;  4839  1        CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  4840  1        CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;  4841  1        CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;  4842  1        DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
;  4843  1        QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;  4844  1        FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;  4845  1        BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;  4846  1        ST_CODE : word,                                      ! CURRENT STATUS CODE
;  4847  1        SB_CODE : word,                                      ! CURRENT SUB-CODE
;  4848  1        STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;  4849  1        OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;  4850  1        SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;  4851  1        CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;  4852  1        NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;  4853  1        CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;  4854  1        CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;  4855  1        CREDIT_BAL : word,                                   ! CREDIT BALANCE
;  4856  1        NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;  4857  1        HOURS: byte,                                         ! TIME OF DAY (HOURS)
;  4858  1        MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;  4859  1        CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
;  4860  1        CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  4861  1        HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;  4862  1        FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
;  4863  1        FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
;  4864  1        FER_BC : word,                                       ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;  4865  1        INIT_OCCURED : byte,                                 ! EXERCISER INITIALIZATION COMPLETE
;  4866  1        ADDR_VECT_OK : byte,                                 ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;  4867  1        DBMS,
;  4868  1        P_INDEX : SIGNED WORD,              !CURRENT MESSAGE PACKET INDEX      ZZZ
```

```
;  4869  1          S_PATTERN : WORD.                !PATTERN FOR DUP WRITES          ZZZ
;  4870  1          S_DUPPKT : WORD.                 !DBN BYTE COUNTER                ZZZ
;  4871  1          DBM107.
;  4872  1          DU_MSG.
;  4873  1          DU_RSN : vector [11].
;  4874  1          RPT1.
;  4875  1          RPT2.
;  4876  1          RPT3.
;  4877  1          RPT4.
;  4878  1          RPT5.
;  4879  1          RPT6.
;  4880  1          RPT7.
;  4881  1          RPT8.
;  4882  1          RPT9.
;  4883  1          RPT10.
;  4884  1          RPT11.
;  4885  1          RPT12.
;  4886  1          RPT13.
;  4887  1          RPT14.
;  4888  1          RPT15.
;  4889  1          RPT16.
;  4890  1      !ZZZ    RPT17.
;  4891  1      !ZZZ    RPT18.
;  4892  1      !ZZZ    RPT19.
;  4893  1
;  4894  1          MSG_01.
;  4895  1          EGS_01.
;  4896  1          EBS_01.
;  4897  1          EBD_10.
;  4898  1          EBD_12.
;  4899  1          EBD_13.
;  4900  1          EBD_14.
;  4901  1          EBD_18.
;  4902  1          EBD_19.
;  4903  1          EBD_24.
;  4904  1          ERR_00.
;  4905  1          ERR_COD : vector [14],
;  4906  1          ELG_00.
;  4907  1          ELG_FMT : vector [5],
;  4908  1          EX_TIM,
;  4909  1          XX13,                                                    !ZZZ
;  4910  1          XX23,                                                    !ZZZ
;  4911  1          XX32,                                                    !ZZZ
;  4912  1          XX33,                                                    !ZZZ
;  4913  1          XX34,                                                    !ZZZ
;  4914  1          EX_SA.
;  4915  1          EX_SC.
;  4916  1          EX_SBO.
;  4917  1          EX_SB.
;  4918  1          EX_RP.
;  4919  1          EX_WRD.
;  4920  1          EX_CMD.
;  4921  1          EX_RD.
```

```
;    4922  1          EX_WRT.
;    4923  1          EX_CMP.
;    4924  1          EX_ONL.
;    4925  1          EX_ACC.
;    4926  1          EX_OP.
;    4927  1          EX_BB.
;    4928  1          EX_BB1.
;    4929  1          EX_BBU.
;    4930  1          EX_LBN.
;    4931  1          EX_PBN.
;    4932  1          EX_LBR.
;    4933  1          EX_LBW.
;    4934  1          EX_RBN.
;    4935  1          EX_CBC.
;    4936  1          EX_CBR.
;    4937  1          EX_CBW.
;    4938  1          EX_BC.
;    4939  1          EX_BD.
;    4940  1          EX_BDR.
;    4941  1          EX_BDW.
;    4942  1          SC_SDI.
;    4943  1          SC_CON.
;    4944  1          SC_DUP.
;    4945  1          SC_ONL.
;    4946  1          SC_SON.
;    4947  1          SC_UNK.
;    4948  1          SC_VOL.
;    4949  1          SC_IOP.
;    4950  1          SC_DIS.
;    4951  1          SC_FER.
;    4952  1          SC_FE2.
;    4953  1          SC_ISH.
;    4954  1          SC_IS2.
;    4955  1          SC_DST.
;    4956  1          SC_DS2.
;    4957  1          SC_ECC.
;    4958  1          SC_ECD.
;    4959  1          SC_RCT.
;    4960  1          SC_FUL.
;    4961  1          SC_576.
;    4962  1          SC_FCT.
;    4963  1          SC_SWP.
;    4964  1          SC_HWP.
;    4965  1          SC_EC1.
;    4966  1          SC_EC2.
;    4967  1          SC_EC3.
;    4968  1          SC_EC4.
;    4969  1          SC_EC5.
;    4970  1          SC_EC6.
;    4971  1          SC_EC7.
;    4972  1          SC_EC8.
;    4973  1          SC_EC9.
;    4974  1          SC_ODA.
```

```
;    4975  1          SC_ODB,
;    4976  1          SC_NXM,
;    4977  1          SC_PAR,
;    4978  1          SC_CTO,
;    4979  1          SC_SDS,
;    4980  1          SC_EDC,
;    4981  1          SC_IDS,
;    4982  1          SC_SRT,
;    4983  1          SC_SRI,
;    4984  1          SC_POE,
;    4985  1          SC_RDY,
;    4986  1          SC_CLK,
;    4987  1          SC_RSP,
;    4988  1          SC_SUR,
;    4989  1          SC_PSP,
;    4990  1          CER_01,
;    4991  1          CER_02,
;    4992  1          CNTR_ERR : vector [23],
;    4993  1          RDRX_ERR : vector [7],
;    4994  1          SPACE4,
;    4995  1          CRLF,
;    4996  1          DASH,
;    4997  1          ASTERISK,
;    4998  1          HWQ1,
;    4999  1          HWQ2,
;    5000  1          HWQ3,
;    5001  1          HWQ4,
;    5002  1          HWQ5,
;    5003  1          HWQ6A,
;    5004  1          HWQ6B,
;    5005  1          HWQ7A,
;    5006  1          HWQ7B,
;    5007  1          HWQ8,
;    5008  1          HWQ9,
;    5009  1          HWQ10,                                          !ZZZ
;    5010  1          HWQ11,                                          !ZZZ
;    5011  1          SWQ1,
;    5012  1          SWQ2,
;    5013  1          SWQ4,
;    5014  1          SWQ7,
;    5015  1          SWQ9,
;    5016  1          SWQ10,
;    5017  1          SWQ11,
;    5018  1          SWQ12,
;    5019  1          SWQ13,
;    5020  1          SWQ14,
;    5021  1          SWQ15,
;    5022  1          SWQ17,
;    5023  1          SWQ19,
;    5024  1          SWQ20,
;    5025  1          SWQ21,
;    5026  1          SWQ22,
;    5027  1          SWQ23,
```

```
;    5028  1          SWQ24,
;    5029  1          SWQ25,
;    5030  1          EH_0,                                          !ZZZ
;    5031  1          EH_1,                                          !ZZZ
;    5032  1          EH_2,                                          !ZZZ
;    5033  1          EH_3,                                          !ZZZ
;    5034  1          EH_4,                                          !ZZZ
;    5035  1          EH_5,                                          !ZZZ
;    5036  1          EH_6,                                          !ZZZ
;    5037  1          EH_7,                                          !ZZZ
;    5038  1          EH_8,                                          !ZZZ
;    5039  1          EH_9,                                          !ZZZ
;    5040  1          EH_10,                                         !ZZZ
;    5041  1          EH_12,                                         !ZZZ
;    5042  1          EH_13,                                         !ZZZ
;    5043  1          SWM1,
;    5044  1          NULL,
;    5045  1          SWP_FLAGS : word,
;    5046  1          L$HIMEM,
;    5047  1          L$LUN,
;    5048  1          L$UNIT;
;    5049  1      !   0_BRK;
;    5050  1
;    5051  1      own
;    5052  1          TBL_SUC : vector [17] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
;    5053  1                    NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON),
;    5054  1          TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
;    5055  1                    NULL, SC_DIS),
;    5056  1          TBL_MFE : vector [11] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC_576,
;    5057  1                    SC_FCT, SC_ECC, SC_RCT, SC_FUL, SC_EC1),
;    5058  1          TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
;    5059  1          TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
;    5060  1                    SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
;    5061  1          TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
;    5062  1          TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_EDC, SC_IDS),
;    5063  1          TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC_POE, SC_RDY, SC_CLK, SC_RSP,
;    5064  1                    SC_SUR, SC_PSP);
```

```
;    5065  1     %sbttl 'TYPE AND DESCRIPTION'
;    5066  1
;    5067  1     EQUALS;
;    5068  1
;    5069  1     DEVTYP (%asciz'RQDX1 or RUX50');                    ! NAME OF DEVICE SUPPORTED BY PROGRAM
;    5070  1     DESCRIPT (%asciz'RD/RX EXERCISER');     ! TEST DESCRIPTION
```

```
;    5071  1     %sbttl 'HARDWARE PARAMETER CODING SECTION'
;    5072  1
;    5073  1     !+
;    5074  1     ! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5075  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5076  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5077  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5078  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5079  1     ! WITH THE OPERATOR.
;    5080  1     !-
;    5081  1
;    5082  1     BGNHRD;
;    5083  1
;    5084  1     GPRMA (HWQ1, 0, 0, %o'160000', %o'177777', YES, 1);                    ! IP ADDRESS
;    5085  1     GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                            ! VECTOR
;    5086  1     GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                     ! BR LEVEL
;    5087  1     GPRMD (HWQ4, 6, 0, %o'17', %decimal'0', %decimal'15', YES, 1);         ! RDRX DRIVE NUMBER
;    5088  1     GPRML (HWQ10, 6, %o'000040', YES, 1);                                  !ALSO RUN DUP EXERCISER     ZZZ
;    5089  1     XFERF (NODU);                                                          !                           ZZZ
;    5090  1     GPRML (HWQ11, 6, %o'000100', YES, 1);                                  !WRITE DIAG AREA            ZZZ
;    5091  1     $L (NODU);                                                             !                           ZZZ
;    5092  1     GPRML (HWQ5, 6, %o'000200', YES, 1);                                   !TEST ENTIRE CUSTOMER AREA? ZZZ
;    5093  1     XFERT (TOQ8);                                                          !BR IF YES                  ZZZ
;    5094  1     GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);      ! STARTING LBN LO           ZZZ
;    5095  1     GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! STARTING LBN HI           ZZZ
;    5096  1     GPRMD (HWQ7A, 12, 0, %o'177777', GP$ATLO (8), %o'177777', YES, 1);     ! ENDING LBN LO             ZZZ
;    5097  1     GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! ENDING LBN HI             ZZZ
;    5098  1     $L (TOQ8);                                                             !                           ZZZ
;    5099  1     GPRML (HWQ8, 6, %o'100000', NO, 0);                                    ! WRITE ON CUST DATA AREA
;    5100  1     XFERF (HWDONE);                                                        ! NO - DONE
;    5101  1     GPRML (HWQ9, 6, %o'100000', NO, 1);                                    ! ** WARNING / CONFIRM
;    5102  1     $L (HWDONE);
;    5103  1
;    5104  1     ENDHRD;
```

```
;    5105  1      %sbttl 'SOFTWARE PARAMETER CODING SECTION'
;    5106  1
;    5107  1      !+
;    5108  1      ! THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5109  1      ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5110  1      ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5111  1      ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5112  1      ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5113  1      ! WITH THE OPERATOR.
;    5114  1      !-
;    5115  1
;    5116  1      BGNSFT;
;    5117  1
;    5118  1      !GPRML (SWQ16, 4, SWF_TRC, YES, 1);                          ! ENABLE DIAGNOSTIC TRACE
;    5119  1      GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);           ! START TIME
;    5120  1      GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);            ! ERROR LIMIT
;    5121  1      GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);               ! TRANSFER LIMIT
;    5122  1      GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);             ! PERCENT OF RD OPERATIONS
;    5123  1      GPRML (SWQ15, 4, SWF_CST, YES, 1);                           ! CLEAR STATISTICAL TABLES ?
;    5124  1      GPRML (SWQ20, 4, SWF_FER, YES, 1);                           ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
;    5125  1      GPRML (SWQ23, 4, SWF_BLK, YES, 1);                           ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
;    5126  1      GPRML (SWQ21, 4, SWF_HRD, YES, 1);                           ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
;    5127  1      GPRML (SWQ22, 4, SWF_SFT, YES, 1);                           ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
;    5128  1      GPRML (SWQ25, 4, SWF_TRY, YES, 1);                           ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
;    5129  1      GPRML (SWQ4, 4, SWF_RDM, YES, 1);                            ! RANDOM SEEK MODE ?
;    5130  1      XFERF (SW1);                                                 ! IF NO, DO NEXT QUESTION
;    5131  1      XFER (SW2);                                                  !
;    5132  1      $L (SW1);                                                    !
;    5133  1      GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                           ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
;    5134  1      $L (SW2);                                                    !
;    5135  1      GPRML (SWQ7, 4, SWF_CRC, YES, 1);                            ! READ-COMPARES AT CONTROLLER ?
;    5136  1      DISPLAY (SWM1);                                              ! REMAINING QUESTIONS ONLY APPLY ...
;    5137  1      GPRML (SWQ9, 4, SWF_CWC, YES, 1);                            ! WRITE-COMPARES AT CONTROLLER ?
;    5138  1      XFERF (SW3);                                                 ! IF NO, DO NEXT QUESTION
;    5139  1      XFER (SW4);                                                  !
;    5140  1      $L (SW3);                                                    !
;    5141  1      GPRML (SWQ10, 4, SWF_HWC, YES, 1);                           ! CHECK WRITES AT HOST BY READING ?
;    5142  1      $L (SW4);                                                    !
;    5143  1      GPRML (SWQ11, 4, SWF_UDP, YES, 1);                           ! USER-DEFINED DATA PATTERN ?
;    5144  1      XFERF (SW5);                                                 ! IF NO, DO NEXT QUESTION
;    5145  1      XFER (SW6);                                                  !
;    5146  1      $L (SW5);                                                    !
;    5147  1      GPRMD (SWQ12, 6, D, %o'177777', 0, DP_CNT, YES, 1);          ! SELECT PRE-DEFINED DATA PATTERN
;    5148  1      XFER (SW7);                                                  ! DONE
;    5149  1      $L (SW6);                                                    !
;    5150  1      GPRMD (SWQ13, 12, D, %o'177777', 1, MAX_UDP_CNT, YES, 1);    ! NO. OF WORDS IN USER DATA PATTERN
;    5151  1      GPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', NO, 12);     ! PATTERN VALUES
;    5152  1      $L (SW7);
;    5153  1
;    5154  1      ENDSFT;
```

```
;    5155  1
;    5156  1
;    5157  1      %sbttl 'REPORT CODING SECTION'
;    5158  1
;    5159  1
;    5160  1      !+
;    5161  1      ! THE REPORT CODING SECTION CONTAINS THE
;    5162  1      ! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;    5163  1      !-
;    5164  1
;    5165  1
;    5166  2      BGNRPT;
;    5167  2
;    5168  2      local
;    5169  2          CUR_PRIORITY : word;
;    5170  2
;    5171  2      GETPRI (CUR_PRIORITY);
;    5172  2      SETPRI (PRI04);
;    5173  2
;    5174  2      PRINTS (RPT1);
;    5175  2      PRINTS (RPT2);
;    5176  2      PRINTS (RPT3);
;    5177  2      PRINTS (RPT4);
;    5178  2      PRINTS (RPT5);
;    5179  2      PRINTS (RPT6);
;    5180  2
;    5181  2      incr CTLR from 0 to MAX_CTLR - 1 do
;    5182  2
;    5183  3          begin
;    5184  3          SET_CPAR (.CTLR);
;    5185  3
;    5186  3          incr DISK from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    5187  3
;    5188  4              begin
;    5189  4              SET_UPAR (.DISK);
;    5190  4
;    5191  4
;    5192  4              if .CST_ADDR [.DISK + OF_DATA, D_PRES] eql PRESENT
;    5193  4              then
;    5194  4
;    5195  5                  begin
; P  5196  5                  PRINTS (RPT7,
;    5197  5                      .L$LUN, .CST_ADDR [.DISK + OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK + OF_NAME_0, D_NAME_0]);
; P  5198  5                  PRINTS (RPT8,
; P  5199  5                      .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
;    5200  5                      .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
; P  5201  5                  PRINTS (RPT8,
; P  5202  5                      .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
;    5203  5                      .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT_LO]);
; P  5204  5                  PRINTS (RPT9,
; P  5205  5                      .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
;    5206  5                      .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
;    5207  4                  end;
```

```
;    5208  3              end;
;    5209  3
;    5210  3
;    5211  3          if .CST [.CTLR, STATE] eql PRESENT
;    5212  3          then
;    5213  3
;    5214  4              begin
;    5215  4              PRINTS (RPT10);
;    5216  4              PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
;    5217  3              end;
;    5218  3
;    5219  3
;    5220  2          end;
;    5221  2
;    5222  2      SETPRI (.CUR_PRIORITY);
;    5223  2
;    5224  2
;    5225  3      begin                                                          ! PRINTS DUP DATA
;    5226  3      prints(crlf);
;    5227  3      PRINTS(RPT13);
;    5228  3      PRINTS(RPT14);
;    5229  3      PRINTS(RPT15);
;    5230  3      INCR CTLR FROM 0 TO MAX_CTLR-1 DO
;    5231  4              BEGIN
;    5232  4              SET_CPAR(.CTLR);
;    5233  4              INCR DISK FROM (0+OF_UN) TO (3*UNIT_SIZE+OF_UN) BY UNIT_SIZE DO
;    5234  5                      BEGIN
;    5235  5                      SET_UPAR(.DISK);
;    5236  5                      IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_51 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;    5237  5                      THEN
; P  5238  5                  PRINTS (RPT16,
; P  5239  5                      .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;    5240  5                      .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;    5241  5
;    5242  5      !ZZZ            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_52 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;    5243  5      !ZZZ            THEN
;    5244  5      !ZZZ        PRINTS (RPT18,
;    5245  5      !ZZZ            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;    5246  5      !ZZZ            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;    5247  4                      END;
;    5248  3                END;
;    5249  3                   PRINTS (CRLF);
;    5250  2      END;
;    5251  2
;    5252  1      ENDRPT;


                                                .TITLE  ZRQAM2 RD/RX EXERCISER
                                                .IDENT  /V01.8/
                                                .ENABL  AMA


000000                                          .PSECT  $CODE$,  RO
```

```
000000      122     121     104         L$DVTYP::
                                                .ASCII  /RQD/
000003      130     061     040                 .ASCII  /X1 /
000006      157     162     040                 .ASCII  /or /
000011      122     125     130                 .ASCII  /RUX/
000014      065     060     000                 .ASCII  /50/<00>
000017      000                                 .ASCII  <00>
000020      122     104     057         L$DESC::.ASCII  /RD/<57>
000023      122     130     040                 .ASCII  /RX /
000026      105     130     105                 .ASCII  /EXE/
000031      122     103     111                 .ASCII  /RCI/
000034      123     105     122                 .ASCII  /SER/
000037      000                                 .ASCII  <00>
000040                                          .BLKB   2
000042  000000C                         L$HRDLN::
                                                .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000044  000031                          GP$1::  .WORD   31
000046  000000G                                 .WORD   HWQ1
000050  160000                                  .WORD   -20000
000052  177777                                  .WORD   -1
000054  001031                          GP$2::  .WORD   1031
000056  000000G                                 .WORD   HWQ2
000060  000004                                  .WORD   4
000062  000774                                  .WORD   774
000064  002032                          GP$3::  .WORD   2032
000066  000000G                                 .WORD   HWQ3
000070  000377                                  .WORD   377
000072  000000                                  .WORD   0
000074  000007                                  .WORD   7
000076  003052                          GP$4::  .WORD   3052
000100  000000G                                 .WORD   HWQ4
000102  000017                                  .WORD   17
000104  000000                                  .WORD   0
000106  000017                                  .WORD   17
000110  003130                          GP$5::  .WORD   3130
000112  000000G                                 .WORD   HWQ10
000114  000040                                  .WORD   40
000116  000000C                         $NODU:  .WORD   <<<<$LNODU-$NODU>*400>+4>+40>
000120  003130                          GP$6::  .WORD   3130
000122  000000G                                 .WORD   HWQ11
000124  000100                                  .WORD   100
000126  001004                          $LNODU: .WORD   1004
000130  003130                          GP$7::  .WORD   3130
000132  000000G                                 .WORD   HWQ5
000134  000200                                  .WORD   200
000136  000000C                         $TOQ8:  .WORD   <<<<$LTOQ8-$TOQ8>*400>+4>+20>
000140  004032                          GP$8::  .WORD   4032
000142  000000G                                 .WORD   HWQ6A
000144  177777                                  .WORD   -1
000146  000000                                  .WORD   0
000150  177777                                  .WORD   -1
000152  005032                          GP$9::  .WORD   5032
000154  000000G                                 .WORD   HWQ6B
```

```
000156  177777                          .WORD   -1
000160  000000                          .WORD   0
000162  177777                          .WORD   -1
000164  006432                  GP$10:: .WORD   6432
000166  000000G                         .WORD   HWQ7A
000170  177777                          .WORD   -1
000172  000004                          .WORD   4
000174  177777                          .WORD   -1
000176  000001                          .WORD   1
000200  007032                  GP$11:: .WORD   7032
000202  000000G                         .WORD   HWQ7B
000204  177777                          .WORD   -1
000206  000000                          .WORD   0
000210  177777                          .WORD   -1
000212  001004                  $LTOQ8: .WORD   1004
000214  003120                  GP$12:: .WORD   3120
000216  000000G                         .WORD   HWQ8
000220  100000                          .WORD   -100000
000222  000000C                 $HWDONE:.WORD   <<<<$LHWDONE-$HWDONE>*400>+4>+40>
000224  003120                  GP$13:: .WORD   3120
000226  000000G                         .WORD   HWQ9
000230  100000                          .WORD   -100000
000232  001004                  $LHWDONE:
                                        .WORD   1004
000234                          L$NDHRD::
                                        .BLKW   1
000236  000000C                 L$SFTLN::
                                        .WORD   <<<L$NDSFT-L$SFTLN>/2>-1>
000240  005052                  GP$14:: .WORD   5052
000242  000000G                         .WORD   SWQ24
000244  177777                          .WORD   -1
000246  000000                          .WORD   0
000250  004467                          .WORD   4467
000252  000052                  GP$15:: .WORD   52
000254  000000G                         .WORD   SWQ1
000256  177777                          .WORD   -1
000260  000000                          .WORD   0
000262  177777                          .WORD   -1
000264  001052                  GP$16:: .WORD   1052
000266  000000G                         .WORD   SWQ2
000270  177777                          .WORD   -1
000272  000000                          .WORD   0
000274  000143                          .WORD   143
000276  004052                  GP$17:: .WORD   4052
000300  000000G                         .WORD   SWQ17
000302  177777                          .WORD   -1
000304  000000                          .WORD   0
000306  000144                          .WORD   144
000310  002130                  GP$18:: .WORD   2130
000312  000000G                         .WORD   SWQ15
000314  000200                          .WORD   200
000316  002130                  GP$19:: .WORD   2130
000320  000000G                         .WORD   SWQ20
```

```
000322  004000                          .WORD   4000
000324  002130                  GP$20:: .WORD   2130
000326  000000G                         .WORD   SWQ23
000330  040000                          .WORD   40000
000332  002130                  GP$21:: .WORD   2130
000334  000000G                         .WORD   SWQ21
000336  010000                          .WORD   10000
000340  002130                  GP$22:: .WORD   2130
000342  000000G                         .WORD   SWQ22
000344  020000                          .WORD   20000
000346  002130                  GP$23:: .WORD   2130
000350  000000G                         .WORD   SWQ25
000352  100000                          .WORD   -100000
000354  002130                  GP$24:: .WORD   2130
000356  000000G                         .WORD   SWQ4
000360  000002                          .WORD   2
000362  000000C                 $SW1:   .WORD   <<<<$LSW1-$SW1>*400>+4>+40>
000364  000000C                 $SW2:   .WORD   <<<$LSW2-$SW2>*400>+4>
000366  001004                  $LSW1:  .WORD   1004
000370  002130                  GP$25:: .WORD   2130
000372  000000G                         .WORD   SWQ19
000374  001000                          .WORD   1000
000376  001004                  $LSW2:  .WORD   1004
000400  002130                  GP$26:: .WORD   2130
000402  000000G                         .WORD   SWQ7
000404  000004                          .WORD   4
000406  000003                  GP$DISP::
                                        .WORD   3
000410  000000G                         .WORD   SWM1
000412  002130                  GP$27:: .WORD   2130
000414  000000G                         .WORD   SWQ9
000416  000020                          .WORD   20
000420  000000C                 $SW3:   .WORD   <<<<$LSW3-$SW3>*400>+4>+40>
000422  000000C                 $SW4:   .WORD   <<<$LSW4-$SW4>*400>+4>
000424  001004                  $LSW3:  .WORD   1004
000426  002130                  GP$28:: .WORD   2130
000430  000000G                         .WORD   SWQ10
000432  000040                          .WORD   40
000434  001004                  $LSW4:  .WORD   1004
000436  002130                  GP$29:: .WORD   2130
000440  000000G                         .WORD   SWQ11
000442  000100                          .WORD   100
000444  000000C                 $SW5:   .WORD   <<<<$LSW5-$SW5>*400>+4>+40>
000446  000000C                 $SW6:   .WORD   <<<$LSW6-$SW6>*400>+4>
000450  001004                  $LSW5:  .WORD   1004
000452  003052                  GP$30:: .WORD   3052
000454  000000G                         .WORD   SWQ12
000456  177777                          .WORD   -1
000460  000000                          .WORD   0
000462  000025                          .WORD   25
000464  000000C                 $SW7:   .WORD   <<<$LSW7-$SW7>*400>+4>
000466  001004                  $LSW6:  .WORD   1004
000470  006052                  GP$31:: .WORD   6052
```

```
000472  000000G                         .WORD   SWQ13
000474  177777                          .WORD   -1
000476  000001                          .WORD   1
000500  000020                          .WORD   20
000502  007222                  GP$32:: .WORD   7222
000504  000000G                         .WORD   SWQ14
000506  177777                          .WORD   -1
000510  000000                          .WORD   0
000512  177777                          .WORD   -1
000514  000006                          .WORD   6
000516  001004                  $LSW7:  .WORD   1004
000520                          L$NDSFT::
                                .BLKW   1


000000                          .PSECT  $OWN$,  D
000000  000000G                 TBL.SUC:.WORD   NULL
000002  000000G                         .WORD   SC.SDI
000004  000000G                         .WORD   SC.CON
000006  000000G                         .WORD   NULL
000010  000000G                         .WORD   SC.DUP
000012  000000G                         .WORD   NULL
000014  000000G                         .WORD   NULL
000016  000000G                         .WORD   NULL
000020  000000G                         .WORD   SC.ONL
000022  000000G                         .WORD   NULL
000024  000000G                         .WORD   NULL
000026  000000G                         .WORD   NULL
000030  000000G                         .WORD   NULL
000032  000000G                         .WORD   NULL
000034  000000G                         .WORD   NULL
000036  000000G                         .WORD   NULL
000040  000000G                         .WORD   SC.SON
000042  000000G                 TBL.OFL:.WORD   SC.UNK
000044  000000G                         .WORD   SC.VOL
000046  000000G                         .WORD   SC.IOP
000050  000000G                         .WORD   NULL
000052  000000G                         .WORD   SC.DUP
000054  000000G                         .WORD   NULL
000056  000000G                         .WORD   NULL
000060  000000G                         .WORD   NULL
000062  000000G                         .WORD   SC.DIS
000064  000000G                 TBL.MFE:.WORD   SC.FER
000066  000000G                         .WORD   NULL
000070  000000G                         .WORD   SC.ISH
000072  000000G                         .WORD   SC.DST
000074  000000G                         .WORD   SC.EC9
000076  000000G                         .WORD   SC.576
000100  000000G                         .WORD   SC.FCT
000102  000000G                         .WORD   SC.ECC
000104  000000G                         .WORD   SC.RCT
000106  000000G                         .WORD   SC.FUL
```

```
000110  000000G                         .WORD   SC.EC1
000112  000000G                TBL.WPT: .WORD   NULL
000114  000000G                         .WORD   SC.SWP
000116  000000G                         .WORD   SC.HWP
000120  000000G                TBL.DAT: .WORD   SC.FE2
000122  000000G                         .WORD   NULL
000124  000000G                         .WORD   SC.IS2
000126  000000G                         .WORD   SC.DS2
000130  000000G                         .WORD   SC.EC9
000132  000000G                         .WORD   NULL
000134  000000G                         .WORD   NULL
000136  000000G                         .WORD   SC.ECD
000140  000000G                         .WORD   SC.EC1
000142  000000G                         .WORD   SC.EC2
000144  000000G                         .WORD   SC.EC3
000146  000000G                         .WORD   SC.EC4
000150  000000G                         .WORD   SC.EC5
000152  000000G                         .WORD   SC.EC6
000154  000000G                         .WORD   SC.EC7
000156  000000G                         .WORD   SC.EC8
000160  000000G                TBL.HST: .WORD   NULL
000162  000000G                         .WORD   SC.ODA
000164  000000G                         .WORD   SC.ODB
000166  000000G                         .WORD   SC.NXM
000170  000000G                         .WORD   SC.PAR
000172  000000G                TBL.CNT: .WORD   SC.CTO
000174  000000G                         .WORD   SC.SDS
000176  000000G                         .WORD   SC.EDC
000200  000000G                         .WORD   SC.IDS
000202  000000G                TBL.DRV: .WORD   NULL
000204  000000G                         .WORD   SC.SRT
000206  000000G                         .WORD   SC.SRI
000210  000000G                         .WORD   SC.POE
000212  000000G                         .WORD   SC.RDY
000214  000000G                         .WORD   SC.CLK
000216  000000G                         .WORD   SC.RSP
000220  000000G                         .WORD   SC.SUR
000222  000000G                         .WORD   SC.PSP

                                .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                                .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                                .GLOBL  DUPPKT, TRK.SGN, RDM.CNT, RANDOM
                                .GLOBL  C.ERR.TBL, MSCP.PKT, IPKT.ADDR
                                .GLOBL  PKT.USE, RETPKT, RP.USE, RP.INDX
                                .GLOBL  RP.ADDR, ELOG.PKT, BUFF.ADDR, BUFF.OWN
                                .GLOBL  IODQ, IODQ.IN, IODQ.OUT, ENTRY.REASON
                                .GLOBL  EOP.FLAG, DUP.FLAGS, CCTLR, CDISK
                                .GLOBL  CUOFF, CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
                                .GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
                                .GLOBL  STEP, OF.RC, SA.REG, CMD.TIME
                                .GLOBL  NEX, CRN.LOW, CRN.HIGH, CREDIT.BAL
                                .GLOBL  NEXT.PKT.USE, HOURS, MINUTES, CLK.TICKS
```

```
                .GLOBL  CLK.PRESENT, HOE.FLAG, FORCED.ERROR
                .GLOBL  FER.LBN, FER.BC, INIT.OCCURED
                .GLOBL  ADDR.VECT.OK, DBM5, P.INDEX, S.PATTERN
                .GLOBL  S.DUPPKT, DBM107, DU.MSG, DU.RSN
                .GLOBL  RPT1, RPT2, RPT3, RPT4, RPT5, RPT6
                .GLOBL  RPT7, RPT8, RPT9, RPT10, RPT11
                .GLOBL  RPT12, RPT13, RPT14, RPT15, RPT16
                .GLOBL  MSG.01, EGS.01, EBS.01, EBD.10
                .GLOBL  EBD.12, EBD.13, EBD.14, EBD.18
                .GLOBL  EBD.19, EBD.24, ERR.00, ERR.COD
                .GLOBL  ELG.00, ELG.FMT, EX.TIM, XX13
                .GLOBL  XX23, XX32, XX33, XX34, EX.SA
                .GLOBL  EX.SC, EX.SBO, EX.SB, EX.RP, EX.WRD
                .GLOBL  EX.CMD, EX.RD, EX.WRT, EX.CMP
                .GLOBL  EX.ONL, EX.ACC, EX.OP, EX.BB, EX.BB1
                .GLOBL  EX.BBU, EX.LBN, EX.PBN, EX.LBR
                .GLOBL  EX.LBW, EX.RBN, EX.CBC, EX.CBR
                .GLOBL  EX.CBW, EX.BC, EX.BD, EX.BDR, EX.BDW
                .GLOBL  SC.SDI, SC.CON, SC.DUP, SC.ONL
                .GLOBL  SC.SON, SC.UNK, SC.VOL, SC.IOP
                .GLOBL  SC.DIS, SC.FER, SC.FE2, SC.ISH
                .GLOBL  SC.IS2, SC.DST, SC.DS2, SC.ECC
                .GLOBL  SC.ECD, SC.RCT, SC.FUL, SC.576
                .GLOBL  SC.FCT, SC.SWP, SC.HWP, SC.EC1
                .GLOBL  SC.EC2, SC.EC3, SC.EC4, SC.EC5
                .GLOBL  SC.EC6, SC.EC7, SC.EC8, SC.EC9
                .GLOBL  SC.ODA, SC.ODB, SC.NXM, SC.PAR
                .GLOBL  SC.CTO, SC.SDS, SC.EDC, SC.IDS
                .GLOBL  SC.SRT, SC.SRI, SC.POE, SC.RDY
                .GLOBL  SC.CLK, SC.RSP, SC.SUR, SC.PSP
                .GLOBL  CER.01, CER.02, CNTR.ERR, RDRX.ERR
                .GLOBL  SPACE4, CRLF, DASH, ASTERISK, HWQ1
                .GLOBL  HWQ2, HWQ3, HWQ4, HWQ5, HWQ6A
                .GLOBL  HWQ6B, HWQ7A, HWQ7B, HWQ8, HWQ9
                .GLOBL  HWQ10, HWQ11, SWQ1, SWQ2, SWQ4
                .GLOBL  SWQ7, SWQ9, SWQ10, SWQ11, SWQ12
                .GLOBL  SWQ13, SWQ14, SWQ15, SWQ17, SWQ19
                .GLOBL  SWQ20, SWQ21, SWQ22, SWQ23, SWQ24
                .GLOBL  SWQ25, EH.0, EH.1, EH.2, EH.3
                .GLOBL  EH.4, EH.5, EH.6, EH.7, EH.8, EH.9
                .GLOBL  EH.10, EH.12, EH.13, SWM1, NULL
                .GLOBL  SWP.FLAGS, L$HIMEM, L$LUN, L$UNIT

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==          1000
        000400                  BIT8==          400
        000200                  BIT7==          200
        000100                  BIT6==          100
        000040                  BIT5==          40
        000020                  BIT4==          20
        000010                  BIT3==          10
        000004                  BIT2==          4
        000002                  BIT1==          2
        000001                  BIT0==          1
        000040                  EF.START==      40
        000037                  EF.RESTART==    37
        000036                  EF.CONTINUE==   36
        000035                  EF.NEW==        35
        000034                  EF.PWR==        34
        000340                  PRI07==         340
        000300                  PRI06==         300
        000240                  PRI05==         240
        000200                  PRI04==         200
        000140                  PRI03==         140
        000100                  PRI02==         100
        000040                  PRI01==         40
        000000                  PRI00==         0
        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
        000044'                 L$HARD==        L$HRDLN+2
        000240'                 L$SOFT==        L$SFTLN+2


                                        .SBTTL  LRPT REPORT CODING SECTION
000522                                  .PSECT  $CODE$,  RO

000000  004137  000000G        LRPT:   JSR     R1,$SAVE4                       ;                       5154
000004  104440                          TRAP    40                              ;                       5171
```

```
000006  010004                          MOV     R0,R4                           ; *,CUR.PRIORITY
000010  012700  000200                  MOV     #200,R0                         ;                               5172
000014  104441                          TRAP    41
000016  012746  000000G                 MOV     #RPT1,-(SP)                     ;                               5174
000022  012746  000001                  MOV     #1,-(SP)
000026  010600                          MOV     SP,R0                           ; SP,*
000030  104416                          TRAP    16
000032  012716  000000G                 MOV     #RPT2,(SP)                      ;                               5175
000036  012746  000001                  MOV     #1,-(SP)
000042  010600                          MOV     SP,R0                           ; SP,*
000044  104416                          TRAP    16
000046  012716  000000G                 MOV     #RPT3,(SP)                      ;                               5176
000052  012746  000001                  MOV     #1,-(SP)
000056  010600                          MOV     SP,R0                           ; SP,*
000060  104416                          TRAP    16
000062  012716  000000G                 MOV     #RPT4,(SP)                      ;                               5177
000066  012746  000001                  MOV     #1,-(SP)
000072  010600                          MOV     SP,R0                           ; SP,*
000074  104416                          TRAP    16
000076  012716  000000G                 MOV     #RPT5,(SP)                      ;                               5178
000102  012746  000001                  MOV     #1,-(SP)
000106  010600                          MOV     SP,R0                           ; SP,*
000110  104416                          TRAP    16
000112  012716  000000G                 MOV     #RPT6,(SP)                      ;                               5179
000116  012746  000001                  MOV     #1,-(SP)
000122  010600                          MOV     SP,R0                           ; SP,*
000124  104416                          TRAP    16
000126  005002                          CLR     R2                              ; CTLR                          5181
000130  010216          1$:             MOV     R2,(SP)                         ; CTLR,*                        5184
000132  004737  000000V                 JSR     PC,SET.CPAR
000136  012703  000003                  MOV     #3,R3                           ; *,DISK                        5186
000142  010316          2$:             MOV     R3,(SP)                         ; DISK,*                        5189
000144  004737  000000V                 JSR     PC,SET.UPAR
000150  010301                          MOV     R3,R1                           ; DISK,*                        5192
000152  006301                          ASL     R1
000154  063701  000000G                 ADD     CST.ADDR,R1
000160  032711  040000                  BIT     #40000,(R1)
000164  001535                          BEQ     3$
000166  010216                          MOV     R2,(SP)                         ; CTLR,*                        5197
000170  012746  000053                  MOV     #53,-(SP)
000174  004737  000000G                 JSR     PC,BL$MUL
000200  060300                          ADD     R3,R0                           ; DISK,*
000202  006300                          ASL     R0
000204  062700  000000G                 ADD     #CST,R0
000210  010016                          MOV     R0,(SP)
000212  062716  000012                  ADD     #12,(SP)
000216  111146                          MOVB    (R1),-(SP)
000220  042716  177760                  BIC     #177760,(SP)
000224  013746  000000G                 MOV     L$LUN,-(SP)
000230  012746  000000G                 MOV     #RPT7,-(SP)
000234  012746  000004                  MOV     #4,-(SP)
000240  010600                          MOV     SP,R0                           ; SP,*
000242  104416                          TRAP    16
```

```
000244  013700  000000G             MOV     T.ADDR,R0                       ;                               5200
000250  016016  000032              MOV     32(R0),(SP)
000254  016046  000034              MOV     34(R0),-(SP)
000260  016046  000036              MOV     36(R0),-(SP)
000264  016046  000016              MOV     16(R0),-(SP)
000270  016046  000020              MOV     20(R0),-(SP)
000274  012746  000000G             MOV     #RPT8,-(SP)
000300  012746  000006              MOV     #6,-(SP)
000304  010600                      MOV     SP,R0                           ; SP,*
000306  104416                      TRAP    16
000310  013700  000000G             MOV     T.ADDR,R0                       ;                               5203
000314  016016  000040              MOV     40(R0),(SP)
000320  016046  000042              MOV     42(R0),-(SP)
000324  016046  000044              MOV     44(R0),-(SP)
000330  016046  000024              MOV     24(R0),-(SP)
000334  016046  000026              MOV     26(R0),-(SP)
000340  012746  000000G             MOV     #RPT8,-(SP)
000344  012746  000006              MOV     #6,-(SP)
000350  010600                      MOV     SP,R0                           ; SP,*
000352  104416                      TRAP    16
000354  013700  000000G             MOV     T.ADDR,R0                       ;                               5206
000360  005016                      CLR     (SP)
000362  116016  000055              MOVB    55(R0),(SP)
000366  005046                      CLR     -(SP)
000370  116016  000054              MOVB    54(R0),(SP)
000374  005046                      CLR     -(SP)
000376  116016  000053              MOVB    53(R0),(SP)
000402  005046                      CLR     -(SP)
000404  116016  000052              MOVB    52(R0),(SP)
000410  005046                      CLR     -(SP)
000412  116016  000051              MOVB    51(R0),(SP)
000416  005046                      CLR     -(SP)
000420  116016  000050              MOVB    50(R0),(SP)
000424  005046                      CLR     -(SP)
000426  116016  000047              MOVB    47(R0),(SP)
000432  005046                      CLR     -(SP)
000434  116016  000046              MOVB    46(R0),(SP)
000440  012746  000000G             MOV     #RPT9,-(SP)
000444  012746  000011              MOV     #11,-(SP)
000450  010600                      MOV     SP,R0                           ; SP,*
000452  104416                      TRAP    16
000454  062706  000064              ADD     #64,SP                          ;                               5195
000460  062703  000012      3$:     ADD     #12,R3                          ; *,DISK                        5186
000464  020327  000041              CMP     R3,#41                          ; DISK,*
000470  003624                      BLE     2$
000472  010216                      MOV     R2,(SP)                         ; CTLR,*                        5211
000474  012746  000126              MOV     #126,-(SP)
000500  004737  GJ0000G             JSR     PC,BL$MUL
000504  005726                      TST     (SP)+
000506  005760  000002G             TST     CST+2(R0)
000512  100026                      BPL     4$
000514  012716  000000G             MOV     #RPT10,(SP)                     ;                               5215
000520  012746  000001              MOV     #1,-(SP)
```

```
000524  010600                          MOV     SP,R0                   ; SP,*
000526  104416                          TRAP    16
000530  010200                          MOV     R2,R0                   ; CTLR,*                5216
000532  006300                          ASL     R0
000534  005016                          CLR     (SP)
000536  116016  000001G                 MOVB    C.ERR.TBL+1(R0),(SP)
000542  005046                          CLR     -(SP)
000544  116016  000000G                 MOVB    C.ERR.TBL(R0),(SP)
000550  012746  000000G                 MOV     #RPT11,-(SP)
000554  012746  000003                  MOV     #3,-(SP)
000560  010600                          MOV     SP,R0                   ; SP,*
000562  104416                          TRAP    16
000564  062706  000010                  ADD     #10,SP                  ;                       5214
000570  005202                  4$:     INC     R2                      ; CTLR                  5181
000572  000243                          .WORD CLV!CLC

000574  003002                          BGT     5$
000576  000137  000652'                JMP     1$
000602  010400                  5$:     MOV     R4,R0                   ; CUR.PRIORITY,*        5222
000604  104441                          TRAP    41
000606  012716  000000G                 MOV     #CRLF,(SP)              ;                       5226
000612  012746  000001                  MOV     #1,-(SP)
000616  010600                          MOV     SP,R0                   ; SP,*
000620  104416                          TRAP    16
000622  012716  000000G                 MOV     #RPT13,(SP)             ;                       5227
000626  012746  000001                  MOV     #1,-(SP)
000632  010600                          MOV     SP,R0                   ; SP,*
000634  104416                          TRAP    16
000636  012716  000000G                 MOV     #RPT14,(SP)             ;                       5228
000642  012746  000001                  MOV     #1,-(SP)
000646  010600                          MOV     SP,R0                   ; SP,*
000650  104416                          TRAP    16
000652  012716  000000G                 MOV     #RPT15,(SP)             ;                       5229
000656  012746  000001                  MOV     #1,-(SP)
000662  010600                          MOV     SP,R0                   ; SP,*
000664  104416                          TRAP    16
000666  005003                          CLR     R3                      ; CTLR                  5230
000670  010316                  6$:     MOV     R3,(SP)                 ; CTLR,*                5232
000672  004737  000000V                 JSR     PC,SET.CPAR
000676  012702  000003                  MOV     #3,R2                   ; *,DISK                5233
000702  010216                  7$:     MOV     R2,(SP)                 ; DISK,*                5235
000704  004737  000000V                 JSR     PC,SET.UPAR
000710  010201                          MOV     R2,R1                   ; DISK,*                5236
000712  006301                          ASL     R1
000714  063701  000000G                 ADD     CST.ADDR,R1
000720  132711  000020                  BITB    #20,(R1)
000724  001432                          BEQ     8$
000726  032711  040000                  BIT     #40000,(R1)
000732  001427                          BEQ     8$
000734  013700  000000G                 MOV     T.ADDR,R0               ;                       5240
000740  016016  000056                  MOV     56(R0),(SP)
000744  016046  000060                  MOV     60(R0),-(SP)
000750  016046  000062                  MOV     62(R0),-(SP)
```

```
000754  016046  000064                        MOV     64(R0),-(SP)
000760  111146                                MOVB    (R1),-(SP)
000762  042716  177760                        BIC     #177760,(SP)
000766  013746  000000G                       MOV     L$LUN,-(SP)
000772  012746  000000G                       MOV     #RPT16,-(SP)
000776  012746  000007                        MOV     #7,-(SP)
001002  010600                                MOV     SP,R0           ; SP,*
001004  104416                                TRAP    16
001006  062706  000016                        ADD     #16,SP
001012  062702  000012              8$:       ADD     #12,R2          ; *,DISK                5233
001016  020227  000041                        CMP     R2,#41          ; DISK,*
001022  003727                                BLE     7$
001024  005203                                INC     R3              ; CTLR                  5230
001026  000243                                .WORD CLV!CLC

001030  003717                                BLE     6$
001032  012716  000000G                       MOV     #CRLF,(SP)      ;                       5249
001036  012746  000001                        MOV     #1,-(SP)
001042  010600                                MOV     SP,R0           ; SP,*
001044  104416                                TRAP    16
001046  062706  000030                        ADD     #30,SP          ;                       5154
001052  000207                                RTS     PC

; Routine Size:  278 words,     Routine Base:  $CODE$ + 0522
; Maximum stack depth per invocation:  40 words



                                              .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000522'              L$RPT:: JSR     PC,LRPT         ;                       5250
000004  104425                                TRAP    25
000006  000207                                RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 1576
; Maximum stack depth per invocation:  2 words


;   5253  1
```

```
;    5254  1     %sbttl 'INITIALIZE SECTION'
;    5255  1
;    5256  2     BGNINIT;
;    5257  2
;    5258  2     local
;    5259  2         DELAY_MULT : word,
;    5260  2         FLAG : byte,
;    5261  2         TEMP : word,
;    5262  2         HWPT_REF : ref block [HWPT_LEN, word] field (HWP_FIELDS),
;    5263  2         CLEAR_TABLES : byte,
;    5264  2         SMALLEST_DRIVE : byte,
;    5265  2         BLANKS : WORD INITIAL (%o'020040'),                    !ZZZ
;    5266  2         HWPT_ADDRESS : vector [MAX_UNITS, word];
;    5267  2
;    5268  2     SETPRI (PRI07);                                            ! NO INTERRUPTS ALLOWED DURING INIT
;    5269  2
;    5270  2     if READEF (EF_NEW)                                         ! IS THIS A NEW PASS?
;    5271  2     then
;    5272  3         begin
;    5273  3         ENTRY_REASON = NEW_PASS;
;    5274  3
;    5275  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5276  3         then
;    5277  3             CLEAR_TABLES = FALSE
;    5278  3         else
;    5279  3             CLEAR_TABLES = TRUE;
;    5280  3
;    5281  2         end;
;    5282  2
;    5283  2     if READEF (EF_START)                                       ! IS THIS A START?
;    5284  2     then
;    5285  3         begin
;    5286  3         BRESET;
;    5287  3         ENTRY_REASON = START;
;    5288  3         CLEAR_TABLES = TRUE;
;    5289  3         ADDR_VECT_OK = FALSE;
;    5290  3         INIT_OCCURED = FALSE;
;    5291  2         end;
;    5292  2
;    5293  2     if READEF (EF_RESTART)                                     ! IS THIS A RESTART?
;    5294  2     then
;    5295  3         begin
;    5296  3         ENTRY_REASON = RESTART;
;    5297  3         CLEAR_TABLES = TRUE;
;    5298  2         end;
;    5299  2
;    5300  2     if READEF (EF_CONTINUE)                                    ! IS THIS A CONTINUE?
;    5301  2     then
;    5302  3         begin
;    5303  3         ENTRY_REASON = CONT;
;    5304  3
;    5305  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5306  3         then
```

```
;    5307  3              CLEAR_TABLES = FALSE
;    5308  3          else
;    5309  3              CLEAR_TABLES = TRUE;
;    5310  3
;    5311  2          end;
;    5312  2
;    5313  2      if READEF (EF_PWR)                                  ! ARE WE HERE BECAUSE OF POWER FAIL
;    5314  2      then
;    5315  3          begin
;    5316  3          ENTRY_REASON = PWR_FAIL;
;    5317  3          ADDR_VECT_OK = FALSE;
;    5318  3          INIT_OCCURED = FALSE;
;    5319  3          CLEAR_TABLES = TRUE;
;    5320  3          PRINTF (MSG_01);                                ! "POWER DELAY - WAITING"
;    5321  3
;    5322  3          incr COUNT from 0 to 60 do                      ! WAIT APPROX. 60 SECONDS
;    5323  4              begin
;    5324  4              DELAY_MULT = 333;
;    5325  4              DELAY (.DELAY_MULT);
;    5326  4              BREAK;                                      ! BREAK FOR ACT
;    5327  3              end;
;    5328  3
;    5329  2          end;
;    5330  2
;    5331  2      !SETVEC (O_TVEC, O_BRK, PRI07);                     ! SET ODT TRAP VECTOR
;    5332  2
;    5333  2      !+
;    5334  2      !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
;    5335  2      !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
;    5336  2      !       (DIAGNOSTIC IS ABORTED).
;    5337  2      !-
;    5338  2
;    5339  2      if .L$UNIT gtru MAX_UNITS
;    5340  2      then
;    5341  3          begin
;    5342  3          ERRSF (1, EGS_01, EMS_01);
;    5343  3          DOCLN;
;    5344  2          end;
;    5345  2
;    5346  2      !+
;    5347  2      !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
;    5348  2      !       ALL RUN-TIME CONTROLLER STATUS TABLES (CSTs) ARE CLEARED TO 0, THEN
;    5349  2      !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
;    5350  2      !-
;    5351  2
;    5352  2      if .ENTRY_REASON neq NEW_PASS
;    5353  2      then
;    5354  3          begin
;    5355  3          SMALLEST_DRIVE = 255;                           ! LARGEST DISK NO. ALLOWED BY MSCP
;    5356  3
;    5357  3          incr COUNT from 0 to ((MAX_CTLR * CST_LEN * 2) - 2) by 2 do
;    5358  3              (CST + .COUNT) = 0;
;    5359  3
```

```
;    5360  3          incr UNIT from 0 to (.L$UNIT - 1) do                          ! LOOP THROUGH ALL UNITS
;    5361  3
;    5362  3              if (HWPT_ADDRESS [.UNIT] = GPHARD (.UNIT, HWPT_REF)) neqa 0   ! IF HWP TABLE FOUND
;    5363  3              then
;    5364  3
;    5365  3                  if .HWPT_REF [HWP_DISK_NUM] lssu .SMALLEST_DRIVE          ! FIND OUT THE SMALLEST DISK NUMBER
;    5366  3                  then
;    5367  3                      SMALLEST_DRIVE = .HWPT_REF [HWP_DISK_NUM];
;    5368  3
;    5369  3          incr UNIT from 0 to (.L$UNIT - 1) do                          ! LOOP THROUGH ALL UNITS
;    5370  3
;    5371  3              if .HWPT_ADDRESS [.UNIT] neqa 0                           ! IF HWP TABLE FOUND
;    5372  3              then
;    5373  4                  begin
;    5374  4                  FLAG = NOT_FOUND;
;    5375  4                  HWPT_REF = .HWPT_ADDRESS [.UNIT];
;    5376  4
;    5377  4                  incr CTLR from 0 to (MAX_CTLR - 1) do                 ! LOOP THROUGH ALL CSTs
;    5378  4
;    5379  4                      if .CST [.CTLR, IP_ADDR] eqla .HWPT_REF [HWP_IP_ADDR]
;    5380  4                      then
;    5381  4
;    5382  4                          if .CST [.CTLR, (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE
;    5383  4                                  + OF_UN + OF_DATA, D_PRES] eql NOT_PRESENT
;    5384  4                          then
;    5385  5                              begin                                     ! IF EMPTY SLOT FOUND
;    5386  5                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;    5387  5                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    5388  5                                                                        ! COPY DISK ADDR AND PROT BIT
;    5389  5                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    5390  5                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    5391  5                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    5392  5
;    5393  5                              IF .HWPT_REF [HWP_ENTIRE] EQL TRUE        !ZZZ IF DEFAULT TEST RANGE,
;    5394  5                              THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;  !ZZZ MAKE HI ADDR ALL ONES
;    5395  5
;    5396  5                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =
;    5397  5                                      .HWPT_REF [HWP_BEG_TRK];          !ZZZ
;    5398  5                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =    !ZZZ
;    5399  5                                      .HWPT_REF [HWP_BEG_TRK1];         !ZZZ
;    5400  5                              CST [.CTLR, .TEMP + OF_END, D_END0] =     !ZZZ
;    5401  5                                      .HWPT_REF [HWP_END_TRK];          !ZZZ
;    5402  5                              CST [.CTLR, .TEMP + OF_END1, D_END1] =    !ZZZ
;    5403  5                                      .HWPT_REF [HWP_END_TRK1];         !ZZZ
;    5404  5
;    5405  5                              CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;    !ZZZ BLANK NAME
;    5406  5                              CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;    !ZZZ BLANK NAME
;    5407  5
;    5408  5
;    5409  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;    5410  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
;    5411  5                                      NOT (.HWPT_REF [HWP_DISK_DUPEX]);       !ZZZ
;    5412  5
```

```
;    5413  5                          CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =     !ZZZ
;    5414  5                              (.HWPT_REF [HWP_DISK_DUPWT]);                !ZZZ
;    5415  5                          CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;      !ZZZ
;    5416  5                          FLAG = FOUND;
;    5417  5                          exitloop;
;    5418  5                          end
;    5419  4                      else
;    5420  5                          begin                                            ! DUPLICATE UNIT
;    5421  5                          PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR]);
;    5422  5                                                                           ! "DUPLICATE UNIT: X AT IP: XXXXXX"
;    5423  5                          DUR [.UNIT] = DU_CONF;                           ! CONFIGURATION ERROR
;    5424  5                          DODU (.UNIT);                                    ! DROP UNIT
;    5425  5                          FLAG = FOUND;
;    5426  5                          exitloop;
;    5427  4                          end;
;    5428  4
;    5429  4                  if .FLAG eql NOT_FOUND                                   ! IF NO IP MATCH TO EXISTING CST
;    5430  4                  then
;    5431  5                      begin
;    5432  5
;    5433  5                      incr CTLR from 0 to (MAX_CTLR - 1) do                ! LOOP THROUGH EACH CST
;    5434  5
;    5435  5                          if .CST [.CTLR, IP_ADDR] eqla 0                  ! IF EMPTY CST FOUND
;    5436  5                          then
;    5437  6                              begin
;    5438  6                              CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;    5439  6                              CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;    5440  6                              CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;    5441  6                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;    5442  6                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    5443  6                                                                           ! COPY DISK ADDR AND PROT BIT
;    5444  6                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    5445  6                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    5446  6                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    5447  6
;    5448  6                              IF .HWPT_REF [HWP_ENTIRE] EQL TRUE           !ZZZ IF DEFAULT TEST RANGE,
;    5449  6                              THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;     !ZZZ MAKE HI ADDR ALL ONES
;    5450  6
;    5451  6                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =        !ZZZ
;    5452  6                                  .HWPT_REF [HWP_BEG_TRK];                 !ZZZ
;    5453  6                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =       !ZZZ
;    5454  6                                  .HWPT_REF [HWP_BEG_TRK1];                !ZZZ
;    5455  6                              CST [.CTLR, .TEMP + OF_END, D_END0] =        !ZZZ
;    5456  6                                  .HWPT_REF [HWP_END_TRK];                 !ZZZ
;    5457  6                              CST [.CTLR, .TEMP + OF_END1, D_END1] =       !ZZZ
;    5458  6                                  .HWPT_REF [HWP_END_TRK1];                !ZZZ
;    5459  6
;    5460  6                              CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;     !ZZZ BLANK NAME
;    5461  6                              CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;     !ZZZ BLANK NAME
;    5462  6
;    5463  6
;    5464  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;     !ZZZ
;    5465  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =   !ZZZ
```

```
                                NOT (.HWPT_REF [HWP_DISK_DUPEX]);                !ZZZ
                        CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =     !ZZZ
                                (.HWPT_REF [HWP_DISK_DUPWT]);            !ZZZ
                        CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;      !ZZZ
                        FLAG = FOUND;
                        exitloop;
                        end;                                                     ! IF EMPTY CST FOUND

                if .FLAG eql NOT_FOUND                                           ! IF NO EMPTY CST FOUND
                then
                    begin
                    PRINTF (CER_02, MAX_CTLR);                                   ! "MORE THAN X IP ADDRESSES."
                    DUR [.UNIT] = DU_CONF;                                       ! CONFIGURATION ERROR
                    DODU (.UNIT);                                                ! DROP UNIT
                    end;

                end;                                                             ! IF NO IP ADDR MATCH IN CST

            end;                                                                 ! IF GPHARD RETURNS A HWP TABLE
        !
        ! CONFIGURATON CHECK FOR LEGAL RDRX UNIT MIX BECAUSE WE HAVE DIFFERENT
        ! DRIVES :  THE RD51, RD52, AND RX50.
        ! (NEEDED?)
        !
        end;                                                                     ! END OF "NON NEW_PASS" INIT

    if .ENTRY_REASON eql NEW_PASS
    then
        begin

        incr UNIT from 0 to (.L$UNIT - 1) do
            GPHARD (.UNIT, HWPT_REF);                                            ! DUMMY GPHARDs FOR NEW PASS

        incr CTLR from 0 to (MAX_CTLR - 1) do
            begin
            CST [.CTLR, U_CNT] = 0;                                              ! REINITIALIZE UNIT COUNT

            incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
                CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;                ! START EACH UNIT AS OFFLINE

            end;

        end;

    if .ENTRY_REASON eql START
    then
        begin
        CTLR_CNT = 0;                                                            ! NUMBER OF CONFIGURED CONTROLLERS

        incr CTLR from 0 to (MAX_CTLR - 1) do

            if .CST [.CTLR, IP_ADDR] neqa 0                                      ! IF CONTROLLER IS PRESENT
            then
```

```
;    5519  3                CTLR_CNT = .CTLR_CNT + 1;                          ! INCREMENT CONTROLLER COUNT
;    5520  3
;    5521  3            MEMORY (FREE_MEM_ADDR);                                ! GET START OF FREE MEMORY
;    5522  3
;    5523  2            end;                                                   ! END OF "START" INITIALIZATION
;    5524  2
;    5525  2        !+
;    5526  2        !       CLEAR STATISTICS TABLES
;    5527  2        !-
;    5528  2
;    5529  2        incr UNITS from 0 to MAX_UNITS - 1 do                      ! CLEAR CURRENT STATISTICS
;    5530  2            incr COUNT from 0 to TALLY_CLEAR - 1 do
;    5531  2                TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;
;    5532  2
;    5533  2        if .CLEAR_TABLES                                           ! IF CLEAR TABLES ON EVERY PASS
;    5534  2        then
;    5535  2            incr UNITS from 0 to MAX_UNITS - 1 do
;    5536  2                incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do    ! INITIALIZE TOTALS
;    5537  2                    TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;       !
;    5538  2
;    5539  2        if .CLEAR_TABLES
;    5540  2        then
;    5541  2            incr CTLR from 0 to MAX_CTLR - 1 do
;    5542  3                begin
;    5543  3                C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;                  ! INITALIZE CONTROLLER ERRORS
;    5544  3                C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;                  !
;    5545  2                end;
;    5546  2
;    5547  2        !+
;    5548  2        !       MISCELLANEOUS INITIALIZATON
;    5549  2        !-
;    5550  2
;    5551  2        incr CTLR from 0 to (MAX_CTLR - 1) do                      ! INIT NO. OF OUTSTANDING QIOs
;    5552  2            QIO [.CTLR] = 0;
;    5553  2
;    5554  2        incr COUNT from 0 to (RP_CNT - 1) do                       ! INITIALIZE RETURN PACKET POOL
;    5555  2            RP_USE [.COUNT] = -1;
;    5556  2
;    5557  2        if .CLK_PRESENT                                            ! STOP CLOCK IF PRESENT
;    5558  2        then
;    5559  2            LINE_CLOCK = 0;
;    5560  2
;    5561  2        IODQ_IN = IODQ_OUT = 0;                                    ! INIT I/O DONE QUEUE POINTERS
;    5562  2        CRN_LOW = CRN_HIGH = 0;                                    ! INIT COMMAND REFERENCE NUMBER
;    5563  2        SETPRI (PRI00);                                            ! SET PROGRAM PRIORITY TO 0
;    5564  2
;    5565  1        ENDINIT;


                                .GLOBL  L$DLY

                                .SBTTL  LINIT INITIALIZE SECTION
```

```
000000  004137  000000G                LINIT:  JSR     R1,$SAVE5                       ;                               5252
000004  162706  000030                          SUB     #30,SP
000010  012746  020040                          MOV     #20040,-(SP)                    ; *,BLANKS
000014  012700  000340                          MOV     #340,R0                         ;                               5268
000020  104441                                  TRAP    41
000022  012700  000035                          MOV     #35,R0                          ;                               5270
000026  104447                                  TRAP    47
000030  103014                                  BHIS    2$
000032  112737  000005  000000G                MOVB    #5,ENTRY.REASON                 ;                               5273
000040  105737  000000G                        TSTB    SWP.FLAGS                       ;                               5275
000044  100403                                  BMI     1$
000046  105066  000012                        1$:     CLRB    12(SP)                  ; CLEAR.TABLES                  5277
000052  000403                                  BR      2$                              ;                               5275
000054  112766  000001  000012          2$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                5279
000062  012700  000040                          MOV     #40,R0                          ;                               5283
000066  104447                                  TRAP    47
000070  103013                                  BHIS    3$
000072  104433                                  TRAP    33                              ;                               5285
000074  112737  000001  000000G                MOVB    #1,ENTRY.REASON                 ;                               5287
000102  112766  000001  000012                  MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                5288
000110  105037  000000G                        CLRB    ADDR.VECT.OK                    ;                               5289
000114  105037  000000G                        CLRB    INIT.OCCURED                    ;                               5290
000120  012700  000037                  3$:     MOV     #37,R0                          ;                               5293
000124  104447                                  TRAP    47
000126  103006                                  BHIS    4$
000130  112737  000002  000000G                MOVB    #2,ENTRY.REASON                 ;                               5296
000136  112766  000001  000012                  MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                5297
000144  012700  000036                  4$:     MOV     #36,R0                          ;                               5300
000150  104447                                  TRAP    47
000152  103014                                  BHIS    6$
000154  112737  000003  000000G                MOVB    #3,ENTRY.REASON                 ;                               5303
000162  105737  000000G                        TSTB    SWP.FLAGS                       ;                               5305
000166  100403                                  BMI     5$
000170  105066  000012                        5$:     CLRB    12(SP)                  ; CLEAR.TABLES                  5307
000174  000403                                  BR      6$                              ;                               5305
000176  112766  000001  000012          6$:     MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                5309
000204  012700  000C34                  7$:     MOV     #34,R0                          ;                               5313
000210  104447                                  TRAP    47
000212  103043                                  BHIS    12$
000214  112737  000004  000000G                MOVB    #4,ENTRY.REASON                 ;                               5316
000222  105037  000000G                        CLRB    ADDR.VECT.OK                    ;                               5317
000226  105037  000000G                        CLRB    INIT.OCCURED                    ;                               5318
000232  112766  000001  000012                  MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                5319
000240  012746  000000G                        MOV     #MSG.01,-(SP)                   ;                               5320
000244  012746  000001                          MOV     #1,-(SP)
000250  010600                                  MOV     SP,R0                           ; SP,*
000252  104417                                  TRAP    17
000254  012702  000075                          MOV     #75,R2                          ; *,COUNT                       5322
000260  012703  000515                  8$:     MOV     #515,R3                         ; *,DELAY.MULT                  5324
000264  010301                                  MOV     R3,R1                           ; DELAY.MULT,$$TMP2             5325
000266  001411                                  BEQ     11$
000270  013700  000000G                        MOV     L$DLY,R0                        ; *,$$TMP1
000274  001404                                  BEQ     10$
```

```
000276  005066  000024                  9$:     CLR     24(SP)                          ; $$TMP
000302  005300                                  DEC     R0                              ; $$TMP1
000304  001374                                  BNE     9$
000306  005301                          10$:    DEC     R1                              ; $$TMP2
000310  000766                                  BR      8$
000312  104422                          11$:    TRAP    22
000314  005302                                  DEC     R2                              ; COUNT                         5322
000316  001360                                  BNE     7$
000320  022626                                  CMP     (SP)+,(SP)+                     ;                               5315
000322  023727  000000G 000004          12$:    CMP     L$UNIT,#4                       ;                               5339
000330  101405                                  BLOS    13$
000332  104454                                  TRAP    54                              ;                               5342
000334  000001                                  .WORD   1
000336  000000G                                 .WORD   EGS.01
000340  000000V                                 .WORD   EMS.01
000342  104444                                  TRAP    44
000344  123727  000000G 000005          13$:    CMPB    ENTRY.REASON,#5                 ;                               5352
000352  001002                                  BNE     14$
000354  000137  003706'                         JMP     43$
000360  112766  000377  000010          14$:    MOVB    #377,10(SP)                     ; *,SMALLEST.DRIVE              5355
000366  005000                                  CLR     R0                              ; COUNT                         5357
000370  005060  000000G                 15$:    CLR     CST(R0)                         ; *(COUNT)                      5358
000374  062700  000002                          ADD     #2,R0                           ; *,COUNT                       5357
000400  020027  000124                          CMP     R0,#124                         ; COUNT,*
000404  003771                                  BLE     15$
000406  013704  000000G                         MOV     L$UNIT,R4                       ;                               5360
000412  005003                                  CLR     R3                              ; UNIT
000414  000435                                  BR      18$
000416  010302                          16$:    MOV     R3,R2                           ; UNIT,*                        5362
000420  006302                                  ASL     R2
000422  012700  000022                          MOV     #22,R0
000426  060600                                  ADD     SP,R0                           ; HWPT.ADDRESS,*
000430  060002                                  ADD     R0,R2
000432  010300                                  MOV     R3,R0                           ; UNIT,*
000434  104442                                  TRAP    42
000436  010001                                  MOV     R0,R1                           ; *,HWPT.REF
000440  010112                                  MOV     R1,(R2)                         ; HWPT.REF,*
000442  001421                                  BEQ     17$
000444  005002                                  CLR     R2                              ;                               5365
000446  156602  000010                          BISB    10(SP),R2                       ; SMALLEST.DRIVE,*
000452  116100  000006                          MOVB    6(R1),R0                        ; *(HWPT.REF),*
000456  042700  177760                          BIC     #177760,R0
000462  020002                                  CMP     R0,R2
000464  103010                                  BHIS    17$
000466  116100  000006                          MOVB    6(R1),R0                        ; *(HWPT.REF),*                 5367
000472  042700  177760                          BIC     #177760,R0
000476  105066  000010                          CLRB    10(SP)                          ; SMALLEST.DRIVE
000502  050066  000010                          BIS     R0,10(SP)                       ; *,SMALLEST.DRIVE
000506  005203                          17$:    INC     R3                              ; UNIT                          5360
000510  020304                          18$:    CMP     R3,R4                           ; UNIT,*
000512  002741                                  BLT     16$
000514  013766  000000G 000016                  MOV     L$UNIT,16(SP)                   ;                               5369
000522  005004                                  CLR     R4                              ; UNIT
```

```
000524  000137  003664'                    JMP     41$
000530  010400                      19$:    MOV     R4,R0                   ; UNIT,*                5371
000532  006300                              ASL     R0
000534  012703  000022                      MOV     #22,R3
000540  060603                              ADD     SP,R3                   ; HWPT.ADDRESS,*
000542  060300                              ADD     R3,R0
000544  005710                              TST     (R0)
000546  001002                              BNE     20$
000550  000137  003662'                    JMP     40$
000554  105066  000006              20$:    CLRB    6(SP)                   ; FLAG                  5374
000560  011001                              MOV     (R0),R1                 ; *,HWPT.REF            5375
000562  005066  000002                      CLR     2(SP)                   ; CTLR                  5377
000566  016646  000002              21$:    MOV     2(SP),-(SP)             ; CTLR,*                5379
000572  012746  000126                      MOV     #126,-(SP)
000576  004737  000000G                     JSR     PC,BL$MUL
000602  022626                              CMP     (SP)+,(SP)+
000604  026011  000000G                     CMP     CST(R0),(R1)            ; *,HWPT.REF
000610  001402                              BEQ     22$
000612  000137  003104'                    JMP     28$
000616  012766  000001  000014    22$:    MOV     #1,14(SP)               ;                       5416
000624  112766  000001  000006            MOVB    #1,6(SP)                ; *,FLAG
000632  012705  000006                      MOV     #6,R5                   ;                       5382
000636  060105                              ADD     R1,R5                   ; HWPT.REF,*
000640  111546                              MOVB    (R5),-(SP)
000642  042716  177760                      BIC     #177760,(SP)
000646  005000                              CLR     R0
000650  156600  000012                      BISB    12(SP),R0               ; SMALLEST.DRIVE,*
000654  160016                              SUB     R0,(SP)
000656  012746  000012                      MOV     #12,-(SP)
000662  004737  000000G                     JSR     PC,BL$MUL
000666  010066  000010                      MOV     R0,10(SP)
000672  005726                              TST     (SP)+
000674  016616  000004                      MOV     4(SP),(SP)              ; CTLR,*                5383
000700  012746  000053                      MOV     #53,-(SP)
000704  004737  000000G                     JSR     PC,BL$MUL
000710  010003                              MOV     R0,R3
000712  022626                              CMP     (SP)+,(SP)+
000714  066600  000004                      ADD     4(SP),R0
000720  006300                              ASL     R0
000722  032760  040000  000006G           BIT     #40000,CST+6(R0)
000730  001140                              BNE     27$
000732  016602  000004                      MOV     4(SP),R2                ; *,TEMP                5386
000736  062702  000003                      ADD     #3,R2                   ; *,TEMP
000742  010300                              MOV     R3,R0                   ;                       5387
000744  060200                              ADD     R2,R0                   ; TEMP,*
000746  006300                              ASL     R0
000750  062700  000000G                     ADD     #CST,R0
000754  011510                              MOV     (R5),(R0)
000756  010446                              MOV     R4,-(SP)                ; UNIT,*                5389
000760  000316                              SWAB    (SP)
000762  042716  170377                      BIC     #170377,(SP)
000766  042710  007400                      BIC     #7400,(R0)
000772  052610                              BIS     (SP)+,(R0)
```

```
000774  042710  010000                          BIC     #10000,(R0)                     ;                               5390
001000  052710  040000                          BIS     #40000,(R0)                     ;                               5391
001004  105715                                  TSTB    (R5)                            ;                               5393
001006  100003                                  BPL     23$
001010  012761  177777  000016                  MOV     #-1,16(R1)                      ; *,*(HWPT.REF)                 5394
001016  010300                          23$:    MOV     R3,R0                           ;                               5396
001020  060200                                  ADD     R2,R0                           ; TEMP,*
001022  006300                                  ASL     R0
001024  016160  000010  000002G                 MOV     10(R1),CST+2(R0)                ; *(HWPT.REF),*
001032  010300                                  MOV     R3,R0                           ;                               5398
001034  060200                                  ADD     R2,R0                           ; TEMP,*
001036  006300                                  ASL     R0
001040  016160  000012  000004G                 MOV     12(R1),CST+4(R0)                ; *(HWPT.REF),*
001046  010300                                  MOV     R3,R0                           ;                               5400
001050  060200                                  ADD     R2,R0                           ; TEMP,*
001052  006300                                  ASL     R0
001054  016160  000014  000006G                 MOV     14(R1),CST+6(R0)                ; *(HWPT.REF),*
001062  010300                                  MOV     R3,R0                           ;                               5402
001064  060200                                  ADD     R2,R0                           ; TEMP,*
001066  006300                                  ASL     R0
001070  016160  000016  000010G                 MOV     16(R1),CST+10(R0)               ; *(HWPT.REF),*
001076  010300                                  MOV     R3,R0                           ;                               5405
001100  060200                                  ADD     R2,R0                           ; TEMP,*
001102  006300                                  ASL     R0
001104  011660  000012G                         MOV     (SP),CST+12(R0)                 ; BLANKS,*
001110  010300                                  MOV     R3,R0                           ;                               5406
001112  060200                                  ADD     R2,R0                           ; TEMP,*
001114  006300                                  ASL     R0
001116  011660  000014G                         MOV     (SP),CST+14(R0)                 ; BLANKS,*
001122  010300                                  MOV     R3,R0                           ;                               5409
001124  060200                                  ADD     R2,R0                           ; TEMP,*
001126  006300                                  ASL     R0
001130  062700  000020G                         ADD     #CST+20,R0
001134  105010                                  CLRB    (R0)
001136  111546                                  MOVB    (R5),-(SP)                      ;                               5411
001140  005046                                  CLR     -(SP)
001142  032766  000040  000002                  BIT     #40,2(SP)
001150  001401                                  BEQ     24$
001152  005216                                  INC     (SP)
001154  005116                          24$:    COM     (SP)
001156  011646                                  MOV     (SP),-(SP)
001160  042710  100000                          BIC     #100000,(R0)
001164  006026                                  ROR     (SP)+
001166  103002                                  BCC     25$
001170  052710  100000                          BIS     #100000,(R0)
001174  005726                          25$:    TST     (SP)+
001176  111516                                  MOVB    (R5),(SP)                       ;                               5413
001200  042710  010000                          BIC     #10000,(R0)
001204  032726  000100                          BIT     #100,(SP)+
001210  001402                                  BEQ     26$
001212  052710  010000                          BIS     #10000,(R0)
001216  010300                          26$:    MOV     R3,R0                           ;                               5415
001220  060200                                  ADD     R2,R0                           ; TEMP,*
```

```
001222  006300                          ASL     R0
001224  005060  000022G                 CLR     CST+22(R0)
001230  000430                          BR      29$                             ;                               5385
001232  011146                  27$:    MOV     (R1),-(SP)                      ; HWPT.REF,*                    5421
001234  111546                          MOVB    (R5),-(SP)
001236  042716  177760                  BIC     #177760,(SP)
001242  012746  000000G                 MOV     #CER.01,-(SP)
001246  012746  000003                  MOV     #3,-(SP)
001252  010600                          MOV     SP,R0                           ; SP,*
001254  104417                          TRAP    17
001256  062706  000010                  ADD     #10,SP
001262  112764  000001  000000G         MOVB    #1,DUR(R4)                      ; *,*(UNIT)                     5423
001270  010400                          MOV     R4,R0                           ; UNIT,*                        5424
001272  104451                          TRAP    51
001274  000406                          BR      29$                             ;                               5420
001276  005266  000002          28$:    INC     2(SP)                           ; CTLR                          5377
001302  000243                          .WORD CLV!CLC

001304  003002                          BGT     29$
001306  000137  002374'                 JMP     21$
001312  105766  000006          29$:    TSTB    6(SP)                           ; FLAG                          5429
001316  001402                          BEQ     30$
001320  000137  003662'                 JMP     40$
001324  005066  000014          30$:    CLR     14(SP)                          ; CTLR                          5433
001330  016646  000014          31$:    MOV     14(SP),-(SP)                    ; CTLR,*                        5435
001334  012746  000126                  MOV     #126,-(SP)
001340  004737  000000G                 JSR     PC,BL$MUL
001344  022626                          CMP     (SP)+,(SP)+
001346  005760  000000G                 TST     CST(R0)
001352  001402                          BEQ     32$
001354  000137  003602'                 JMP     37$
001360  011160  000000G          32$:    MOV     (R1),CST(R0)                    ; HWPT.REF,*                    5438
001364  016103  000002                  MOV     2(R1),R3                        ; *(HWPT.REF),*                 5439
001370  042703  177000                  BIC     #177000,R3
001374  042760  000777  000002G         BIC     #777,CST+2(R0)
001402  050360  000002G                 BIS     R3,CST+2(R0)
001406  116160  000004  000004G         MOVB    4(R1),CST+4(R0)                 ; *(HWPT.REF),*                 5440
001414  012705  000006                  MOV     #6,R5                           ;                               5441
001420  060105                          ADD     R1,R5                           ; HWPT.REF,*
001422  111546                          MOVB    (R5),-(SP)
001424  042716  177760                  BIC     #177760,(SP)
001430  005000                          CLR     R0
001432  156600  000012                  BISB    12(SP),R0                       ; SMALLEST.DRIVE,*
001436  160016                          SUB     R0,(SP)
001440  012746  000012                  MOV     #12,-(SP)
001444  004737  000000G                 JSR     PC,BL$MUL
001450  005726                          TST     (SP)+
001452  010002                          MOV     R0,R2                           ; *,TEMP
001454  062702  000003                  ADD     #3,R2                           ; *,TEMP
001460  016616  000016                  MOV     16(SP),(SP)                     ; CTLR,*                        5442
001464  012746  000053                  MOV     #53,-(SP)
001470  004737  000000G                 JSR     PC,BL$MUL
001474  010003                          MOV     R0,R3
```

```
001476  005726                          TST     (SP)+
001500  060200                          ADD     R2,R0                   ; TEMP,*
001502  006300                          ASL     R0
001504  062700  000000G                 ADD     #CST,R0
001510  011510                          MOV     (R5),(R0)
001512  010416                          MOV     R4,(SP)                 ; UNIT,*                5444
001514  000316                          SWAB    (SP)
001516  042716  170377                  BIC     #170377,(SP)
001522  042710  007400                  BIC     #7400,(R0)
001526  052610                          BIS     (SP)+,(R0)
001530  042710  010000                  BIC     #10000,(R0)             ;                       5445
001534  052710  040000                  BIS     #40000,(R0)             ;                       5446
001540  105715                          TSTB    (R5)                    ;                       5448
001542  100003                          BPL     33$
001544  012761  177777  000016          MOV     #-1,16(R1)              ; *,*(HWPT.REF)         5449
001552  010300                  33$:    MOV     R3,R0                   ;                       5451
001554  060200                          ADD     R2,R0                   ; TEMP,*
001556  006300                          ASL     R0
001560  016160  000010  000002G         MOV     10(R1),CST+2(R0)        ; *(HWPT.REF),*
001566  010300                          MOV     R3,R0                   ;                       5453
001570  060200                          ADD     R2,R0                   ; TEMP,*
001572  006300                          ASL     R0
001574  016160  000012  000004G         MOV     12(R1),CST+4(R0)        ; *(HWPT.REF),*
001602  010300                          MOV     R3,R0                   ;                       5455
001604  060200                          ADD     R2,R0                   ; TEMP,*
001606  006300                          ASL     R0
001610  016160  000014  000006G         MOV     14(R1),CST+6(R0)        ; *(HWPT.REF),*
001616  010300                          MOV     R3,R0                   ;                       5457
001620  060200                          ADD     R2,R0                   ; TEMP,*
001622  006300                          ASL     R0
001624  016160  000016  000010G         MOV     16(R1),CST+10(R0)       ; *(HWFT.REF),*
001632  010300                          MOV     R3,R0                   ;                       5460
001634  060200                          ADD     R2,R0                   ; TEMP,*
001636  006300                          ASL     R0
001640  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,*
001644  010300                          MOV     R3,R0                   ;                       5461
001646  060200                          ADD     R2,R0                   ; TEMP,*
001650  006300                          ASL     R0
001652  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,*
001656  010300                          MOV     R3,R0                   ;                       5464
001660  060200                          ADD     R2,R0                   ; TEMP,*
001662  006300                          ASL     R0
001664  062700  000020G                 ADD     #CST+20,R0
001670  105010                          CLRB    (R0)
001672  111546                          MOVB    (R5),-(SP)              ;                       5466
001674  005046                          CLR     -(SP)
001676  032766  000040  000002          BIT     #40,2(SP)
001704  001401                          BEQ     34$
001706  005216                          INC     (SP)
001710  005116                  34$:    COM     (SP)
001712  011646                          MOV     (SP),-(SP)
001714  042710  100000                  BIC     #100000,(R0)
001720  006026                          ROR     (SP)+
```

```
001722  103002                          BCC     35$
001724  052710  100000                  BIS     #100000,(R0)
001730  005726                  35$:    TST     (SP)+
001732  111516                          MOVB    (R5),(SP)               ;                       5467
001734  042710  010000                  BIC     #10000,(R0)
001740  032726  000100                  BIT     #100,(SP)+
001744  001402                          BEQ     36$
001746  052710  010000                  BIS     #10000,(R0)
001752  010300                  36$:    MOV     R3,R0                   ;                       5469
001754  060200                          ADD     R2,R0                   ; TEMP,*
001756  006300                          ASL     R0
001760  005060  000022G                 CLR     CST+22(R0)
001764  112766  000001  000006          MOVB    #1,6(SP)                ; *,FLAG                5470
001772  000410                          BR      39$                     ;                       5437
001774  005266  000014          37$:    INC     14(SP)                  ; CTLR                  5433
002000  000243                          .WORD CLV!CLC

002002  003002                          BGT     38$
002004  000137  003136'                JMP     31$
002010  105766  000006          38$:    TSTB    6(SP)                   ; FLAG                  5474
002014  001017                  39$:    BNE     40$
002016  012746  000001                  MOV     #1,-(SP)                ;                       5477
002022  012746  000000G                 MOV     #CER.02,-(SP)
002026  012746  000002                  MOV     #2,-(SP)
002032  010600                          MOV     SP,R0                   ; SP,*
002034  104417                          TRAP    17
002036  112764  000001  000000G         MOVB    #1,DUR(R4)              ; *,*(UNIT)             5478
002044  010400                          MOV     R4,R0                   ; UNIT,*                5479
002046  104451                          TRAP    51
002050  062706  000006                  ADD     #6,SP                   ;                       5476
002054  005204                  40$:    INC     R4                      ; UNIT                  5369
002056  020466  000016          41$:    CMP     R4,16(SP)               ; UNIT,*
002062  002002                          BGE     42$
002064  000137  002336'                JMP     19$
002070  123727  000000G 000005  42$:    CMPB    ENTRY.REASON,#5         ;                       5492
002076  001051                          BNE     48$
002100  013703  000000G         43$:    MOV     L$UNIT,R3               ;                       5496
002104  005004                          CLR     R4                      ; UNIT
002106  000404                          BR      45$
002110  010400                  44$:    MOV     R4,R0                   ; UNIT,*                5497
002112  104442                          TRAP    42
002114  010001                          MOV     R0,R1                   ; *,HWPT.REF
002116  005204                          INC     R4                      ; UNIT                  5496
002120  020403                  45$:    CMP     R4,R3                   ; UNIT,*
002122  002772                          BLT     44$
002124  005003                          CLR     R3                      ; CTLR                  5499
002126  010346                  46$:    MOV     R3,-(SP)                ; CTLR,*                5501
002130  012746  000126                  MOV     #126,-(SP)
002134  004737  000000G                 JSR     PC,BL$MUL
002140  105060  000005G                 CLRB    CST+5(R0)
002144  010316                          MOV     R3,(SP)                 ; CTLR,*                5504
002146  012746  000053                  MOV     #53,-(SP)
002152  004737  000000G                 JSR     PC,BL$MUL
```

```
002156  012701  000003                  MOV     #3,R1                   ; *,OFFSET                      5503
002162  010002                  47$:     MOV     R0,R2                   ;                               5504
002164  060102                           ADD     R1,R2                   ; OFFSET,*
002166  006302                           ASL     R2
002170  042762  020000  000000G          BIC     #20000,CST(R2)
002176  062701  000012                   ADD     #12,R1                  ; *,OFFSET                      5503
002202  020127  000041                   CMP     R1,#41                  ; OFFSET,*
002206  003765                           BLE     47$
002210  062706  000006                   ADD     #6,SP                   ;                               5500
002214  005203                           INC     R3                      ; CTLR                          5499
002216  000243                           .WORD CLV!CLC

002220  003742                           BLE     46$
002222  123727  000000G 000001  48$:     CMPB    ENTRY.REASON,#1         ;                               5510
002230  001017                           BNE     51$
002232  005037  000000G                  CLR     CTLR.CNT                ;                               5513
002236  005000                           CLR     R0                      ; CTLR                          5515
002240  005760  000000G          49$:     TST     CST(R0)                 ; *(CTLR)                       5517
002244  001402                           BEQ     50$
002246  005237  000000G                  INC     CTLR.CNT                ;                               5519
002252  062700  000126          50$:     ADD     #126,R0                 ; *,CTLR                        5515
002256  000243                           .WORD CLV!CLC

002260  003767                           BLE     49$
002262  104431                           TRAP    31                      ;                               5521
002264  010037  000000G                  MOV     R0,FREE.MEM.ADDR
002270  005001                  51$:     CLR     R1                      ; UNITS                         5529
002272  005003                  52$:     CLR     R3                      ; COUNT                         5530
002274  010300                  53$:     MOV     R3,R0                   ; COUNT,*                       5531
002276  060100                           ADD     R1,R0                   ; UNITS,*
002300  006300                           ASL     R0
002302  005060  000000G                  CLR     TALLY(R0)
002306  005203                           INC     R3                      ; COUNT                         5530
002310  020327  000006                   CMP     R3,#6                   ; COUNT,*
002314  003767                           BLE     53$
002316  062701  000033                   ADD     #33,R1                  ; *,UNITS                       5529
002322  020127  000121                   CMP     R1,#121                 ; UNITS,*
002326  003761                           BLE     52$
002330  032766  000001  000012          BIT     #1,12(SP)               ; *,CLEAR.TABLES                5533
002336  001436                           BEQ     57$
002340  005001                           CLR     R1                      ; UNITS                         5535
002342  012703  000007          54$:     MOV     #7,R3                   ; *,COUNT                       5536
002346  010300                  55$:     MOV     R3,R0                   ; COUNT,*                       5537
002350  060100                           ADD     R1,R0                   ; UNITS,*
002352  006300                           ASL     R0
002354  005060  000000G                  CLR     TALLY(R0)
002360  005203                           INC     R3                      ; COUNT                         5536
002362  020327  000032                   CMP     R3,#32                  ; COUNT,*
002366  003767                           BLE     55$
002370  062701  000033                   ADD     #33,R1                  ; *,UNITS                       5535
002374  020127  000121                   CMP     R1,#121                 ; UNITS,*
002400  003760                           BLE     54$
002402  032766  000001  000012          BIT     #1,12(SP)               ; *,CLEAR.TABLES                5539
```

```
002410  001411                                  BEQ     57$
002412  005000                                  CLR     R0                      ; CTLR                          5541
002414  105060  000000G                 56$:    CLRB    C.ERR.TBL(R0)           ; *(CTLR)                       5543
002420  105060  000001G                         CLRP    C.ERR.TBL+1(R0)         ; *(CTLR)                       5544
002424  062700  000002                          ADD     #2,R0                   ; *,CTLR                        5541
002430  000243                                  .WORD CLV!CLC

002432  003770                                  BLE     56$
002434  005000                          57$:    CLR     R0                      ; CTLR                          5551
002436  105060  000000G                 58$:    CLRB    CTR(R0)                 ; *(CTLR)                       5552
002442  005200                                  INC     R0                      ; CTLR                          5551
002444  000243                                  .WORD CLV!CLC

002446  003773                                  BLE     58$
002450  005000                                  CLR     R0                      ; COUNT                         5554
002452  112760  000377  000000G         59$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)                    5555
002460  005200                                  INC     R0                      ; COUNT                         5554
002462  020027  000007                          CMP     R0,#7                   ; COUNT,*
002466  003771                                  BLE     59$
002470  132737  000001  000000G               BITB    #1,CLK.PRESENT          ;                               5557
002476  001402                                  BEQ     60$
002500  005037  177546                          CLR     @#177546                ;                               5559
002504  005037  000000G                 60$:    CLR     IODQ.OUT                ;                               5561
002510  005037  000000G                         CLR     IODQ.IN
002514  005037  000000G                         CLR     CRN.HIGH                ;                               5562
002520  005037  000000G                         CLR     CRN.LOW
002524  005000                                  CLR     R0                      ;                               5563
002526  104441                                  TRAP    41
002530  062706  000032                          ADD     #32,SP                  ;                               5252
002534  000207                                  RTS     PC

; Routine Size:  687 words,     Routine Base:  $CODE$ + 1606
; Maximum stack depth per invocation:  25 words




                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001606'         L$INIT::JSR     PC,LINIT                ;                               5563
000004  104411                          TRAP    11
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4344
; Maximum stack depth per invocation:  2 words
```

```
;    5566  1     %sbttl 'AUTODROP SECTION'
;    5567  1
;    5568  1     !+
;    5569  1     ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;    5570  1     ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;    5571  1     ! SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;    5572  1     ! DROPPED FROM TESTING.
;    5573  1     !-
;    5574  1
;    5575  2     BGNAUTO;
;    5576  2
;    5577  2     !if BIT_TST (SWP_FLAGS, SWF_TRC)
;    5578  2     !then
;    5579  2     !   PRINTF (DBM3);
;    5580  2
;    5581  2     return;
;    5582  2
;    5583  1     ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                              ;       5565

; Routine Size:  1 word,        Routine Base:  $CODE$ + 4354
; Maximum stack depth per invocation:  0 words




                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  004354'         L$AUTO::JSR     PC,LAUTO                        ;       5581
000004  104461                          TRAP    61
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4356
; Maximum stack depth per invocation:  2 words
```

```
;    5584  1     %sbttl 'CLEANUP CODING SECTION'
;    5585  1
;    5586  1     !+
;    5587  1     ! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;    5588  1     ! AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;    5589  1     !-
;    5590  1
;    5591  2     BGNCLN;
;    5592  2
;    5593  2     DORPT;
;    5594  2
;    5595  2     !CLRVEC (O_TVEC);                                   ! RETURN ODT TRAP TO DIAGNOSTIC SUPERVISER
;    5596  2
;    5597  2     if .CLK_PRESENT
;    5598  2     then
;    5599  3         begin
;    5600  3         LINE_CLOCK = 0;                                 ! STOP THE LINE-CLOCK
;    5601  3     !   CLRVEC (%o'100');                               ! RETURN LINE-CLOCK'S VECTOR TO SUPERVISOR
;    5602  2         end;
;    5603  2
;    5604  2     incr CTLR from 0 to (MAX_CTLR - 1) do               ! FOR EACH CONTROLLER
;    5605  2
;    5606  2         if (RDRX_ADDR = .CST [.CTLR, IP_ADDR]) neqa 0   ! IF CONTROLLER EXISTS
;    5607  2         then
;    5608  3             begin
;    5609  3
;    5610  3             if .ADDR_VECT_OK
;    5611  3             then
;    5612  4                 begin
;    5613  4
;    5614  4                 if .DCT [.CTLR, STAT] eql ONLINE        ! IF CONTROLLER ALIVE
;    5615  4                 then
;    5616  4
;    5617  4                     incr COUNT from 1 to 10000 do
;    5618  5                         begin
;    5619  5                         DELAY (1);
;    5620  5
;    5621  5                         if .DCT [.CTLR, CRING_CNT] eql 0    ! WAIT TILL OUTSTANDING COMMANDS FINISHED
;    5622  5                         then
;    5623  5                             exitloop;
;    5624  5
;    5625  4                         end;
;    5626  4
;    5627  4                 WRT_RDRX (RCIP, RC_ALL, ALL_ONES);      ! WRITE IP TO STOP DEVICE
;    5628  3                 end;
;    5629  3
;    5630  3             CLRVEC (.CST[.CTLR, VEC_ADDR]);             ! RETURN CONTROLLER'S TRAP VECTOR TO SUPERVISOR
;    5631  2             end;
;    5632  2
;    5633  1     ENDCLN;

                                    .SBTTL  LCLEAN CLEANUP CODING SECTION
```

```
000000  004137  000000G                 LCLEAN: JSR     R1,$SAVE5                       ;                               5583
000004  005746                                  TST     -(SP)
000006  104424                                  TRAP    24                              ;                               5591
000010  132737  000001  000000G                BITB    #1,CLK.PRESENT                  ;                               5597
000016  001402                                  BEQ     1$
000020  005037  177546                          CLR     @#177546                        ;                               5600
000024  005005                          1$:     CLR     R5                              ; CTLR                          5604
000026  010546                          2$:     MOV     R5,-(SP)                        ; CTLR,*                        5606
000030  012746  000126                          MOV     #126,-(SP)
000034  004737  000000G                         JSR     PC,BL$MUL
000040  010001                                  MOV     R0,R1
000042  022626                                  CMP     (SP)+,(SP)+
000044  016137  000000G 000000G                 MOV     CST(R1),RDRX.ADDR
000052  001450                                  BEQ     10$
000054  132737  000001  000000G                BITB    #1,ADDR.VECT.OK                 ;                               5610
000062  001437                                  BEQ     9$
000064  010546                                  MOV     R5,-(SP)                        ; CTLR,*                        5614
000066  012746  000022                          MOV     #22,-(SP)
000072  004737  000000G                         JSR     PC,BL$MUL
000076  022626                                  CMP     (SP)+,(SP)+
000100  005760  000000G                         TST     DCT(R0)
000104  100022                                  BPL     8$
000106  012704  023420                          MOV     #23420,R4                       ; *,COUNT                       5617
000112  012703  000001                  3$:     MOV     #1,R3                           ; *,$$TMP2                      5619
000116  001410                          4$:     BEQ     7$
000120  013702  000000G                         MOV     L$DLY,R2                        ; *,$$TMP1
000124  001403                                  BEQ     6$
000126  005016                          5$:     CLR     (SP)                            ; $$TMP
000130  005302                                  DEC     R2                              ; $$TMP1
000132  001375                                  BNE     5$
000134  005303                          6$:     DEC     R3                              ; $$TMP2
000136  000767                                  BR      4$
000140  105760  000000G                 7$:     TSTB    DCT(R0)                         ;                               5621
000144  001402                                  BEQ     8$                              ;                               5623
000146  005304                                  DEC     R4                              ; COUNT                         5617
000150  001360                                  BNE     3$
000152  012700  177777                  8$:     MOV     #-1,R0                          ; *,RC.REG                      5627
000156  010077  000000G                         MOV     R0,@RDRX.ADDR                   ; RC.REG,*
000162  016100  000002G                 9$:     MOV     CST+2(R1),R0                    ;                               5630
000166  042700  177000                          BIC     #177000,R0
000172  104436                                  TRAP    36
000174  005205                          10$:    INC     R5                              ; CTLR                          5604
000176  000243                                  .WORD CLV!CLC

000200  003712                                  BLE     2$
000202  005726                                  TST     (SP)+                           ;                               5583
000204  000207                                  RTS     PC

; Routine Size:  67 words,      Routine Base:  $CODE$ + 4366
; Maximum stack depth per invocation:  10 words
```

```
                                        .SBTTL  L$CLEAN CLEANUP CODING SECTION

000000  004737  004366'            L$CLEAN::
                                        JSR     PC,LCLEAN                       ;                               5631
000004  104412                            TRAP    12
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 4574
; Maximum stack depth per invocation:  2 words
```

```
;    5634  1     %sbttl 'DROP UNIT SECTION'
;    5635  1
;    5636  1     !+
;    5637  1     ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    5638  1     ! TO NO LONGER BE TESTED.
;    5639  1     !-
;    5640  1
;    5641  2     BGNDU;
;    5642  2
;    5643  2     local
;    5644  2         UNIT : word,                                                    ! UNIT NUMBER
;    5645  2         PRINT : byte initial (byte (FALSE));                            ! NO PRINTING
;    5646  2
;    5647  2     label
;    5648  2         SEARCH;
;    5649  2
;    5650  3     begin
;    5651  3
;    5652  3     register
;    5653  3         INPUT = 0;                                                      ! R0 = UNIT NO.
;    5654  3
;    5655  3     UNIT = .INPUT;                                                      ! GET UNIT NUMBER
;    5656  2     end;                                                                ! UNDECLARE R0
;    5657  2
;    5658  3     if BIT_TST (SWP_FLAGS, SWF_TRC)
;    5659  2     then
;    5660  2         PRINTF (DBM5, .UNIT);
;    5661  2
;    5662  2     SEARCH :                                                            ! SEARCH BLOCK
;    5663  3     begin
;    5664  3
;    5665  3     incr CTLR from 0 to (MAX_CTLR - 1) do                               ! FOR EACH CNTR
;    5666  3
;    5667  3         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    5668  3
;    5669  3             if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and  ! IF UNIT MATCHES
;    5670  4                (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    5671  3             then
;    5672  4                 begin
;    5673  4
;    5674  4                 if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or  ! IF UNIT ALIVE
;    5675  4                    (.DUR [.UNIT] eql DU_ONLINE) or
;    5676  5                    (.DUR [.UNIT] eql DU_PROTECT)
;    5677  4                 then
;    5678  5                     begin
;    5679  5                     PRINT = TRUE;                                       ! O.K. TO PRINT
;    5680  5
;    5681  5                     if (.CST [.CTLR, U_CNT] gtru 0) and
;    5682  6                        (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    5683  5                     then
;    5684  5                         CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;  ! DECREMENT COUNT
;    5685  5
;    5686  5                     if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;   5687  6                    (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;   5688  5                 then
;   5689  5                     EOP_FLAG = TRUE;                                        ! ALL UNITS OFFLINE
;   5690  5
;   5691  5                 CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;           ! MARK UNIT OFFLINE
;   5692  4                 end;                                                        ! IF UNIT ALIVE
;   5693  4
;   5694  4             leave SEARCH;                                                   ! EXIT SEARCH BLOCK
;   5695  3             end;                                                            ! IF UNIT FOUND
;   5696  3
;   5697  2     end;
;   5698  2
;   5699  2     if .PRINT or                                                            ! IF OK TO PRINT
;   5700  2        (.DUR [.UNIT] eql DU_CONF) or
;   5701  2        (.DUR [.UNIT] eql DU_INIT) or
;   5702  2        (.DUR [.UNIT] eql DU_ONLINE) or
;   5703  3        (.DUR [.UNIT] eql DU_PROTECT)
;   5704  2     then
;   5705  3         begin
;   5706  3         PRINTF (DU_MSG, .UNIT);                                             ! "UNIT XX DROPPED"
;   5707  3         PRINTF (.DU_RSN [.DUR [.UNIT]]);                                    ! REASON
;   5708  2         end;
;   5709  2
;   5710  1     ENDDU;


                                        .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G          LDU:   JSR     R1,$SAVE5                   ;                   5633
000004  005746                          TST     -(SP)
000006  105046                          CLRB    -(SP)                       ; PRINT
000010  010001                          MOV     R0,R1                       ; INPUT,UNIT        5655
000012  032737  000001  000000G         BIT     #1,SWP.FLAGS                ;                   5658
000020  001411                          BEQ     1$
000022  010146                          MOV     R1,-(SP)                    ; UNIT,*            5660
000024  012746  000000G                 MOV     #DBM5,-(SP)
000030  012746  000002                  MOV     #2,-(SP)
000034  010600                          MOV     SP,R0                       ; SP,*
000036  104417                          TRAP    17
000040  062706  000006                  ADD     #6,SP
000044  005005                   1$:    CLR     R5                          ; CTLR              5665
000046  010546                   2$:    MOV     R5,-(SP)                    ; CTLR,*            5669
000050  012746  000053                  MOV     #53,-(SP)
000054  004737  000000G                 JSR     PC,BL$MUL
000060  010066  000006                  MOV     R0,6(SP)
000064  012703  000003                  MOV     #3,R3                       ; *,OFFSET          5667
000070  010300                   3$:    MOV     R3,R0                       ; OFFSET,*          5669
000072  066600  000006                  ADD     6(SP),R0
000076  006300                          ASL     R0
000100  012702  000000G                 MOV     #CST,R2
000104  060002                          ADD     R0,R2
000106  010104                          MOV     R1,R4                       ; UNIT,*
000110  011200                          MOV     (R2),R0
000112  000300                          SWAB    R0
```

```
000114  042700  177760                      BIC     #177760,R0
000120  020004                              CMP     R0,R4
000122  001055                              BNE     9$
000124  032712  040000                      BIT     #40000,(R2)             ;                       5670
000130  001452                              BEQ     9$
000132  005004                              CLR     R4                      ;                       5674
000134  032712  020000                      BIT     #20000,(R2)
000140  001402                              BEQ     4$
000142  005204                              INC     R4
000144  000410                              BR      5$
000146  126127  000000G 000007      4$:     CMPB    DUR(R1),#7              ; *(UNIT),*             5675
000154  001404                              BEQ     5$
000156  126127  000000G 000011              CMPB    DUR(R1),#11             ; *(UNIT),*             5676
000164  001032                              BNE     8$
000166  112766  000001  000004      5$:     MOVB    #1,4(SP)                ; *,PRINT               5679
000174  010516                              MOV     R5,(SP)                 ; CTLR,*                5681
000176  012746  000126                      MOV     #126,-(SP)
000202  004737  000000G                     JSR     PC,BL$MUL
000206  005726                              TST     (SP)+
000210  062700  000004G                     ADD     #CST+4,R0
000214  105760  000001                      TSTB    1(R0)
000220  001404                              BEQ     6$
000222  006004                              ROR     R4                      ;                       5682
000224  105660  000001                      SBCB    1(R0)                   ;                       5684
000230  001006                              BNE     7$                      ;                       5686
000232  032712  020000              6$:     BIT     #20000,(R2)             ;                       5687
000236  001403                              BEQ     7$
000240  112737  000001  000000G             MOVB    #1,EOP.FLAG             ;                       5689
000246  042712  020000              7$:     BIC     #20000,(R2)             ;                       5691
000252  022626                      8$:     CMP     (SP)+,(SP)+             ;                       5672
000254  000411                              BR      10$
000256  062703  000012              9$:     ADD     #12,R3                  ; *,OFFSET              5667
000262  020327  000041                      CMP     R3,#41                  ; OFFSET,*
000266  003700                              BLE     3$
000270  022626                              CMP     (SP)+,(SP)+
000272  005205                              INC     R5                      ; CTLR                  5665
000274  000243                              .WORD CLV!CLC

000276  003663                              BLE     2$
000300  032716  000001              10$:    BIT     #1,(SP)                 ; *,PRINT               5699
000304  001020                              BNE     11$
000306  126127  000000G 000001              CMPB    DUR(R1),#1              ; *(UNIT),*             5700
000314  001414                              BEQ     11$
000316  126127  000000G 000002              CMPB    DUR(R1),#2              ; *(UNIT),*             5701
000324  001410                              BEQ     11$
000326  126127  000000G 000007              CMPB    DUR(R1),#7              ; *(UNIT),*             5702
000334  001404                              BEQ     11$
000336  126127  000000G 000011              CMPB    DUR(R1),#11             ; *(UNIT),*             5703
000344  001024                              BNE     12$
000346  010146                      11$:    MOV     R1,-(SP)                ; UNIT,*                5706
000350  012746  000000G                     MOV     #DU.MSG,-(SP)
000354  012746  000002                      MOV     #2,-(SP)
000360  010600                              MOV     SP,R0                   ; SP,*
```

```
000362  104417                          TRAP    17
000364  116101  000000G                 MOVB    DUR(R1),R1              ; *(UNIT),*                     5707
000370  042701  177400                  BIC     #177400,R1
000374  006301                          ASL     R1
000376  016116  000000G                 MOV     DU.RSN(R1),(SP)
000402  012746  000001                  MOV     #1,-(SP)
000406  010600                          MOV     SP,R0                   ; SP,*
000410  104417                          TRAP    17
000412  062706  000010                  ADD     #10,SP                  ;                               5705
000416  022626                  12$:    CMP     (SP)+,(SP)+             ;                               5633
000420  000207                          RTS     PC

; Routine Size:  137 words,     Routine Base:  $CODE$ + 4604
; Maximum stack depth per invocation:  14 words



                                        .SBTTL  L$DU DROP UNIT SECTION
000000  004737  004604'          L$DU::  JSR     PC,LDU                  ;                               5708
000004  104453                          TRAP    53
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5226
; Maximum stack depth per invocation:  2 words
```

```
;    5711  1     %sbttl 'ADD UNIT SECTION'
;    5712  1
;    5713  1     !+
;    5714  1     ! THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    5715  1     !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    5716  1     !  TO THE TEST CYCLE.
;    5717  1     !-
;    5718  1
;    5719  2     BGNAU;
;    5720  2
;    5721  2     local
;    5722  2         STINDX : word,
;    5723  2         ENDIDX : word;
;    5724  2
;    5725  2     register
;    5726  2         UNIT = 0;                                   ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;    5727  2
;    5728  3     if BIT_TST (SWP_FLAGS, SWF_CST)
;    5729  2     then
;    5730  3         begin                                       ! IF CLEAR STAT. TABLES TRUE....
;    5731  3         STINDX = .UNIT * TALLY_LEN;                 ! ZERO OUT
;    5732  3         ENDIDX = .STINDX + TALLY_LEN - 1;           !    ADDED
;    5733  3
;    5734  3         incr COUNT from .STINDX to .ENDIDX do       !    UNIT'S
;    5735  3             TALLY [.COUNT] = 0;                     !    STATISTICS
;    5736  3
;    5737  2         end;
;    5738  2
;    5739  1     ENDAU;


                                .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G    LAU: JSR     R1,$SAVE2                  ;                          5710
000004  105737  000000G         TSTB    SWP.FLAGS                  ;                          5728
000010  100023                 BPL     3$
000012  010046                 MOV     R0,-(SP)                   ; UNIT,*                   5731
000014  012746  000033         MOV     #33,-(SP)
000020  004737  000000G        JSR     PC,BL$MUL
000024  010002                 MOV     R0,R2                      ; STINDX,ENDIDX            5732
000026  062702  000032         ADD     #32,R2                     ; *,ENDIDX
000032  010001                 MOV     R0,R1                      ; STINDX,COUNT             5734
000034  005301                 DEC     R1                         ; COUNT
000036  000404                 BR      2$
000040  010100             1$: MOV     R1,R0                      ; COUNT,*                  5735
000042  006300                 ASL     R0
000044  005060  000000G        CLR     TALLY(R0)
000050  005201             2$: INC     R1                         ; COUNT                    5734
000052  020102                 CMP     R1,R2                      ; COUNT,ENDIDX
000054  003771                 BLE     1$
000056  022626                 CMP     (SP)+,(SP)+                ;                          5730
000060  000207             3$: RTS     PC                         ;                          5710

; Routine Size:  25 words,    Routine Base:  $CODE$ + 5236
```

```
; Maximum stack depth per invocation:  6 words



                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  005236'          L$AU::  JSR     PC,LAU                  ;                5737
000004  104452                          TRAP    52
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 5320
; Maximum stack depth per invocation:  2 words
```

```
;   5740  1     %sbttl 'NON-EXISTENT MEMORY TRAP HANDLER'
;   5741  1
;   5742  1     !+
;   5743  1     !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
;   5744  1     !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON-EXISTENT
;   5745  1     !       MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RDRX
;   5746  1     !       REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
;   5747  1     !-
;   5748  1
;   5749  2     BGNSRV (NEX_TRAP);
;   5750  2
;   5751  2     NEX = TRUE;                                     ! NEX TRAP OCCURRED
;   5752  2
;   5753  1     ENDSRV;


                                        .SBTTL  NEX.TRAP NON-EXISTENT MEMORY TRAP HANDLER
000000  012737  000001  000000G   NEX.TRAP::
                                        MOV     #1,NEX                          ;       5751
000006  000002                          RTI                                     ;       5749

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5330
; Maximum stack depth per invocation:  0 words
```

```
;   5754  1     %sbttl 'TIME OF DAY'
;   5755  1
;   5756  1     !+
;   5757  1     !       THIS INTERRUPT SERVICE ROUTINE KEEPS TRACK OF THE TIME-OF-DAY
;   5758  1     !-
;   5759  1
;   5760  2     BGNSRV (TIME);
;   5761  2
;   5762  2     CLK_TICKS = .CLK_TICKS + 1;             ! INCREMENT CLOCK-TICKS
;   5763  2
;   5764  2     if .CLK_TICKS gequ 3600
;   5765  2     then
;   5766  3         begin
;   5767  3         MINUTES = .MINUTES + 1;             ! UPDATE MINUTE COUNT
;   5768  3         CLK_TICKS = 0;
;   5769  2         end;
;   5770  2
;   5771  2     if .MINUTES gequ 60
;   5772  2     then
;   5773  3         begin
;   5774  3         HOURS = .HOURS + 1;                 ! UPDATE HOUR COUNT
;   5775  3         MINUTES = 0;
;   5776  2         end;
;   5777  2
;   5778  2     if .HOURS gequ 24
;   5779  2     then
;   5780  2         HOURS = 0;                          ! RATIONALIZE HOURS
;   5781  2
;   5782  1     ENDSRV;


                                        .SBTTL  TIME TIME OF DAY
000000  005237  000000G                TIME::  INC     CLK.TICKS                       ;   5762
000004  023727  000000G 007020                 CMP     CLK.TICKS,#7020                 ;   5764
000012  103404                                 BLO     1$
000014  105237  000000G                        INCB    MINUTES                         ;   5767
000020  005037  000000G                        CLR     CLK.TICKS                       ;   5768
000024  123727  000000G 000074        1$:     CMPB    MINUTES,#74                     ;   5771
000032  103404                                 BLO     2$
000034  105237  000000G                        INCB    HOURS                           ;   5774
000040  105037  000000G                        CLRB    MINUTES                         ;   5775
000044  123727  000000G 000030        2$:     CMPB    HOURS,#30                       ;   5778
000052  103402                                 BLO     3$
000054  105037  000000G                        CLRB    HOURS                           ;   5780
000060  000002                        3$:     RTI                                     ;   5760

; Routine Size:  25 words,      Routine Base:  $CODE$ + 5340
; Maximum stack depth per invocation:  0 words
```

```
;    5783  1     %sbttl 'GLOBAL ROUTINES'
;    5784  1
;    5785  1     global routine SET_CPAR (CTLR) : novalue =
;    5786  1
;    5787  1     !+
;    5788  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    5789  1     !       FOR THE GIVEN CONTROLLER NUMBER.
;    5790  1     !
;    5791  1     !       INPUTS:
;    5792  1     !               CTLR - CONTROLLER NUMBER
;    5793  1     !
;    5794  1     !       IMPLICIT OUTPUTS:
;    5795  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5796  1     !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;    5797  1     !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;    5798  1     !               RDRX_ADDR - ADDRESS OF CONTROLLER'S IP REGISTER
;    5799  1     !-
;    5800  1
;    5801  2         begin
;    5802  2         CCTLR = .CTLR;                          ! SET CURRENT CONTROLLER NUMBER
;    5803  2         CST_ADDR = CST + (.CTLR * CST_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S CST
;    5804  2         DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S DCT
;    5805  2         RDRX_ADDR = .CST_ADDR [IP_ADDR];            ! GET CONTROLLER'S DEVICE ADDRESS
;    5806  1         end;


                                        .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                 SET.CPAR::
                                        MOV     R1,-(SP)                        ;                       5785
000002  016601  000004                  MOV     4(SP),R1                        ; CTLR,*                5802
000006  010137  000000G                 MOV     R1,CCTLR
000012  010146                          MOV     R1,-(SP)                        ;                       5803
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  062700  000000G                 ADD     #CST,R0
000030  010037  000000G                 MOV     R0,CST.ADDR
000034  010116                          MOV     R1,(SP)                         ;                       5804
000036  012746  000022                  MOV     #22,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  062700  000000G                 ADD     #DCT,R0
000052  010037  000000G                 MOV     R0,DCT.ADDR
000056  017737  000000G 000000G         MOV     @CST.ADDR,RDRX.ADDR             ;                       5805
000064  062706  000006                  ADD     #6,SP                           ;                       5801
000070  012601                          MOV     (SP)+,R1                        ;                       5785
000072  000207                          RTS     PC

; Routine Size:  30 words,      Routine Base:  $CODE$ + 5422
; Maximum stack depth per invocation:  5 words
```

```
;    5807  1     global routine SET_UPAR (OFFSET) : novalue =
;    5808  1     !+
;    5809  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
;    5810  1     !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
;    5811  1     !
;    5812  1     !       INPUTS:
;    5813  1     !               OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
;    5814  1     !                        DESCRIBES A UNIT
;    5815  1     !
;    5816  1     !       IMPLICIT INPUTS:
;    5817  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    5818  1     !
;    5819  1     !       IMPLICIT OUTPUTS:
;    5820  1     !               CUOFF - CURRENT UNIT'S CST OFFSET
;    5821  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    5822  1     !               L$LUN - CURRENT UNIT NUMBER (DRS UNIT NUMBER)
;    5823  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5824  1     !-
;    5825  2         begin
;    5826  2         CUOFF = .OFFSET;
;    5827  2         CDISK = .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM];
;    5828  2         L$LUN = .CST_ADDR [.OFFSET + OF_DATA, D_UNIT];
;    5829  2         T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
;    5830  1         end;


                                        .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                 SET.UPAR::
                                        MOV     R1,-(SP)                        ;                                       5807
000002  016637  000004  000000G        MOV     4(SP),CUOFF                     ; OFFSET,*                              5826
000010  016600  000004                  MOV     4(SP),R0                        ; CUOFF,*                               5827
000014  006300                          ASL     R0
000016  063700  000000G                 ADD     CST.ADDR,R0
000022  111037  000000G                 MOVB    (R0),CDISK
000026  042737  177760  000000G        BIC     #177760,CDISK
000034  011001                          MOV     (R0),R1                         ;                                       5828
000036  000301                          SWAB    R1
000040  042701  177760                  BIC     #177760,R1
000044  010137  000000G                 MOV     R1,L$LUN
000050  010146                          MOV     R1,-(SP)                        ; L$LUN,*                               5829
000052  012746  000066                  MOV     #66,-(SP)
000056  004737  000000G                 JSR     PC,BL$MUL
000062  062700  000000G                 ADD     #TALLY,R0
000066  010037  000000G                 MOV     R0,T.ADDR
000072  022626                          CMP     (SP)+,(SP)+                     ;                                       5825
000074  012601                          MOV     (SP)+,R1                        ;                                       5807
000076  000207                          RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 5516
; Maximum stack depth per invocation:  4 words
```

```
;    5831  1
;    5832  1      global routine GET_PKT (CTLR) =
;    5833  1
;    5834  1      !+
;    5835  1      !       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
;    5836  1      !       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    5837  1      !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
;    5838  1      !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    5839  1      !
;    5840  1      !       INPUTS:
;    5841  1      !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    5842  1      !-
;    5843  1
;    5844  2          begin
;    5845  2
;    5846  2          local
;    5847  2              index : signed word initial (-1),
;    5848  2              RING_ADDR : word,
;    5849  2              PACKET_OWNED : byte,
;    5850  2              NEXT_PACKET : byte;
;    5851  2
;    5852  2
;    5853  2          NEXT_PACKET = .NEXT_PKT_USE;                    ! NEXT PACKET TO TRY
;    5854  2
;    5855  2          incr COUNT from 0 to (PKT_CNT - 1) do           ! FOR EACH ENTRY IN ALLOCATION TABLE
;    5856  3              begin
;    5857  3              PACKET_OWNED = FALSE;
;    5858  3
;    5859  3              if .PKT_USE [.NEXT_PACKET] lss 0            ! IF ENTRY INDICATES FREE PACKET
;    5860  3
;    5861  3              then
;    5862  4                  begin
;    5863  4                  RING_ADDR = .DCT_ADDR [RR_BEG];         ! FIRST RESPONSE PACKET'S ADDRESS
;    5864  4
;    5865  4                  incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS
;    5866  4
;    5867  4                      if (..RING_ADDR eqla .MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
;    5868  5                         (((.RING_ADDR + 2) and ED_OWN) eql ED_OWN)
;    5869  5
;    5870  4                      then
;    5871  5                          begin                           ! CHECK ADDRESS AND OWNERSHIP
;    5872  5                          PACKET_OWNED = TRUE;            ! PACKET OWNED BY CONTROLLER
;    5873  5                          exitloop;
;    5874  5                          end
;    5875  4                      else
;    5876  4                          RING_ADDR = .RING_ADDR + 4;     ! ADDRESS OF NEXT PACKET IN RING
;    5877  4
;    5878  4                  if not .PACKET_OWNED                    ! IF NOT ALREADY USED
;    5879  4
;    5880  4                  then
;    5881  5                      begin
;    5882  5                      PKT_USE [.NEXT_PACKET] = .CTLR;     ! ALLOCATE PACKET TO CONTROLLER
;    5883  5                      index = .NEXT_PACKET;
```

```
;   5884  5
;   5885  5                 incr J from 2 to (PKT_LEN - 1) do        ! ZERO OUT PACKET
;   5886  5                     MSCP_PKT [.NEXT_PACKET, .J, 0, 16, 0] = 0;
;   5887  5
;   5888  5                 exitloop;                                ! DONE
;   5889  5
;   5890  4                 end;
;   5891  4
;   5892  3             end;
;   5893  3
;   5894  3         NEXT_PACKET = .NEXT_PACKET + 1;                  ! TRY NEXT PACKET IN RING
;   5895  3
;   5896  3         if .NEXT_PACKET gequ PKT_CNT
;   5897  3
;   5898  3         then
;   5899  3             NEXT_PACKET = 0;                             ! IF BEYOND ALL PACKETS, START AT THE TOP
;   5900  3
;   5901  2         end;
;   5902  2
;   5903  2     if (.index geq 0) and                                ! IF PACKET FOUND
;   5904  3        (.PKT_USE [.index] geq 0)
;   5905  2     then
;   5906  2
;   5907  3         begin
;   5908  3         MSCP_PKT [.index, MSGLEN] = SZ_GEN;              ! PACKET SIZE - ONLY ONLINE AND SCC CHANGE IT
;   5909  3         MSCP_PKT [.index, CREDITS] = 1;                  ! CREDIT SIZE
;   5910  3         NEXT_PKT_USE = .NEXT_PACKET + 1;                 ! NEXT PACKET TO ALLOCATE
;   5911  3
;   5912  3         if.NEXT_PKT_USE gequ PKT_CNT
;   5913  3         then
;   5914  3             NEXT_PKT_USE = 0;                            ! CYCLE BACK TO BEGINNING IF AT END
;   5915  3
;   5916  2         end;
;   5917  2
;   5918  2     return .index;
;   5919  2
;   5920  1     end;


                                        .SBTTL  GET.PKT GLOBAL ROUTINES
000000  004137  000000G             GET.PKT::
                                        JSR     R1,$SAVE5                   ;                                   5832
000004  162706  000006                  SUB     #6,SP
000010  012704  177777                  MOV     #-1,R4                      ; *,INDEX                           5844
000014  113766  000000G 000004          MOVB    NEXT.PKT.USE,4(SP)          ; *,NEXT.PACKET                     5853
000022  012766  000014  000002          MOV     #14,2(SP)                   ; *,COUNT                           5855
000030  105016                      1$:     CLRB    (SP)                        ; PACKET.OWNED                      5857
000032  005001                          CLR     R1                          ;                                   5859
000034  156601  000004                  BISB    4(SP),R1                    ; NEXT.PACKET,*
000040  105761  000000G                 TSTB    PKT.USE(R1)
000044  002072                          BGE     7$
000046  013700  000000G                 MOV     DCT.ADDR,R0                 ;                                   5863
000052  016005  000004                  MOV     4(R0),R5                    ; *,RING.ADDR
```

```
000056  010146                          MOV     R1,-(SP)                        ;                               5867
000060  012746  000106                  MOV     #106,-(SP)
000064  004737  000000G                 JSR     PC,BL$MUL
000070  012702  000010                  MOV     #10,R2                          ; *,I                           5865
000074  021560  000000G         2$:     CMP     (R5),MSCP.PKT(R0)               ; RING.ADDR,*                   5867
000100  001014                          BNE     3$
000102  012703  000002                  MOV     #2,R3                           ;                               5868
000106  060503                          ADD     R5,R3                           ; RING.ADDR,*
000110  042703  077777                  BIC     #77777,R3
000114  020327  100000                  CMP     R3,#-100000
000120  001004                          BNE     3$
000122  112766  000001  000004         MOVB    #1,4(SP)                        ; *,PACKET.OWNED                5872
000130  000404                          BR      4$                              ;                               5871
000132  062705  000004         3$:     ADD     #4,R5                           ; *,RING.ADDR                   5876
000136  005302                          DEC     R2                              ; I                             5865
000140  001355                          BNE     2$
000142  032766  000001  000004         BIT     #1,4(SP)                        ; *,PACKET.OWNED                5878
000150  001027                          BNE     6$
000152  116661  000030  000000G         MOVB    30(SP),PKT.USE(R1)              ; CTLR,*                        5882
000160  010104                          MOV     R1,R4                           ; *,INDEX                       5883
000162  010116                          MOV     R1,(SP)                         ;                               5886
000164  012746  000043                  MOV     #43,-(SP)
000170  004737  000000G                 JSR     PC,BL$MUL
000174  005726                          TST     (SP)+
000176  012702  000002                  MOV     #2,R2                           ; *,J                           5885
000202  010003                 5$:     MOV     R0,R3                           ;                               5886
000204  060203                          ADD     R2,R3                           ; J,*
000206  006303                          ASL     R3
000210  005063  000000G                 CLR     MSCP.PKT(R3)
000214  005202                          INC     R2                              ; J                             5885
000216  020227  000042                  CMP     R2,#42                          ; J,*
000222  003767                          BLE     5$
000224  022626                          CMP     (SP)+,(SP)+                     ;                               5881
000226  000414                          BR      9$
000230  022626                 6$:     CMP     (SP)+,(SP)+                     ;                               5862
000232  105266  000004         7$:     INCB    4(SP)                           ; NEXT.PACKET                   5894
000236  126627  000004  000014         CMPB    4(SP),#14                       ; NEXT.PACKET,*                 5896
000244  103402                          BLO     8$
000246  105066  000004                  CLRB    4(SP)                           ; NEXT.PACKET                   5899
000252  005366  000002         8$:     DEC     2(SP)                           ; COUNT                         5855
000256  001264                          BNE     1$
000260  005704                 9$:     TST     R4                              ; INDEX                         5903
000262  002435                          BLT     11$
000264  105764  000000G                 TSTB    PKT.USE(R4)                     ; *(INDEX)                      5904
000270  002432                          BLT     11$
000272  010446                          MOV     R4,-(SP)                        ; INDEX,*                       5908
000274  012746  000106                  MOV     #106,-(SP)
000300  004737  000000G                 JSR     PC,BL$MUL
000304  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000312  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)             ;                               5909
000320  152760  000001  000010G         BISB    #1,MSCP.PKT+10(R0)
000326  005000                          CLR     R0                              ;                               5910
000330  156600  000010                  BISB    10(SP),R0                       ; NEXT.PACKET,*
```

```
000334  005200                          INC     R0
000336  110037  000000G                 MOVB    R0,NEXT.PKT.USE
000342  120027  000014                  CMPB    R0,#14                  ; NEXT.PKT.USE,*                5912
000346  103402                          BLO     10$
000350  105037  000000G                 CLRB    NEXT.PKT.USE            ;                               5914
000354  022626                  10$:    CMP     (SP)+,(SP)+             ;                               5907
000356  010400                  11$:    MOV     R4,R0                   ; INDEX,*                       5844
000360  062706  000006                  ADD     #6,SP                   ;                               5832
000364  000207                          RTS     PC

; Routine Size:  123 words,    Routine Base:  $CODE$ + 5616
; Maximum stack depth per invocation:  13 words


;   5921  1
;   5922  1
```

```
;    5923  1
;    5924  1
;    5925  1     global routine PUT_PKT (index) : novalue =
;    5926  1
;    5927  1     !+
;    5928  1     !       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    5929  1     !       ROUTINE.
;    5930  1     !-
;    5931  1
;    5932  1
;    5933  2         begin
;    5934  2
;    5935  2
;    5936  2         local
;    5937  2             RING_ADDR : word,
;    5938  2             OWNER : word;
;    5939  2
;    5940  2         RING_ADDR = .DCT_ADDR [RR_BEG];                                 ! ADDRESS IN FIRST RESPONSE RING
;    5941  2
;    5942  2         incr COUNT from 1 to (RRING_LEN + CRING_LEN) do                 ! FOR EACH ADDRESS IN THE RINGS
;    5943  3             begin
;    5944  3
;    5945  3             if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR              ! IF ADDRESS MATCHES
;    5946  3
;    5947  3             then
;    5948  4                 begin
;    5949  4                 OWNER = .RING_ADDR + 2;                                 ! ADDRESS OF OWNERSHIP WORD
;    5950  4                 .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;   ! GIVE OWNERSHIP TO HOST
;    5951  3                 end;
;    5952  3
;    5953  3
;    5954  3                 RING_ADDR = .RING_ADDR + 4;                             ! LOOK AT NEXT PACKET ADDRESS IN RING
;    5955  2                 end;
;    5956  2
;    5957  2
;    5958  2             PKT_USE [.index] = -1;
;    5959  2
;    5960  1             end;


                                        .SBTTL  PUT.PKT GLOBAL ROUTINES
000000  004137  000000G        PUT.PKT::
                                        JSR     R1,$SAVE4                       ;                                       5925
000004  013700  000000G                 MOV     DCT.ADDR,R0                     ;                                       5940
000010  016001  000004                  MOV     4(R0),R1                        ; *,RING.ADDR
000014  016602  000014                  MOV     14(SP),R2                       ; INDEX,*                               5945
000020  010246                          MOV     R2,-(SP)
000022  012746  000106                  MOV     #106,-(SP)
000026  004737  000000G                 JSR     PC,BL$MUL
000032  012704  000010                  MOV     #10,R4                          ; *,COUNT                               5942
000036  026011  000000G        1$:     CMP     MSCP.PKT(R0),(R1)               ; *,RING.ADDR                           5945
000042  001005                          BNE     2$
000044  012703  000002                  MOV     #2,R3                           ; *,OWNER                               5949
```

```
000050  060103                          ADD     R1,R3                           ; RING.ADDR,OWNER
000052  042713  140000                  BIC     #140000,(R3)                    ; *,OWNER                       5950
000056  062701  000004          2$:     ADD     #4,R1                           ; *,RING.ADDR                   5954
000062  005304                          DEC     R4                              ; COUNT                         5942
000064  001364                          BNE     1$
000066  112762  000377  000000G         MOVB    #377,PKT.USE(R2)                ;                               5958
000074  022626                          CMP     (SP)+,(SP)+                     ;                               5933
000076  000207                          RTS     PC                              ;                               5925

; Routine Size:  32 words,      Routine Base:  $CODE$ + 6204
; Maximum stack depth per invocation:  8 words


;   5961  1
;   5962  1
```

```
;    5963  1     routine PUTA_PKT (CTLR) : novalue =
;    5964  1
;    5965  1     !+
;    5966  1     !       THIS ROUTINE DEALLOCATES ALL MSCP PACKETS WHICH HAVE BEEN ALLOCATED
;    5967  1     !       TO A PARTICULAR CONTROLLER.
;    5968  1     !
;    5969  1     !       INPUTS:
;    5970  1     !               CTLR - CONTROLLER NUMBER
;    5971  1     !-
;    5972  1
;    5973  1         incr COUNT from 0 to (PKT_CNT - 1) do       ! FOR EACH ENTRY IN ALLOCATION TABLE
;    5974  1
;    5975  1             if .PKT_USE [.COUNT] eql .CTLR          ! IF PACKET IS ALLOCATED TO GIVEN CONTROLLER
;    5976  1             then
;    5977  1                 PKT_USE [.COUNT] = -1;              ! DEALLOCATE IT


                                        .SBTTL  PUTA.PKT GLOBAL ROUTINES
000000  010146                 PUTA.PKT:
                                        MOV     R1,-(SP)                        ;                               5963
000002  005000                          CLR     R0                              ; COUNT                         5973
000004  116001  000000G         1$:     MOVB    PKT.USE(R0),R1                  ; *(COUNT),*                    5975
000010  020166  000004                  CMP     R1,4(SP)                        ; *,CTLR
000014  001003                          BNE     2$
000016  112760  000377  000000G        MOVB    #377,PKT.USE(R0)                ; *,*(COUNT)                    5977
000024  005200                  2$:     INC     R0                              ; COUNT                         5973
000026  020027  000013                  CMP     R0,#13                          ; COUNT,*
000032  003764                          BLE     1$
000034  012601                          MOV     (SP)+,R1                        ;                               5963
000036  000207                          RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 6304
; Maximum stack depth per invocation:  2 words
```

```
;    5978  1     global routine GET_RETPKT (CTLR) =
;    5979  1
;    5980  1     !+
;    5981  1     !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;    5982  1     !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    5983  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;    5984  1     !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    5985  1     !
;    5986  1     !       INPUTS:
;    5987  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    5988  1     !-
;    5989  1
;    5990  2         begin
;    5991  2
;    5992  2         local
;    5993  2             index : signed word initial (-1);       ! ASSUME NONE AVAILABLE
;    5994  2
;    5995  2         incr COUNT from 0 to (RP_CNT - 1) do        ! FOR EACH ENTRY IN TABLE
;    5996  2
;    5997  2             if .RP_USE [.COUNT] lss 0               ! IF FREE RETPKT IS FOUND
;    5998  2             then
;    5999  3                 begin
;    6000  3                 RP_USE [.COUNT] = .CTLR;            ! ALLOCATE RETURN PACKET TO CONTROLLER
;    6001  3                 index = .COUNT;
;    6002  3
;    6003  3                 incr J from 0 to (RP_LEN - 1) do    ! ZERO OUT RETPKT
;    6004  3                     RETPKT [.COUNT, .J, 0, 16, 0] = 0;
;    6005  3
;    6006  3                 exitloop;                           ! DONE
;    6007  2                 end;
;    6008  2
;    6009  2         return .index;                              ! RETURN PACKET INDEX (OR -1) TO CALLER
;    6010  1         end;


                                        .SBTTL  GET.RETPKT GLOBAL ROUTINES
000000  004137  000000G        GET.RETPKT::
                                JSR     R1,$SAVE4                       ;                       5978
000004  012703  177777                  MOV     #-1,R3                  ; *,INDEX               5990
000010  005001                          CLR     R1                      ; COUNT                 5995
000012  105761  000000G        1$:     TSTB    RP.USE(R1)              ; *(COUNT)              5997
000016  002025                          BGE     3$
000020  116661  000014  000000G        MOVB    14(SP),RP.USE(R1)       ; CTLR,*(COUNT)         6000
000026  010103                          MOV     R1,R3                   ; COUNT,INDEX           6001
000030  010146                          MOV     R1,-(SP)                ; COUNT,*               6004
000032  012746  000026                  MOV     #26,-(SP)
000036  004737  000000G                JSR     PC,BL$MUL
000042  022626                          CMP     (SP)+,(SP)+
000044  005002                          CLR     R2                      ; J                     6003
000046  010004                2$:     MOV     R0,R4                   ;                       6004
000050  060204                          ADD     R2,R4                   ; J,*
000052  006304                          ASL     R4
000054  005064  000000G                CLR     RETPKT(R4)
```

```
000060  005202                          INC     R2                      ; J                     6003
000062  020227  000025                  CMP     R2,#25                  ; J,*
000066  003767                          BLE     2$
000070  000404                          BR      4$                      ;                       5999
000072  005201                  3$:     INC     R1                      ; COUNT                 5995
000074  020127  000007                  CMP     R1,#7                   ; COUNT,*
000100  003744                          BLE     1$
000102  010300                  4$:     MOV     R3,R0                   ; INDEX,*               5990
000104  000207                          RTS     PC                      ;                       5978

; Routine Size:  35 words,      Routine Base:  $CODE$ + 6344
; Maximum stack depth per invocation:  8 words
```

```
;   6011  1     global routine PUT_RETPKT (index) : novalue =
;   6012  1
;   6013  1     !+
;   6014  1     !       THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;   6015  1     !       ROUTINE.
;   6016  1     !-
;   6017  1
;   6018  1         RP_USE [.index] = -1;


                                        .SBTTL  PUT.RETPKT GLOBAL ROUTINES
000000  016600  000002             PUT.RETPKT::
                                        MOV     2(SP),R0                        ; INDEX,*                       6018
000004  112760  000377  000000G         MOVB    #377,RP.USE(R0)
000012  000207                          RTS     PC                              ;                               6011

; Routine Size:  6 words,       Routine Base:  $CODE$ + 6452
; Maximum stack depth per invocation:  0 words
```

```
;    6019  1
;    6020  1
;    6021  1     global routine GET_IO_BUFF (ADDR) : novalue =
;    6022  1
;    6023  1
;    6024  1     !+
;    6025  1     !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
;    6026  1     !       POOL.
;    6027  1     !
;    6028  1     !       INPUTS:
;    6029  1     !               ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
;    6030  1     !
;    6031  1     !       IMPLICIT INPUTS:
;    6032  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    6033  1     !
;    6034  1     !       OUTPUTS:
;    6035  1     !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
;    6036  1     !               WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A ZERO IS RETURNED
;    6037  1     !               AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
;    6038  1     !-
;    6039  1
;    6040  1
;    6041  1
;    6042  2         begin
;    6043  2         .ADDR = 0;                                          ! ASSUME FAILURE
;    6044  2
;    6045  2         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do   ! FOR EACH ENTRY IN BUFFER TABLE
;    6046  2
;    6047  2             if .BUFF_OWN [.COUNT] lss 0                     ! IF BUFFER IS FREE
;    6048  2
;    6049  2             then
;    6050  2
;    6051  3                 begin
;    6052  3                 BUFF_OWN [.COUNT] = .CCTLR;                 ! ALLOCATE BUFFER TO CONTROLLER
;    6053  3                 .ADDR = .BUFF_ADDR [.COUNT];                ! RETURN BUFFER DESCRIPTOR
;    6054  3                 exitloop;                                   ! DONE
;    6055  2                 end;
;    6056  2
;    6057  2
;    6058  1         end;                                                ! ROUTINE GET_IO_BUFF


                                    .SBTTL  GET.IO.BUFF GLOBAL ROUTINES
000000  010146                GET.IO.BUFF::
                                    MOV     R1,-(SP)                    ;                       6021
000002  005076  000004              CLR     @4(SP)                      ; ADDR                  6043
000006  005001                      CLR     R1                          ; COUNT                 6045
000010  105761  000000G       1$:   TSTB    BUFF.OWN(R1)                ; *(COUNT)              6047
000014  002011                      BGE     2$
000016  113761  000000G 000000G     MOVB    CCTLR,BUFF.OWN(R1)          ; *,*(COUNT)            6052
000024  010100                      MOV     R1,R0                       ; COUNT,*               6053
000026  006300                      ASL     R0
000030  016076  000000G 000004      MOV     BUFF.ADDR(R0),@4(SP)        ; *,ADDR
```

```
000036  000404                          BR      3$                              ;                                       6051
000040  005201                  2$:     INC     R1                              ; COUNT                                 6045
000042  020127  000007                  CMP     R1,#7                           ; COUNT,*
000046  003760                          BLE     1$
000050  012601                  3$:     MOV     (SP)+,R1                        ;                                       6021
000052  000207                          RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 6466
; Maximum stack depth per invocation:  2 words


;   6059  1
;   6060  1
```

```
;    6061  1     global routine PUT_IO_BUFF (ADDR) : novalue =
;    6062  1
;    6063  1     !+
;    6064  1     !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;    6065  1     !       TO THE BUFFER POOL.
;    6066  1     !
;    6067  1     !       INPUTS:
;    6068  1     !               ADDR - ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
;    6069  1     !                       DEALLOCATED
;    6070  1     !-
;    6071  1
;    6072  1         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;    6073  1
;    6074  1             if .BUFF_ADDR [.COUNT] eqla ..ADDR                      ! IF THIS IS THE BUFFER'S ENTRY
;    6075  1             then
;    6076  2                 begin
;    6077  2                 BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE BUFFER
;    6078  2                 exitloop;                                           ! DONE
;    6079  1                 end;


                                .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  010146                 PUT.IO.BUFF::
                                MOV     R1,-(SP)                        ;                                       6061
000002  005001                  CLR     R1                              ; COUNT                                 6072
000004  010100                  1$:     MOV     R1,R0                   ; COUNT,*                               6074
000006  006300                          ASL     R0
000010  026076  000000G 000004          CMP     BUFF.ADDR(R0),@4(SP)            ; *,ADDR
000016  001004                          BNE     2$
000020  112761  000377  000000G         MOVB    #377,BUFF.OWN(R1)               ; *,*(COUNT)                    6077
000026  000404                          BR      3$                              ;                               6076
000030  005201                  2$:     INC     R1                              ; COUNT                         6072
000032  020127  000007                  CMP     R1,#7                           ; COUNT,*
000036  003762                          BLE     1$
000040  012601                  3$:     MOV     (SP)+,R1                        ;                               6061
000042  000207                          RTS     PC

; Routine Size:  18 words,      Routine Base:  $CODE$ + 6542
; Maximum stack depth per invocation:  2 words
```

```
;   6080  1     global routine PUTA_BUFF : novalue =
;   6081  1
;   6082  1     !+
;   6083  1     !       THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
;   6084  1     !       THE CURRENT CONTROLLER (CCTLR).
;   6085  1     !-
;   6086  1
;   6087  1         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;   6088  1
;   6089  1             if .BUFF_OWN [.COUNT] eql .CCTLR                        ! IF THIS BUFFER ALLOCATED TO CURRENT CONTROLLER
;   6090  1             then
;   6091  1                 BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE IT


                                    .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                 PUTA.BUFF::
                                    MOV     R1,-(SP)                    ;                                   6080
000002  005000                      CLR     R0                          ; COUNT                             6087
000004  116001  000000G        1$:  MOVB    BUFF.OWN(R0),R1             ; *(COUNT),*                        6089
000010  020137  000000G             CMP     R1,CCTLR
000014  001003                      BNE     2$
000016  112760  000377  000000G     MOVB    #377,BUFF.OWN(R0)           ; *,*(COUNT)                        6091
000024  005200                 2$:  INC     R0                          ; COUNT                             6087
000026  020027  000007              CMP     R0,#7                       ; COUNT,*
000032  003764                      BLE     1$
000034  012601                      MOV     (SP)+,R1                    ;                                   6080
000036  000207                      RTS     PC

; Routine Size:  16 words,    Routine Base:  $CODE$ + 6606
; Maximum stack depth per invocation:  2 words
```

```
;   6092  1     global routine OUT_IODQ =
;   6093  1
;   6094  1     !+
;   6095  1     !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;   6096  1     !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;   6097  1     !       QUEUE IS ALSO UPDATED.
;   6098  1     !
;   6099  1     !       INPUTS:
;   6100  1     !               NONE
;   6101  1     !
;   6102  1     !       OUTPUTS:
;   6103  1     !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;   6104  1     !-
;   6105  1
;   6106  2         begin
;   6107  2
;   6108  2         local
;   6109  2             index : word;
;   6110  2
;   6111  2         index = .IODQ [.IODQ_OUT];                  ! GET NEXT RETPKT INDEX
;   6112  2         IODQ_OUT = .IODQ_OUT + 1;                   ! ADVANCE "OUT" POINTER
;   6113  2
;   6114  2         if .IODQ_OUT gequ IODQ_LEN                  ! IF BEYOND END OF QUEUE
;   6115  2         then
;   6116  2             IODQ_OUT = 0;                           ! SET POINTER TO BEGINNING OF QUEUE
;   6117  2
;   6118  2         return .index;                              ! RETURN INDEX TO CALLER
;   6119  1         end;


                                        .SBTTL  OUT.IODQ GLOBAL ROUTINES
000000  013700  000000G           OUT.IODQ::
                                        MOV     IODQ.OUT,R0                 ;                   6111
000004  116000  000000G                 MOVB    IODQ(R0),R0                 ; *,INDEX
000010  042700  177400                  BIC     #177400,R0                  ; *,INDEX
000014  005237  000000G                 INC     IODQ.OUT                    ;                   6112
000020  023727  000000G 000010          CMP     IODQ.OUT,#10                ;                   6114
000026  103402                          BLO     1$
000030  005037  000000G                 CLR     IODQ.OUT                    ;                   6116
000034  000207                  1$:     RTS     PC                          ;                   6092

; Routine Size:  15 words,      Routine Base:  $CODE$ + 6646
; Maximum stack depth per invocation:  0 words
```

```
;    6120  1     global routine IN_IODQ (index) : novalue =
;    6121  1
;    6122  1     !+
;    6123  1     !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;    6124  1     !       UPDATES THE IODQ_IN POINTER.
;    6125  1     !-
;    6126  1
;    6127  1         if ((.IODQ_IN + 1) eql .IODQ_OUT) or
;    6128  2            (.IODQ_IN - (IODQ_LEN - 1) eql .IODQ_OUT)
;    6129  1         then
;    6130  1             return
;    6131  1         else
;    6132  2             begin
;    6133  2             IODQ [.IODQ_IN] = .index;              ! LOAD INDEX INTO QUEUE
;    6134  2             IODQ_IN = .IODQ_IN + 1;                ! ADVANCE "IN" POINTER
;    6135  2
;    6136  2             if .IODQ_IN gequ IODQ_LEN              ! IF BEYOND END OF QUEUE
;    6137  2             then
;    6138  2                 IODQ_IN = 0;                       ! CYCLE BACK TO BEGINNING OF QUEUE
;    6139  2
;    6140  1             end;                                   ! IF IODQ IS NOT FULL


                                        .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                    IN.IODQ::
                                        MOV     R1,-(SP)                        ;                       6120
000002  013701  000000G                 MOV     IODQ.IN,R1                      ;                       6127
000006  010100                          MOV     R1,R0
000010  005200                          INC     R0
000012  020037  000000G                 CMP     R0,IODQ.OUT
000016  001421                          BEQ     1$
000020  010100                          MOV     R1,R0                           ;                       6128
000022  162700  000007                  SUB     #7,R0
000026  020037  000000G                 CMP     R0,IODQ.OUT
000032  001413                          BEQ     1$                              ;                       6130
000034  116661  000004  000000G         MOVB    4(SP),IODQ(R1)                  ; INDEX,*               6133
000042  005237  000000G                 INC     IODQ.IN                         ;                       6134
000046  023727  000000G 000010          CMP     IODQ.IN,#10                     ;                       6136
000054  103402                          BLO     1$
000056  005037  000000G                 CLR     IODQ.IN                         ;                       6138
000062  012601                    1$:   MOV     (SP)+,R1                        ;                       6120
000064  000207                          RTS     PC

; Routine Size:  27 words,      Routine Base:  $CODE$ + 6704
; Maximum stack depth per invocation:  2 words
```

```
;    6141  1
;    6142  1
;    6143  1     global routine DROP_CTLR (CTLR, REASON) : novalue =
;    6144  1
;    6145  1
;    6146  1
;    6147  1     !+
;    6148  1     !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;    6149  1     !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;    6150  1     !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;    6151  1     !       SECTION.
;    6152  1     !-
;    6153  1
;    6154  1
;    6155  2         begin
;    6156  2
;    6157  2         local
;    6158  2             UNIT;
;    6159  2
;    6160  2         incr N from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do       ! FOR EACH UNIT
;    6161  2
;    6162  2             if .CST [.CTLR, .N + OF_DATA, D_PRES] eql PRESENT                                       ! IF CONFIGURED
;    6163  2             then
;    6164  3                 begin
;    6165  3                 UNIT = .CST [.CTLR, .N + OF_DATA, D_UNIT];                                          ! DRS UNIT NUMBER
;    6166  3                 DUR [.UNIT] = .REASON;                                                              ! DROP REASON
;    6167  3                 DODU (.UNIT);                                                                       ! DROP UNIT
;    6168  2                 end;
;    6169  2
;    6170  1         end;


                                        .SBTTL  DROP.CTLR GLOBAL ROUTINES
000000  004137  000000G           DROP.CTLR::
                                        JSR     R1,$SAVE3                       ;                               6143
000004  016646  000014                  MOV     14(SP),-(SP)                    ; CTLR,*                        6162
000010  012746  000053                  MOV     #53,-(SP)
000014  004737  000000G                 JSR     PC,BL$MUL
000020  010003                          MOV     R0,R3
000022  012702  000003                  MOV     #3,R2                           ; *,N                           6160
000026  010300                  1$:     MOV     R3,R0                           ;                               6162
000030  060200                          ADD     R2,R0                           ; N,*
000032  006300                          ASL     R0
000034  032760  040000  000000G         BIT     #40000,CST(R0)
000042  001412                          BEQ     2$
000044  016001  000000G                 MOV     CST(R0),R1                      ; *,UNIT                        6165
000050  000301                          SWAB    R1                              ; UNIT
000052  042701  177760                  BIC     #177760,R1                      ; *,UNIT
000056  116661  000016  000000G         MOVB    16(SP),DUR(R1)                  ; REASON,*(UNIT)                6166
000064  010100                          MOV     R1,R0                           ; UNIT,*                        6167
000066  104451                          TRAP    51
000070  062702  000012          2$:     ADD     #12,R2                          ; *,N                           6160
000074  020227  000041                  CMP     R2,#41                          ; N,*
```

```
000100  003752                          BLE     1$
000102  022626                          CMP     (SP)+,(SP)+                     ;                       6155
000104  000207                          RTS     PC                              ;                       6143

; Routine Size:  35 words,      Routine Base:  $CODE$ + 6772
; Maximum stack depth per invocation:  8 words


;   6171  1
;   6172  1
```

```
;  6172  1     global routine DRV_CTLERR (CTLR) : novalue =
;  6174  1
;  6175  1     !+
;  6176  1     !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;  6177  1     !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
;  6178  1     !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;  6179  1     !       PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;  6180  1     !       C-RING COUNT, AND DEALLOCATING MSCP PACKETS DESCRIBED IN THE RESPONSE
;  6181  1     !       RING.
;  6182  1     !
;  6183  1     !       INPUTS:
;  6184  1     !               CTLR - DYING CONTROLLER NUMBER
;  6185  1     !-
;  6186  1
;  6187  2         begin
;  6188  2
;  6189  2         local
;  6190  2             D_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS);  ! CONTROLLER'S DCT ADDRESS
;  6191  2
;  6192  2         D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                       ! GET CONTROLLER'S DCT ADDR
;  6193  2         D_ADDR [WORD0] = OFFLINE;                                   ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;  6194  2         PUTA_PKT (.CTLR);                                           ! RELEASE ALL PACKETS ALLOCATED TO CONTROLLER
;  6195  2         DROP_CTLR (.CTLR, DU_CFATAL);                               ! DROP ALL UNITS ON THE CONTROLLER
;  6196  1         end;                                                        ! ROUTINE DRV_CTLERR


                                .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  010146                DRV.CTLERR::
                                MOV     R1,-(SP)                            ;                       6173
000002  016601  000004          MOV     4(SP),R1                            ; CTLR,*                6192
000006  010146                  MOV     R1,-(SP)
000010  012746  000022          MOV     #22,-(SP)
000014  004737  000000G         JSR     PC,BL$MUL
000020  062700  000000G         ADD     #DCT,R0
000024  005010                  CLR     (R0)                                ; D.ADDR                6193
000026  010116                  MOV     R1,(SP)                             ;                       6194
000030  004737  006304'         JSR     PC,PUTA.PKT
000034  010116                  MOV     R1,(SP)                             ;                       6195
000036  012746  000006          MOV     #6,-(SP)
000042  004737  006772'         JSR     PC,DROP.CTLR
000046  062706  000006          ADD     #6,SP                               ;                       6187
000052  012601                  MOV     (SP)+,R1                            ;                       6173
000054  000207                  RTS     PC

; Routine Size:  23 words,      Routine Base:  $CODE$ + 7100
; Maximum stack depth per invocation:  5 words
```

```
;  6197  1     global routine SEND (index) =
;  6198  1
;  6199  1     !+
;  6200  1     !       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;  6201  1     !       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
;  6202  1     !       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
;  6203  1     !       DEVICE'S IP REGISTER.  IF THE
;  6204  1     !       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;  6205  1     !       THE CALLER, AND NO ACTION IS TAKEN.
;  6206  1     !
;  6207  1     !       INPUTS:
;  6208  1     !               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
;  6209  1     !                       BE SENT
;  6210  1     !
;  6211  1     !       IMPLICIT INPUTS:
;  6212  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  6213  1     !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;  6214  1     !-
;  6215  1
;  6216  2         begin
;  6217  2
;  6218  2         local
;  6219  2             SLOT_ADDR,
;  6220  2             TEMP : word,
;  6221  2             CUR_PRIORITY : word;
;  6222  2
;  6223  2         if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                   ! IF CRING IS NOT FULL AND
;  6224  3            ((.DCT_ADDR [STAT] eql ONLINE) or                            !   IF DEVICE IS ONLINE OR
;  6225  3             (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                    !   IT IS A SET-CTRL-CHAR COMMAND
;  6226  2         then
;  6227  2
;  6228  4             if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
;  6229  4                      (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
;  6230  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                  !ZZZ
;  6231  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                  !ZZZ
;  6232  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                  !ZZZ
;  6233  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                  !ZZZ
;  6234  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                  !ZZZ
;  6235  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ESP) OR                  !ZZZ
;  6236  3                      (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
;  6237  2             then
;  6238  3                 begin
;  6239  3                 PRINTF (DBM107, .MSCP_PKT [.index, OPCODE]);
;  6240  3                 return FAILURE;
;  6241  3                 end
;  6242  2             else
;  6243  3                 begin
;  6244  3
;  6245  3                 do
;  6246  3                     BREAK                                               ! LOOP TILL CREDIT BALANCE POSITIVE
;  6247  4                 until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
;  6248  3                        (.CREDIT_BAL gequ 1)) or
;  6249  3                       (.CREDIT_BAL gtru 1);
```

```
;   6250  3
;   6251  3                 MSCP_PKT [.index, CRN_LO] = (CRN_LOW = .CRN_LOW + 1);        ! ASSIGN CMD REF NUM
;   6252  3
;   6253  3                 if .CRN_LOW eql 0
;   6254  3                 then
;   6255  3                     CRN_HIGH = .CRN_HIGH + 1;                                ! CMD REF NUM (HIGH ORDER)
;   6256  3
;   6257  3                 MSCP_PKT [.index, CRN_HI] = .CRN_HIGH;                       !
;   6258  3                 SLOT_ADDR = .DCT_ADDR [CR_NEXT];                             ! ADDR OF NEXT COMMAND SLOT
;   6259  3
;   6260  3                 do                                                           ! WAIT TILL NEXT SLOT HOST OWNED
;   6261  3                     BREAK                                                    !
;   6262  3                 until ((.(.SLOT_ADDR + 2) and ED_OWN) eql 0);                !
;   6263  3
;   6264  3                 GETPRI (CUR_PRIORITY);                                       ! NO INTERRUPTS WHILE POINTERS UPDATED
;   6265  3                 SETPRI (PRI04);                                              !
;   6266  3
;   6267  3                 .SLOT_ADDR = .MSCP_PKT [.index, PKT_LO];                     ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
;   6268  3                 SLOT_ADDR = .SLOT_ADDR + 2;                                  ! ADVANCE TO NEXT WORD
;   6269  3                 .SLOT_ADDR = .MSCP_PKT [.index, PKT_HI];                     ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
;   6270  3                 .SLOT_ADDR = ..SLOT_ADDR and (not (ED_FLAG));                ! CLEAR INTERRUPT FLAG IN CASE SET
;   6271  3                 .SLOT_ADDR = ..SLOT_ADDR or ED_OWN;                          ! GIVE OWNERSHIP TO CONTROLLER
;   6272  3                 SLOT_ADDR = .SLOT_ADDR + 2;                                  ! ADVANCE TO NEXT COMMAND SLOT
;   6273  3
;   6274  3                 if .SLOT_ADDR gtra .DCT_ADDR [CR_END]                        ! IF BEYOND END OF CRING
;   6275  3                 then
;   6276  3                     SLOT_ADDR = .DCT_ADDR [CR_BEG];                          ! CYCLE BACK TO BEGINNING
;   6277  3
;   6278  3                 DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                             ! RESTORE CR_NEXT POINTER IN DCT
;   6279  3                 DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;            ! INCR # OF COMMANDS IN CRING
;   6280  4                 IF (.MSCP_PKT [.INDEX, CONNID] EQL CID_MSCP)                 !IF MSCP COMMAND                ZZZ
;   6281  3                 THEN (CREDIT_BAL = .CREDIT_BAL - 1);                         !DECR CREDIT BALANCE            ZZZ
;   6282  3                 TEMP = .RDRX_ADDR [RCIP, RC_ALL];                            ! READ IP TO FORCE PORT TO POLL
;   6283  3                 SETPRI (.CUR_PRIORITY);                                      ! LOWER PRIORITY
;   6284  3                 return SUCCESS;
;   6285  3                 end
;   6286  3
;   6287  2         else
;   6288  2             return FAILURE;                                                  ! IF DEVICE IS NOT ONLINE
;   6289  2
;   6290  1         end;                                                                 ! ROUTINE SEND


                                                .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G                 SEND::  JSR     R1,$SAVE3                       ;                                   6197
000004  005746                                  TST     -(SP)
000006  127727  000000G 000004                  CMPB    @DCT.ADDR,#4                    ;                                   6223
000014  103100                                  BHIS    2$
000016  005777  000000G                         TST     @DCT.ADDR                       ;                                   6224
000022  100413                                  BMI     1$
000024  016646  000014                          MOV     14(SP),-(SP)                    ; INDEX,*                           6225
000030  012746  000106                          MOV     #106,-(SP)
000034  004737  000000G                         JSR     PC,BL$MUL
```

```
000040  022626                          CMP     (SP)+,(SP)+
000042  126027  000022G 000004          CMPB    MSCP.PKT+22(R0),#4
000050  001167                          BNE     10$
000052  016646  000014          1$:     MOV     14(SP),-(SP)            ; INDEX,*                       6228
000056  012746  000106                  MOV     #106,-(SP)
000062  004737  000000G                 JSR     PC,BL$MUL
000066  010002                          MOV     R0,R2
000070  022626                          CMP     (SP)+,(SP)+
000072  005000                          CLR     R0
000074  156200  000022G                 BISB    MSCP.PKT+22(R2),R0
000100  020027  000020                  CMP     R0,#20
000104  001445                          BEQ     3$
000106  020027  000011                  CMP     R0,#11
000112  001442                          BEQ     3$
000114  020027  000041                  CMP     R0,#41                  ;                               6229
000120  001437                          BEQ     3$
000122  020027  000004                  CMP     R0,#4
000126  001434                          BEQ     3$
000130  020027  000005                  CMP     R0,#5                   ;                               6231
000134  001431                          BEQ     3$
000136  020027  000001                  CMP     R0,#1                   ;                               6232
000142  001426                          BEQ     3$
000144  020027  000003                  CMP     R0,#3                   ;                               6233
000150  001423                          BEQ     3$
000152  020027  000006                  CMP     R0,#6                   ;                               6234
000156  001420                          BEQ     3$
000160  020027  000002                  CMP     R0,#2                   ;                               6235
000164  001415                          BEQ     3$
000166  020027  000042                  CMP     R0,#42                  ;                               6236
000172  001412                          BEQ     3$
000174  010046                          MOV     R0,-(SP)                ;                               6239
000176  012746  000000G                 MOV     #DBM107,-(SP)
000202  012746  000002                  MOV     #2,-(SP)
000206  010600                          MOV     SP,R0                   ; SP,*
000210  104417                          TRAP    17
000212  062706  000006                  ADD     #6,SP                   ;                               6238
000216  000504                  2$:     BR      10$                     ;                               6228
000220  104422                  3$:     TRAP    22                      ;                               6245
000222  105762  000004G                 TSTB    MSCP.PKT+4(R2)          ;                               6247
000226  001003                          BNE     4$
000230  005737  000000G                 TST     CREDIT.BAL              ;                               6248
000234  001004                          BNE     5$
000236  023727  000000G 000001  4$:     CMP     CREDIT.BAL,#1           ;                               6249
000244  101765                          BLOS    3$
000246  013700  000000G          5$:     MOV     CRN.LOW,R0              ;                               6251
000252  005200                          INC     R0
000254  010037  000000G                 MOV     R0,CRN.LOW
000260  010062  000012G                 MOV     R0,MSCP.PKT+12(R2)
000264  001002                          BNE     6$                      ;                               6253
000266  005237  000000G                 INC     CRN.HIGH                ;                               6255
000272  013762  000000G 000014G  6$:     MOV     CRN.HIGH,MSCP.PKT+14(R2)  ;                             6257
000300  013700  000000G                 MOV     DCT.ADDR,R0             ;                               6258
000304  016001  000020                  MOV     20(R0),R1               ; *,SLOT.ADDR
```

```
000310  104422                  7$:     TRAP    22                              ;                               6260
000312  032761  100000  000002            BIT     #-100000,2(R1)                  ; *,*(SLOT.ADDR)                6262
000320  001373                            BNE     7$
000322  104440                            TRAP    40                              ;                               6264
000324  010003                            MOV     R0,R3                           ; *,CUR.PRIORITY
000326  012700  000200                     MOV     #200,R0                         ;                               6265
000332  104441                            TRAP    41
000334  016221  000000G                   MOV     MSCP.PKT(R2),(R1)+              ; *,SLOT.ADDR                   6267
000340  016211  000002G                   MOV     MSCP.PKT+2(R2),(R1)             ; *,SLOT.ADDR                   6269
000344  042711  040000                    BIC     #40000,(R1)                     ; *,SLOT.ADDR                   6270
000350  052721  100000                    BIS     #100000,(R1)+                   ; *,SLOT.ADDR                   6271
000354  013700  000000G                   MOV     DCT.ADDR,R0                     ;                               6274
000360  020160  000012                    CMP     R1,12(R0)                       ; SLOT.ADDR,*
000364  101402                            BLOS    8$
000366  016001  000010                    MOV     10(R0),R1                       ; *,SLOT.ADDR                   6276
000372  010160  000020          8$:       MOV     R1,20(R0)                       ; SLOT.ADDR,*                   6278
000376  105210                            INCB    (R0)                            ;                               6279
000400  105762  000011G                   TSTB    MSCP.PKT+11(R2)                 ;                               6280
000404  001002                            BNE     9$
000406  005337  000000G                   DEC     CREDIT.BAL                      ;                               6281
000412  017716  000000G          9$:       MOV     @RDRX.ADDR,(SP)                 ; *,RC.REG                      6282
000416  010300                            MOV     R3,R0                           ; CUR.PRIORITY,*                6283
000420  104441                            TRAP    41
000422  012700  000001                     MOV     #1,R0                           ;                               6228
000426  000401                            BR      11$                             ;                               6288
000430  005000                  10$:      CLR     R0
000432  005726                  11$:      TST     (SP)+                           ;                               6197
000434  000207                            RTS     PC

; Routine Size:  143 words,    Routine Base:  $CODE$ + 7156
; Maximum stack depth per invocation:  10 words
```

```
;   6291  1     global routine WAIT : novalue =
;   6292  1
;   6293  1     !+
;   6294  1     !       THE PURPOSE OF THIS ROUTINE IS TO KILL TIME UNTIL AN RDRX INTERRUPT
;   6295  1     !       RESULTS IN A RETURN PACKET INDEX BEING DEPOSITED INTO THE I/O DONE
;   6296  1     !       QUEUE (IODQ).
;   6297  1     !-
;   6298  1
;   6299  1         do
;   6300  1             BREAK                                   ! BREAK FOR ACT
;   6301  1         until .IODQ_IN neq .IODQ_OUT;


                                        .SBTTL  WAIT GLOBAL ROUTINES
000000  104422                         WAIT::
000000                                  1$:     TRAP    22                      ;                       6299
000002  023737  000000G 000000G                 CMP     IODQ.IN,IODQ.OUT        ;                       6301
000010  001773                                 BEQ     1$
000012  000207                                 RTS     PC                      ;                       6291

; Routine Size:  6 words,       Routine Base:  $CODE$ + 7614
; Maximum stack depth per invocation:  2 words


;   6302  1
```

```
;    6303  1
;    6304  1     GLOBAL ROUTINE MODULAS (LO_LIMIT, HI_LIMIT) =                                  !ZZZ
;    6305  1                                                                                    !ZZZ
;    6306  1     !+ THE PURPOSE OF THIS ROUTINE IS TO GET A RANDOM NUMBER BETWEEN               !ZZZ
;    6307  1     !  THE LOW AND HIGH LIMITS. THIS SHOULD WORK FOR A 16 BIT WORD.                !ZZZ
;    6308  1     !- THE "MOD" FUNC ONLY WORKS ON 15 BITS.                                       !ZZZ
;    6309  1                                                                                    !ZZZ
;    6310  2     BEGIN                                                                          !ZZZ
;    6311  2     OWN     X : WORD;                   !VARIABLE FOR RANDOM WD TABLE              !ZZZ
;    6312  2     LOCAL   ANSWER : UNSIGNED WORD,     !FINAL ANSWER                              !ZZZ
;    6313  2             SAVESZ : UNSIGNED WORD,     !SAVES SIZE OF WINDOW                      !ZZZ
;    6314  2             SIZE : UNSIGNED WORD;       !SIZE OF WINDOW                            !ZZZ
;    6315  2                                                                                    !ZZZ
;    6316  2                                                                                    !ZZZ
;    6317  2     X = .X + 1;                                                                    !ZZZ
;    6318  2     IF .X GEQ RDM_LEN                                                              !ZZZ
;    6319  2     THEN X = 0;                 !KEEP ROTATING RANDOM NUMBERS USED                 !ZZZ
;    6320  2
                         !ZZZ
;    6321  2     SIZE = .HI_LIMIT - .LO_LIMIT;                                                  !ZZZ
;    6322  2     SAVESZ = .HI_LIMIT - .LO_LIMIT;                                                !ZZZ
;    6323  3     IF (.SIZE LEQU %O'077777')      !IF BIT 15 NOT SET                             !ZZZ
;    6324  3     THEN ANSWER = ((.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1))                  !ZZZ
;    6325  3                                     !ONLY 15 BIT WD, SO TAKE RANDOM SAMPLE         !ZZZ
;    6326  2     ELSE                            !16 BIT WD                                     !ZZZ
;    6327  3             BEGIN                                                                  !ZZZ
;    6328  3             SIZE = .SIZE ^ -1;      !MAKES SIZE A 15 BIT LENGTH, OR DIV BY 2       !ZZZ
;    6329  3             ANSWER = (.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1);                !ZZZ
;    6330  3                                     !GIVES 15 BIT RANDOM NUMBER                    !ZZZ
;    6331  3             ANSWER = .ANSWER ^ 1;   !BUILD UP TO REGULAR SIZE                      !ZZZ
;    6332  3             ANSWER = .ANSWER + (.RANDOM [.X + 1] AND 1);                           !ZZZ
;    6333  3                                     !RANDOMLY FILL BIT 0                           !ZZZ
;    6334  4             IF (.ANSWER GTRU SAVESZ) !ITS POSSIBLE TO BE 1 LARGER THAN SIZE        !ZZZ
;    6335  3             THEN ANSWER = .SAVESZ;  !SO CHECK.                                     !ZZZ
;    6336  2             END;                                                                   !ZZZ
;    6337  2     RETURN .ANSWER;
;    6338  1     END;                            !END MODULAS ROUTINE                           !ZZZ


007630                              X:      .BLKW   1


                                            .SBTTL  MODULAS GLOBAL ROUTINES
000000  004137  000000G             MODULAS::
                                            JSR     R1,$SAVE2                   ;                   6304
000004  005746                              TST     -(SP)
000006  005237  007630'                      INC     X                           ;                   6317
000012  023727  007630' 000020              CMP     X,#20                       ;                   6318
000020  002402                              BLT     1$
000022  005037  007630'                      CLR     X                           ;                   6319
000026  016600  000012              1$:     MOV     12(SP),R0                   ; HI.LIMIT,*        6321
000032  166600  000014                      SUB     14(SP),R0                   ; LO.LIMIT,*
000036  010001                              MOV     R0,R1                       ; *,SIZE
000040  010016                              MOV     R0,(SP)                     ; *,SAVESZ          6322
```

```
000042  013700  007630'                    MOV     X,R0                            ;                               6324
000046  006300                             ASL     R0
000050  020127  077777                     CMP     R1,#77777                       ; SIZE,*                        6323
000054  101011                             BHI     2$
000056  016046  000000G                    MOV     RANDOM(R0),-(SP)                ;                               6324
000062  042716  100000                     BIC     #100000,(SP)
000066  010146                             MOV     R1,-(SP)                        ; SIZE,*
000070  005216                             INC     (SP)
000072  004737  000000G                    JSR     PC,BL$MOD
000076  000431                             BR      3$                              ;                               6323
000100  006201                     2$:     ASR     R1                              ; SIZE                          6328
000102  016046  000000G                    MOV     RANDOM(R0),-(SP)                ;                               6329
000106  042716  100000                     BIC     #100000,(SP)
000112  010146                             MOV     R1,-(SP)                        ; SIZE,*
000114  005216                             INC     (SP)
000116  004737  000000G                    JSR     PC,BL$MOD
000122  006300                             ASL     R0                              ; ANSWER                        6331
000124  013701  007630'                    MOV     X,R1                            ;                               6332
000130  006301                             ASL     R1
000132  116102  000002G                    MOVB    RANDOM+2(R1),R2
000136  042702  177776                     BIC     #177776,R2
000142  060200                             ADD     R2,R0                           ; *,ANSWER
000144  012701  000004                     MOV     #4,R1                           ;                               6334
000150  060601                             ADD     SP,R1                           ; SAVESZ,*
000152  020001                             CMP     R0,R1                           ; ANSWER,*
000154  101402                             BLOS    3$
000156  016600  000004                     MOV     4(SP),R0                        ; SAVESZ,ANSWER                 6335
000162  062706  000006             3$:     ADD     #6,SP                           ;                               6304
000166  000207                             RTS     PC

; Routine Size:  60 words,      Routine Base:  $CODE$ + 7632
; Maximum stack depth per invocation:  7 words
```

```
;    6339  1     %sbttl 'ERROR MESSAGE SUBROUTINES'
;    6340  1
;    6341  1     routine EMS_SA : novalue =
;    6342  1
;    6343  1     !+
;    6344  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;    6345  1     !       THE CONTENTS OF THE SA REGISTER.
;    6346  1     !-
;    6347  1
;    6348  2         begin
;    6349  2
;    6350  2         if .SA_REG eql %o'177777'                                   ! IF CONTROLLER TIME-OUT
;    6351  2         then
;    6352  3             begin
;    6353  3             PRINTX (CRLF);
;    6354  3             PRINTX (ASTERISK);
;    6355  3             PRINTX (.CNTR_ERR [0]);
;    6356  3             end
;    6357  2         else
;    6358  2
;    6359  2             if (.SA_REG and %o'003777') lequ 22                     ! IF GENERIC CONTROLLER ERROR
;    6360  2             then
;    6361  3                 begin
;    6362  3                 PRINTX (CRLF);
;    6363  3                 PRINTX (ASTERISK);
;    6364  3                 PRINTX (.CNTR_ERR [.SA_REG and %o'003777']);
;    6365  3                 end
;    6366  2             else
;    6367  2
;    6368  2                 if ((.SA_REG and %o'003777') - 400) lequ 6          ! IF RDRX SPECIFIC CONTROLLER ERROR
;    6369  2                 then
;    6370  3                     begin
;    6371  3                     PRINTX (CRLF);
;    6372  3                     PRINTX (ASTERISK);
;    6373  3                     PRINTX (.RDRX_ERR [(.SA_REG and %o'003777') - 400]);
;    6374  3                     end
;    6375  2                 else
;    6376  2                     PRINTX (EX_SA, .SA_REG);                       ! JUST PRINT CONTENTS OF SA
;    6377  2
;    6378  2         EMS_TIM ();                                                ! TIME
;    6379  1         end;


                                    .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  010146                EMS.SA: MOV     R1,-(SP)                        ;                                   6341
000002  013701  000000G               MOV     SA.REG,R1                       ;                                   6350
000006  020127  177777                CMP     R1,#-1
000012  001023                        BNE     1$
000014  012746  000000G               MOV     #CRLF,-(SP)                     ;                                   6353
000020  012746  000001                MOV     #1,-(SP)
000024  010600                        MOV     SP,R0                           ; SP,*
000026  104415                        TRAP    15
000030  012716  000000G               MOV     #ASTERISK,(SP)                  ;                                   6354
```

```
000034  012746  000001                    MOV     #1,-(SP)
000040  010600                            MOV     SP,R0           ; SP,*
000042  104415                            TRAP    15
000044  013716  000000G                   MOV     CNTR.ERR,(SP)   ;                       6355
000050  012746  000001                    MOV     #1,-(SP)
000054  010600                            MOV     SP,R0           ; SP,*
000056  104415                            TRAP    15
000060  000475                            BR      3$              ;                       6352
000062  010100                    1$:     MOV     R1,R0           ;                       6359
000064  042700  174000                    BIC     #174000,R0
000070  020027  000026                    CMP     R0,#26
000074  101030                            BHI     2$
000076  012746  000000G                   MOV     #CRLF,-(SP)     ;                       6362
000102  012746  000001                    MOV     #1,-(SP)
000106  010600                            MOV     SP,R0           ; SP,*
000110  104415                            TRAP    15
000112  012716  000000G                   MOV     #ASTERISK,(SP)  ;                       6363
000116  012746  000001                    MOV     #1,-(SP)
000122  010600                            MOV     SP,R0           ; SP,*
000124  104415                            TRAP    15
000126  013700  000000G                   MOV     SA.REG,R0       ;                       6364
000132  042700  174000                    BIC     #174000,R0
000136  006300                            ASL     R0
000140  016016  000000G                   MOV     CNTR.ERR(R0),(SP)
000144  012746  000001                    MOV     #1,-(SP)
000150  010600                            MOV     SP,R0           ; SP,*
000152  104415                            TRAP    15
000154  000437                            BR      3$              ;                       6361
000156  010100                    2$:     MOV     R1,R0           ;                       6368
000160  042700  174000                    BIC     #174000,R0
000164  162700  000620                    SUB     #620,R0
000170  020027  000006                    CMP     R0,#6
000174  101031                            BHI     4$
000176  012746  000000G                   MOV     #CRLF,-(SP)     ;                       6371
000202  012746  000001                    MOV     #1,-(SP)
000206  010600                            MOV     SP,R0           ; SP,*
000210  104415                            TRAP    15
000212  012716  000000G                   MOV     #ASTERISK,(SP)  ;                       6372
000216  012746  000001                    MOV     #1,-(SP)
000222  010600                            MOV     SP,R0           ; SP,*
000224  104415                            TRAP    15
000226  013700  000000G                   MOV     SA.REG,R0       ;                       6373
000232  042700  174000                    BIC     #174000,R0
000236  006300                            ASL     R0
000240  016016  176340G                   MOV     RDRX.ERR-1440(R0),(SP)
000244  012746  000001                    MOV     #1,-(SP)
000250  010600                            MOV     SP,R0           ; SP,*
000252  104415                            TRAP    15
000254  005726                    3$:     TST     (SP)+           ;                       6370
000256  000407                            BR      5$              ;                       6368
000260  010146                    4$:     MOV     R1,-(SP)        ;                       6376
000262  012746  000000G                   MOV     #EX.SA,-(SP)
000266  012746  000002                    MOV     #2,-(SP)
```

```
000272  010600                          MOV     SP,R0                   ; SP,*
000274  104415                          TRAP    15
000276  004737  000000V          5$:     JSR     PC,EMS.TIM              ;                       6378
000302  062706  000006                  ADD     #6,SP                   ;                       6348
000306  012601                          MOV     (SP)+,R1                ;                       6341
000310  000207                          RTS     PC

; Routine Size:  101 words,     Routine Base:  $CODE$ + 10022
; Maximum stack depth per invocation:  7 words
```

```
;    6380  1     routine EMS_SBC : novalue =
;    6381  1
;    6382  1     !+
;    6383  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SB_CODE" (SUB-CODE) IF
;    6384  1     !       EITHER THE STATUS CODE (ST_CODE) OR THE SUB-CODE IS NON-ZERO. (A
;    6385  1     !       NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
;    6386  1     !       HAS MEANING WITH A NON-ZERO STATUS CODE).
;    6387  1     !-
;    6388  1
;    6389  2         begin
;    6390  2
;    6391  2         if (.ST_CODE or .SB_CODE) neq 0                         ! PRINT SUB-CODE ONLY ON ERROR
;    6392  2         then
;    6393  3             begin
;    6394  3             PRINTX (EX_SB);                                     ! SUB-CODE :
;    6395  3
;    6396  3             case .ST_CODE from ST_SUC to ST_DRV of
;    6397  3                 set
;    6398  3
;    6399  3                 [ST_SUC]:       if .SB_CODE lequ 16             ! SUCCESS SUB-CODES
;    6400  3                                 then
;    6401  3                                     PRINTX (.TBL_SUC [.SB_CODE]);
;    6402  3
;    6403  3                 [ST_CMD]:       PRINTX (EX_SBO, .SB_CODE / 8);  ! INVALID COMMAND
;    6404  3
;    6405  3                 [ST_ABO]:       ;                               ! COMMAND ABORTED
;    6406  3
;    6407  3                 [ST_OFL]:       if .SB_CODE lequ 8              ! UNIT OFFLINE
;    6408  3                                 then
;    6409  3                                     PRINTX (.TBL_OFL [.SB_CODE]);
;    6410  3
;    6411  3                 [ST_AVL]:       ;                               ! UNIT AVAILABLE
;    6412  3
;    6413  3                 [ST_MFE]:       if .SB_CODE lequ 10             ! MEDIA FORMAT ERROR
;    6414  3                                 then
;    6415  3                                     PRINTX (.TBL_MFE [.SB_CODE]);
;    6416  3
;    6417  3                 [ST_WPT]:       if (.SB_CODE / 128) lequ 2      ! WRITE PROTECTED
;    6418  3                                 then
;    6419  3                                     PRINTX (.TBL_WPT [(.SB_CODE / 128)]);
;    6420  3
;    6421  3                 [ST_CMP]:       ;                               ! COMPARE ERROR
;    6422  3
;    6423  3                 [ST_DAT]:       if .SB_CODE lequ 15             ! DATA ERROR
;    6424  3                                 then
;    6425  3                                     PRINTX (.TBL_DAT [.SB_CODE]);
;    6426  3
;    6427  3                 [ST_HST]:       if .SB_CODE lequ 4              ! HOST ACCESS ERROR
;    6428  3                                 then
;    6429  3                                     PRINTX (.TBL_HST [.SB_CODE]);
;    6430  3
;    6431  3                 [ST_CNT]:       if .SB_CODE lequ 3              ! CONTROLLER ERROR
;    6432  3                                 then
```

```
;   6433  3                                   PRINTX (.TBL_CNT [.SB_CODE]);
;   6434  3
;   6435  3           [ST_DRV]:               if .SB_CODE lequ 8                  ! DRIVE ERROR
;   6436  3                                   then
;   6437  3                                       PRINTX (.TBL_DRV [.SB_CODE]);
;   6438  3
;   6439  3           [outrange]:             PRINTX (EX_SBO, .SB_CODE);          ! JUST PRINT SUB-CODE IF NO MATCH
;   6440  3           tes;
;   6441  3
;   6442  2       end;
;   6443  2
;   6444  1   end;


                                    .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000  013700  000000G      EMS.SBC:MOV     ST.CODE,R0                          ;                                   6391
000004  053700  000000G              BIS     SB.CODE,R0
000010  001001                       BNE     1$
000012  000207                       RTS     PC
000014  012746  000000G      1$:     MOV     #EX.SB,-(SP)                        ;                                   6394
000020  012746  000001               MOV     #1,-(SP)
000024  010600                       MOV     SP,R0                               ; SP,*
000026  104415                       TRAP    15
000030  013700  000000G              MOV     ST.CODE,R0                          ;                                   6396
000034  020027  000013               CMP     R0,#13
000040  101003                       BHI     3$
000042  006300                       ASL     R0
000044  066007  000000'              ADD     P.AAA(R0),PC                        ; Case dispatch
000050  013716  000000G      3$:     MOV     SB.CODE,(SP)                        ;                                   6439
000054  012746  000000G              MOV     #EX.SBO,-(SP)
000060  012746  000002               MOV     #2,-(SP)
000064  010600                       MOV     SP,R0                               ; SP,*
000066  104415                       TRAP    15
000070  022626                       CMP     (SP)+,(SP)+
000072  000435                       BR      6$                                  ;                                   6396
000074  023727  000000G 000020 4$:   CMP     SB.CODE,#20                         ;                                   6399
000102  101165                       BHI     14$
000104  013700  000000G              MOV     SB.CODE,R0                          ;                                   6401
000110  006300                       ASL     R0
000112  016016  000000'              MOV     TBL.SUC(R0),(SP)
000116  012746  000001               MOV     #1,-(SP)
000122  010600                       MOV     SP,R0                               ; SP,*
000124  104415                       TRAP    15
000126  000565                       BR      15$
000130  013716  000000G      5$:     MOV     SB.CODE,(SP)                        ;                                   6403
000134  012746  000010               MOV     #10,-(SP)
000140  004737  000000G              JSR     PC,BL$DIV
000144  010016                       MOV     R0,(SP)
000146  012746  000000G              MOV     #EX.SBO,-(SP)
000152  012746  000002               MOV     #2,-(SP)
000156  010600                       MOV     SP,R0                               ; SP,*
000160  104415                       TRAP    15
000162  062706  000006               ADD     #6,SP
```

```
000166  000546                          6$:     BR      16$                             ;                       6396
000170  023727  000000G 000010          7$:     CMP     SB.CODE,#10                     ;                       6407
000176  101142                                  BHI     16$
000200  013700  000000G                         MOV     SB.CODE,R0                      ;                       6409
000204  006300                                  ASL     R0
000206  016016  000042'                         MOV     TBL.OFL(R0),(SP)
000212  012746  000001                          MOV     #1,-(SP)
000216  010600                                  MOV     SP,R0                           ; SP,*
000220  104415                                  TRAP    15
000222  000527                                  BR      15$
000224  023727  000000G 000012          8$:     CMP     SB.CODE,#12                     ;                       6413
000232  101124                                  BHI     16$
000234  013700  000000G                         MOV     SB.CODE,R0                      ;                       6415
000240  006300                                  ASL     R0
000242  016016  000064'                         MOV     TBL.MFE(R0),(SP)
000246  012746  000001                          MOV     #1,-(SP)
000252  010600                                  MOV     SP,R0                           ; SP,*
000254  104415                                  TRAP    15
000256  000511                                  BR      15$
000260  013716  000000G                 9$:     MOV     SB.CODE,(SP)                    ;                       6417
000264  012746  000200                          MOV     #200,-(SP)
000270  004737  000000G                         JSR     PC,BL$DIV
000274  005726                                  TST     (SP)+
000276  020027  000002                          CMP     R0,#2
000302  101100                                  BHI     16$
000304  006300                                  ASL     R0                              ;                       6419
000306  016016  000112'                         MOV     TBL.WPT(R0),(SP)
000312  012746  000001                          MOV     #1,-(SP)
000316  010600                                  MOV     SP,R0                           ; SP,*
000320  104415                                  TRAP    15
000322  000467                                  BR      15$
000324  023727  000000G 000017          10$:    CMP     SB.CODE,#17                     ;                       6423
000332  101064                                  BHI     16$
000334  013700  000000G                         MOV     SB.CODE,R0                      ;                       6425
000340  006300                                  ASL     R0
000342  016016  000120'                         MOV     TBL.DAT(R0),(SP)
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                           ; SP,*
000354  104415                                  TRAP    15
000356  000451                                  BR      15$
000360  023727  000000G 000004          11$:    CMP     SB.CODE,#4                      ;                       6427
000366  101046                                  BHI     16$
000370  013700  000000G                         MOV     SB.CODE,R0                      ;                       6429
000374  006300                                  ASL     R0
000376  016016  000160'                         MOV     TBL.HST(R0),(SP)
000402  012746  000001                          MOV     #1,-(SP)
000406  010600                                  MOV     SP,R0                           ; SP,*
000410  104415                                  TRAP    15
000412  000433                                  BR      15$
000414  023727  000000G 000003          12$:    CMP     SB.CODE,#3                      ;                       6431
000422  101030                                  BHI     16$
000424  013700  000000G                         MOV     SB.CODE,R0                      ;                       6433
000430  006300                                  ASL     R0
```

```
000432  016016  000172'                        MOV     TBL.CNT(R0),(SP)
000436  012746  000001                        MOV     #1,-(SP)
000442  010600                                MOV     SP,R0                   ; SP,*
000444  104415                                TRAP    15
000446  000415                                BR      15$
000450  023727  C00000G 000010        13$:    CMP     SB.CODE,#10             ;                       6435
000456  101012                        14$:    BHI     16$
000460  013700  000000G                       MOV     SB.CODE,R0              ;                       6437
000464  006300                                ASL     R0
000466  016016  000202'                        MOV     TBL.DRV(R0),(SP)
000472  012746  000001                        MOV     #1,-(SP)
000476  010600                                MOV     SP,R0                   ; SP,*
000500  104415                                TRAP    15
000502  005726                        15$:    TST     (SP)+
000504  022626                        16$:    CMP     (SP)+,(SP)+             ;                       6393
000506  000207                                RTS     PC                      ;                       6380

; Routine Size:  164 words,    Routine Base:  $CODE$ + 10334
; Maximum stack depth per invocation:  7 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:                                          ; CASE Table for EMS.SBC+0044   6396
000000  000024                  2$:     .WORD   24                              ; [4$]
000002  000060                          .WORD   60                              ; [5$]
000004  000434                          .WORD   434                             ; [16$]
000006  000120                          .WORD   120                             ; [7$]
000010  000434                          .WORD   434                             ; [16$]
000012  000154                          .WORD   154                             ; [8$]
000014  000210                          .WORD   210                             ; [9$]
000016  000434                          .WORD   434                             ; [16$]
000020  000254                          .WORD   254                             ; [10$]
000022  000310                          .WORD   310                             ; [11$]
000024  000344                          .WORD   344                             ; [12$]
000026  000400                          .WORD   400                             ; [13$]
```

```
;    6445  1     routine EMS_CMD : novalue =
;    6446  1
;    6447  1     !+
;    6448  1     !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;    6449  1     !       PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;    6450  1     !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;    6451  1     !
;    6452  1     !       IMPLICIT INPUTS:
;    6453  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6454  1     !-
;    6455  1
;    6456  2         begin
;    6457  2         PRINTX (EX_CMD);                                        ! "COMMAND: "
;    6458  2
;    6459  2         selectoneu (.RP_ADDR [ENDCOD] and OP_MSK) of
;    6460  2             set
;    6461  2
;    6462  2             [OP_ONL]:       PRINTX (EX_ONL);                    ! ONLINE
;    6463  2
;    6464  2             [OP_ACC]:       PRINTX (EX_ACC);                    ! ACCESS
;    6465  2
;    6466  3             [OP_RD]:        begin
;    6467  3                             PRINTX (EX_RD);                     ! READ
;    6468  3
;    6469  3                             if .RP_ADDR [CMDMOD] neq 0
;    6470  3                             then
;    6471  3                                 PRINTX (EX_CMP);                ! COMPARE
;    6472  3
;    6473  2                             end;
;    6474  2
;    6475  3             [OP_WRT]:       begin
;    6476  3                             PRINTX (EX_WRT);                    ! WRITE
;    6477  3
;    6478  3                             if .RP_ADDR [CMDMOD] neq 0
;    6479  3                             then
;    6480  3                                 PRINTX (EX_CMP);                ! COMPARE
;    6481  3
;    6482  2                             end;
;    6483  2
;    6484  2             [otherwise]:    PRINTX (EX_OP, .RP_ADDR [ENDCOD]);  ! ENDCODE VALUE IF NO MATCH
;    6485  2             tes;
;    6486  2
;    6487  1         end;                                                    ! ROUTINE EMS_CMD


                                .SBTTL  EMS.CMD ERROR MESSAGE SUBROUTINES
011044                          .PSECT  $CODE$,  RO

000000  004137  000000G  EMS.CMD:JSR     R1,$SAVE2                       ;                       6445
000004  012746  000000G          MOV     #EX.CMD,-(SP)                   ;                       6457
000010  012746  000001           MOV     #1,-(SP)
000014  010600                   MOV     SP,R0                           ; SP,*
000016  104415                   TRAP    15
```

```
000020  013702  000000G                    MOV     RP.ADDR,R2                      ;                       6459
000024  116201  000014                      MOVB    14(R2),R1
000030  042701  177600                      BIC     #177600,R1
000034  020127  000011                      CMP     R1,#11                          ;                       6462
000040  001007                              BNE     1$
000042  012716  000000G                     MOV     #EX.ONL,(SP)
000046  012746  000001                      MOV     #1,-(SP)
000052  010600                              MOV     SP,R0                           ; SP,*
000054  104415                              TRAP    15
000056  000464                              BR      5$
000060  020127  000020          1$:         CMP     R1,#20                          ;                       6464
000064  001007                              BNE     2$
000066  012716  000000G                     MOV     #EX.ACC,(SP)
000072  012746  000001                      MOV     #1,-(SP)
000076  010600                              MOV     SP,R0                           ; SP,*
000100  104415                              TRAP    15
000102  000452                              BR      5$
000104  020127  000041          2$:         CMP     R1,#41                          ;                       6466
000110  001022                              BNE     3$
000112  012716  000000G                     MOV     #EX.RD,(SP)                     ;                       6467
000116  012746  000001                      MOV     #1,-(SP)
000122  010600                              MOV     SP,R0                           ; SP,*
000124  104415                              TRAP    15
000126  013700  000000G                     MOV     RP.ADDR,R0                      ;                       6469
000132  005760  000012                      TST     12(R0)
000136  001434                              BEQ     5$
000140  012716  000000G                     MOV     #EX.CMP,(SP)                    ;                       6471
000144  012746  000001                      MOV     #1,-(SP)
000150  010600                              MOV     SP,R0                           ; SP,*
000152  104415                              TRAP    15
000154  000424                              BR      4$
000156  020127  000042          3$:         CMP     R1,#42                          ;                       6475
000162  001024                              BNE     6$
000164  012716  000000G                     MOV     #EX.WRT,(SP)                    ;                       6476
000170  012746  000001                      MOV     #1,-(SP)
000174  010600                              MOV     SP,R0                           ; SP,*
000176  104415                              TRAP    15
000200  013700  000000G                     MOV     RP.ADDR,R0                      ;                       6478
000204  005760  000012                      TST     12(R0)
000210  001407                              BEQ     5$
000212  012716  000000G                     MOV     #EX.CMP,(SP)                    ;                       6480
000216  012746  000001                      MOV     #1,-(SP)
000222  010600                              MOV     SP,R0                           ; SP,*
000224  104415                              TRAP    15
000226  005726                  4$:         TST     (SP)+
000230  005726                  5$:         TST     (SP)+                           ;                       6475
000232  000412                              BR      7$                              ;                       6459
000234  005016                  6$:         CLR     (SP)                            ;                       6484
000236  116216  000014                      MOVB    14(R2),(SP)
000242  012746  000000G                     MOV     #EX.OP,-(SP)
000246  012746  000002                      MOV     #2,-(SP)
000252  010600                              MOV     SP,R0                           ; SP,*
000254  104415                              TRAP    15
```

```
000256  022626                          CMP     (SP)+,(SP)+
000260  022626                  7$:     CMP     (SP)+,(SP)+                     ;                       6456
000262  000207                          RTS     PC                              ;                       6445

; Routine Size:  90 words,      Routine Base:  $CODE$ + 11044
; Maximum stack depth per invocation:  9 words
```

```
;   6488  1     GLOBAL ROUTINE EMS_DBN : NOVALUE =                                  !ZZZ
;   6489  1     !+                                                                  !ZZZ
;   6490  1     !       THIS ROUTINE PRINTS THE PRESENT DBN                         !ZZZ
;   6491  1     !                                                                   !ZZZ
;   6492  1     !       IMPLICIT IMPUTS:                                            !ZZZ
;   6493  1     !               CST_ADDR - ADDRESS OF CONTROLLER STATUS TABLE       !ZZZ
;   6494  1     !-                                                                  !ZZZ
;   6495  1                                                                         !ZZZ
;   6496  2     BEGIN                                                               !ZZZ
;   6497  2     PRINTB (XX13, .CDISK);                      !"DISK XXX"             !ZZZ
; P 6498  2     PRINTB (XX23, .CST_ADDR [.CUOFF + OF_DBN, D_DBN], .CST_ADDR         !ZZZ
;   6499  2                           [.CUOFF + OF_DBN, D_DBN]);     !"DBN: XXXXXX." !ZZZ
;   6500  2     PRINTB (XX32, .S_DUPPKT - 2);               !PRINT BYTE COUNT       !ZZZ
;   6501  2     PRINTB (XX33, .S_PATTERN);                  !PRINT THE PATTERN      !ZZZ
;   6502  2     PRINTB (XX34, .(DUPPKT + .S_DUPPKT), .(DUPPKT + .S_DUPPKT));    !PRINT THE WORD READ    !ZZZ
;   6503  2     EMS_BLK (DUPPKT +2, 256);                   !PRINT WHOLE BLOCK READ !ZZZ
;   6504  1     END;                                        !IN OCTAL               !ZZZ


                                    .SBTTL  EMS.DBN ERROR MESSAGE SUBROUTINES
000000  013746  000000G       EMS.DBN::
                                    MOV     CDISK,-(SP)                     ;                                       6497
000004  012746  000000G             MOV     #XX13,-(SP)
000010  012746  000002              MOV     #2,-(SP)
000014  010600                      MOV     SP,R0                           ; SP,*
000016  104414                      TRAP    14
000020  013700  000000G             MOV     CUOFF,R0                        ;                                       6499
000024  006300                      ASL     R0
000026  063700  000000G             ADD     CST.ADDR,R0
000032  005016                      CLR     (SP)
000034  116016  000020              MOVB    20(R0),(SP)
000040  005046                      CLR     -(SP)
000042  116016  000020              MOVB    20(R0),(SP)
000046  012746  000000G             MOV     #XX23,-(SP)
000052  012746  000003              MOV     #3,-(SP)
000056  010600                      MOV     SP,R0                           ; SP,*
000060  104414                      TRAP    14
000062  013716  000000G             MOV     S.DUPPKT,(SP)                   ;                                       6500
000066  162716  000002              SUB     #2,(SP)
000072  012746  000000G             MOV     #XX32,-(SP)
000076  012746  000002              MOV     #2,-(SP)
000102  010600                      MOV     SP,R0                           ; SP,*
000104  104414                      TRAP    14
000106  013716  000000G             MOV     S.PATTERN,(SP)                  ;                                       6501
000112  012746  000000G             MOV     #XX33,-(SP)
000116  012746  000002              MOV     #2,-(SP)
000122  010600                      MOV     SP,R0                           ; SP,*
000124  104414                      TRAP    14
000126  013700  000000G             MOV     S.DUPPKT,R0                     ;                                       6502
000132  016016  000000G             MOV     DUPPKT(R0),(SP)
000136  011646                      MOV     (SP),-(SP)
000140  012746  000000G             MOV     #XX34,-(SP)
000144  012746  000003              MOV     #3,-(SP)
```

```
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104414                          TRAP    14
000154  012716  000002G                 MOV     #DUPPKT+2,(SP)                  ;                               6503
000160  012746  000400                  MOV     #400,-(SP)
000164  004737  000000V                 JSR     PC,EMS.BLK
000170  062706  000034                  ADD     #34,SP                          ;                               6496
000174  000207                          RTS     PC                              ;                               6488

; Routine Size:  63 words,      Routine Base:  $CODE$ + 11330
; Maximum stack depth per invocation:  15 words


;   6505  1
;   6506  1
```

```
;    6507  1
;    6508  1     GLOBAL ROUTINE EMS_BLK (ADDR, LENGTH) : NOVALUE =                        !ZZZ
;    6509  1                                                                             !ZZZ
;    6510  1     !+                                                                      !ZZZ
;    6511  1     !       THIS ROUTINE WILL PRINTX A BLOCK OF MEMORY, WHICH IS 'LENGTH'   !ZZZ
;    6512  1     !       WORDS LONG STARTING AT ADDRESS 'ADDR'. PRINTING IS DONE IN OCTAL!ZZZ
;    6513  1     !       8 WDS TO A LINE.                                                !ZZZ
;    6514  1     !-                                                                      !ZZZ
;    6515  1                                                                             !ZZZ
;    6516  2     BEGIN                                                                   !ZZZ
;    6517  2     LITERAL                                                                 !ZZZ
;    6518  2             MASK = %o'7';                                                   !ZZZ
;    6519  2                                                                             !ZZZ
;    6520  2     PRINTX (CRLF);                                                          !ZZZ
;    6521  2             INCR COUNT FROM 1 TO .LENGTH DO         !FOR EACH WD TO PRINT   !ZZZ
;    6522  3                     BEGIN                                                   !ZZZ
;    6523  3                     IF ((.COUNT - 1) AND MASK) EQL 0 !IF START OF NEW LINE  !ZZZ
;    6524  3                     THEN                                                    !ZZZ
;    6525  3                             PRINTX (SPACE4);        !PRINT 4 BLANKS         !ZZZ
;    6526  3                                                                             !ZZZ
;    6527  3                     PRINTX (EX_WRD, ..ADDR);        !PRINTX A WORD          !ZZZ
;    6528  3                     ADDR = .ADDR +2;                !TO NEXT ADDRESS        !ZZZ
;    6529  3                                                                             !ZZZ
;    6530  4                     IF (((.COUNT AND MASK) EQL 0) OR !END OF LINE OR        !ZZZ
;    6531  4                         (.COUNT EQL .LENGTH))       !WHEN DONE              !ZZZ
;    6532  3                     THEN                                                    !ZZZ
;    6533  3                             PRINTX (CRLF);          !PRINT CR LF            !ZZZ
;    6534  2                     END;                                                    !ZZZ
;    6535  1             END;                                                            !ZZZ


                                        .SBTTL  EMS.BLK ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.BLK::
                                        MOV     R1,-(SP)                ;               6508
000002  012746  000000G                 MOV     #CRLF,-(SP)             ;               6520
000006  012746  000001                  MOV     #1,-(SP)
000012  010600                          MOV     SP,R0                   ; SP,*
000014  104415                          TRAP    15
000016  005001                          CLR     R1                      ; COUNT         6521
000020  000445                          BR      5$
000022  010100                  1$:     MOV     R1,R0                   ; COUNT,*       6523
000024  005300                          DEC     R0
000026  032700  000007                  BIT     #7,R0
000032  001007                          BNE     2$
000034  012716  000000G                 MOV     #SPACE4,(SP)            ;               6525
000040  012746  000001                  MOV     #1,-(SP)
000044  010600                          MOV     SP,R0                   ; SP,*
000046  104415                          TRAP    15
000050  005726                          TST     (SP)+
000052  017616  000012          2$:     MOV     @12(SP),(SP)            ; ADDR,*        6527
000056  012746  000000G                 MOV     #EX.WRD,-(SP)
000062  012746  000002                  MOV     #2,-(SP)
000066  010600                          MOV     SP,R0                   ; SP,*
```

```
000070  104415                          TRAP    15
000072  062766  000002  000016          ADD     #2,16(SP)               ; *,ADDR                        6528
000100  032701  000007                  BIT     #7,R1                   ; *,COUNT                       6530
000104  001403                          BEQ     3$
000106  020166  000014                  CMP     R1,14(SP)               ; COUNT,LENGTH                  6531
000112  001007                          BNE     4$
000114  012716  000000G           3$:    MOV     #CRLF,(SP)              ;                               6533
000120  012746  000001                  MOV     #1,-(SP)
000124  010600                          MOV     SP,R0                   ; SP,*
000126  104415                          TRAP    15
000130  005726                          TST     (SP)+
000132  022626                   4$:    CMP     (SP)+,(SP)+             ;                               6522
000134  005201                   5$:    INC     R1                      ; COUNT                         6521
000136  020166  000010                  CMP     R1,10(SP)               ; COUNT,LENGTH
000142  003727                          BLE     1$
000144  022626                          CMP     (SP)+,(SP)+             ;                               6516
000146  012601                          MOV     (SP)+,R1                ;                               6508
000150  000207                          RTS     PC

; Routine Size:  53 words,      Routine Base:  $CODE$ + 11526
; Maximum stack depth per invocation:  8 words


;   6536  1
;   6537  1
```

```
;    6538  1     routine EMS_LBN : novalue =
;    6539  1
;    6540  1     !+
;    6541  1     !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;    6542  1     !       THE CURRENT RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;    6543  1     !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;    6544  1     !       PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE CURRENT RETURN PACKET
;    6545  1     !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;    6546  1     !       PRINTED.
;    6547  1     !
;    6548  1     !       IMPLICIT INPUTS:
;    6549  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6550  1     !-
;    6551  1
;    6552  2         begin
;    6553  2
;    6554  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and              ! IF NO BAD BLOCK FOUND
;    6555  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6556  2         then
;    6557  2             PRINTX (EX_LBN, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);
;    6558  2
;    6559  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and              ! IF BAD BLOCKS FOUND AND REPLACED
;    6560  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6561  2         then
;    6562  2             PRINTX (EX_BBU, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    6563  2
;    6564  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                  ! IF HOST REPLACEABLE BAD BLOCK FOUND
;    6565  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6566  2         then
;    6567  2             PRINTX (EX_BB, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    6568  2
;    6569  2         ;if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;    6570  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    6571  2         then
;    6572  2             PRINTX (EX_BB1, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;    6573  1         end;


                                        .SBTTL  EMS.LBN ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.LBN:MOV     RP.ADDR,R0                      ;                       6554
000004  105760  000015                  TSTB    15(R0)
000010  100417                          BMI     1$
000012  132760  000100  000015          BITB    #100,15(R0)                     ;                       6555
000020  001013                          BNE     1$
000022  016046  000050                  MOV     50(R0),-(SP)                    ;                       6557
000026  011646                          MOV     (SP),-(SP)
000030  012746  000000G                 MOV     #EX.LBN,-(SP)
000034  012746  000003                  MOV     #3,-(SP)
000040  010600                          MOV     SP,R0                           ; SP,*
000042  104415                          TRAP    15
000044  062706  000010                  ADD     #10,SP
000050  013700  000000G          1$:     MOV     RP.ADDR,R0                      ;                       6559
000054  105760  000015                  TSTB    15(R0)
```

```
000060  100417                          BMI     2$
000062  132760  000100  000015          BITB    #100,15(R0)                     ;                       6560
000070  001413                          BEQ     2$
000072  016046  000040                  MOV     40(R0),-(SP)                    ;                       6562
000076  011646                          MOV     (SP),-(SP)
000100  012746  000000G                 MOV     #EX.BBU,-(SP)
000104  012746  000003                  MOV     #3,-(SP)
000110  010600                          MOV     SP,R0                           ; SP,*
000112  104415                          TRAP    15
000114  062706  000010                  ADD     #10,SP
000120  013700  000000G         2$:     MOV     RP.ADDR,R0                      ;                       6564
000124  105760  000015                  TSTB    15(R0)
000130  100017                          BPL     3$
000132  132760  000100  000015          BITB    #100,15(R0)                     ;                       6565
000140  001013                          BNE     3$
000142  016046  000040                  MOV     40(R0),-(SP)                    ;                       6567
000146  011646                          MOV     (SP),-(SP)
000150  012746  000000G                 MOV     #EX.BB,-(SP)
000154  012746  000003                  MOV     #3,-(SP)
000160  010600                          MOV     SP,R0                           ; SP,*
000162  104415                          TRAP    15
000164  062706  000010                  ADD     #10,SP
000170  013700  000000G         3$:     MOV     RP.ADDR,R0                      ;                       6569
000174  105760  000015                  TSTB    15(R0)
000200  100017                          BPL     4$
000202  132760  000100  000015          BITB    #100,15(R0)                     ;                       6570
000210  001413                          BEQ     4$
000212  016046  000040                  MOV     40(R0),-(SP)                    ;                       6572
000216  011646                          MOV     (SP),-(SP)
000220  012746  000000G                 MOV     #EX.BB1,-(SP)
000224  012746  000003                  MOV     #3,-(SP)
000230  010600                          MOV     SP,R0                           ; SP,*
000232  104415                          TRAP    15
000234  062706  000010                  ADD     #10,SP
000240  000207                 4$:      RTS     PC                              ;                       6538

; Routine Size:  81 words,    Routine Base:  $CODE$ + 11700
; Maximum stack depth per invocation:  6 words
```

```
;   6574  1     routine EMS_BC : novalue =
;   6575  1
;   6576  1     !+
;   6577  1     !       THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE CURRENT
;   6578  1     !       RETURN PACKET: THE BYTE COUNT FROM THE COMMAND PACKET AND THE
;   6579  1     !       ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE PACKET).
;   6580  1     !
;   6581  1     !       IMPLICIT INPUTS:
;   6582  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;   6583  1     !-
;   6584  1
;   6585  2         begin
;   6586  2         PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]);       ! "BYTE COUNT IN COMMAND: XXXXX."
;   6587  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);         ! "ACTUAL # OF BYTES TRANSFERRED: XXXXX."
;   6588  1         end;                                        ! ROUTINE EMS_BC


                                .SBTTL  EMS.BC ERROR MESSAGE SUBROUTINES
000000  013700  000000G  EMS.BC: MOV     RP.ADDR,R0                      ;                       6586
000004  016046  000044           MOV     44(R0),-(SP)
000010  012746  000000G          MOV     #EX.CBC,-(SP)
000014  012746  000002           MOV     #2,-(SP)
000020  010600                   MOV     SP,R0                           ; SP,*
000022  104415                   TRAP    15
000024  013700  000000G          MOV     RP.ADDR,R0                      ;                       6587
000030  016016  000020           MOV     20(R0),(SP)
000034  012746  000000G          MOV     #EX.BC,-(SP)
000040  012746  000002           MOV     #2,-(SP)
000044  010600                   MOV     SP,R0                           ; SP,*
000046  104415                   TRAP    15
000050  062706  000012           ADD     #12,SP                          ;                       6585
000054  000207                   RTS     PC                              ;                       6574

; Routine Size:  23 words,      Routine Base:  $CODE$ + 12142
; Maximum stack depth per invocation:  7 words
```

```
;   6589  1     routine EMS_BD : novalue =
;   6590  1
;   6591  1     !+
;   6592  1     !       THIS ROUTINE PRINTS (EXTENDED) THE TWO-WORD I/O BUFFER DESCRIPTOR
;   6593  1     !       APPEARING IN THE CURRENT RETURN PACKET.
;   6594  1     !
;   6595  1     !       IMPLICIT INPUTS:
;   6596  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;   6597  1     !-
;   6598  1
;   6599  1         PRINTX (EX_BD, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);       ! "I/O BUFFER DESCRIPTOR: XXXXXX XXXXXX"


                                        .SBTTL  EMS.BD ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.BD: MOV     RP.ADDR,R0                      ;                                       6599
000004  016046  000024                  MOV     24(R0),-(SP)
000010  016046  000026                  MOV     26(R0),-(SP)
000014  012746  000000G                  MOV     #EX.BD,-(SP)
000020  012746  000003                  MOV     #3,-(SP)
000024  010600                          MOV     SP,R0                           ; SP,*
000026  104415                          TRAP    15
000030  062706  000010                  ADD     #10,SP
000034  000207                          RTS     PC                              ;                                       6589

; Routine Size:  15 words,      Routine Base:  $CODE$ + 12220
; Maximum stack depth per invocation:  6 words
```

```
;    6600  1
;    6601  1
;    6602  1     routine EMS_RP : novalue =
;    6603  1
;    6604  1
;    6605  1     !+
;    6606  1     !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;    6607  1     !       OF THE CURRENT RETURN PACKET.
;    6608  1     !-
;    6609  1
;    6610  1
;    6611  2         begin
;    6612  2         EMS_SBC ();                                     ! SUB-CODE
;    6613  2         EMS_CMD ();                                     ! COMMAND (AND MODIFIER)
;    6614  2
;    6615  2         if (.RP_ADDR [ENDCOD] and OP_MSK) neq OP_ONL
;    6616  2
;    6617  2         then
;    6618  2             EMS_LBN ();                                 ! LBN OR BAD BLOCK NUMBER
;    6619  2
;    6620  2         if ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD) or
;    6621  3            ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_WRT)
;    6622  3
;    6623  2         then
;    6624  3             begin
;    6625  3             EMS_BC ();                                  ! BYTE COUNTS
;    6626  3             EMS_BD ();                                  ! I/O BUFFER DESCRIPTOR
;    6627  2             end;
;    6628  2
;    6629  2         EMS_TIM ();                                     ! TIME
;    6630  1         end;                                            ! ROUTINE EMS_RP


                                .SBTTL  EMS.RP ERROR MESSAGE SUBROUTINES
000000  010146                 EMS.RP: MOV     R1,-(SP)                        ;  6602
000002  004737  010334'                 JSR     PC,EMS.SBC                      ;  6612
000006  004737  011044'                 JSR     PC,EMS.CMD                      ;  6613
000012  013700  000000G                 MOV     RP.ADDR,R0                      ;  6615
000016  116000  000014                  MOVB    14(R0),R0
000022  042700  177600                  BIC     #177600,R0
000026  020027  000011                  CMP     R0,#11
000032  001402                          BEQ     1$
000034  004737  011700'                 JSR     PC,EMS.LBN                      ;  6618
000040  013700  000000G         1$:     MOV     RP.ADDR,R0                      ;  6620
000044  116001  000014                  MOVB    14(R0),R1
000050  042701  177600                  BIC     #177600,R1
000054  020127  000041                  CMP     R1,#41
000060  001407                          BEQ     2$
000062  116000  000014                  MOVB    14(R0),R0                       ;  6621
000066  042700  177600                  BIC     #177600,R0
000072  020027  000042                  CMP     R0,#42
000076  001004                          BNE     3$
000100  004737  012142'         2$:     JSR     PC,EMS.BC                       ;  6625
```

```
000104  004737  012220'                        JSR     PC,EMS.BD                       ;                       6626
000110  004737  000000V                3$:     JSR     PC,EMS.TIM                      ;                       6629
000114  012601                                  MOV     (SP)+,R1                        ;                       6602
000116  000207                                  RTS     PC

; Routine Size:  40 words,      Routine Base:  $CODE$ + 12256
; Maximum stack depth per invocation:  2 words


;   6631  1
;   6632  1
```

```
;   6633  1     global routine EMS_RP1 : novalue =
;   6634  1
;   6635  1     !+
;   6636  1     !       THIS  ROUTINE  IS  CALLED  TO  PRINT  THE  ENTIRE CONTENTS OF THE
;   6637  1     !       RETURN PACKET DESIGNATED BY THE GLOBAL DATUM "RP_ADDR". HOWEVER, THE
;   6638  1     !       PRINTING WILL ONLY OCCUR IF EXTENDED ERROR PRINTING IS ENABLED.
;   6639  1     !-
;   6640  1
;   6641  2         begin
;   6642  2         PRINTX (EX_RP);                         ! "CONTENTS OF RETURN PACKET:"
;   6643  2         EMS_BLK (.RP_ADDR, RP_LEN);             ! PRINT BLOCK OF WORDS
;   6644  1         end;


                                        .SBTTL  EMS.RP1 ERROR MESSAGE SUBROUTINES
000000  012746  000000G            EMS.RP1::
                                        MOV     #EX.RP,-(SP)                    ;                       6642
000004  012746  000001                  MOV     #1,-(SP)
000010  010600                          MOV     SP,R0                           ; SP,*
000012  104415                          TRAP    15
000014  013716  000000G                 MOV     RP.ADDR,(SP)                    ;                       6643
000020  012746  000026                  MOV     #26,-(SP)
000024  004737  011526'                 JSR     PC,EMS.BLK
000030  062706  000006                  ADD     #6,SP                           ;                       6644
000034  000207                          RTS     PC                              ;                       6633

; Routine Size:  15 words,      Routine Base:  $CODE$ + 12376
; Maximum stack depth per invocation:  4 words
```

```
;    6645  1     global routine EMS_EL (index) : novalue =
;    6646  1
;    6647  1     !+
;    6648  1     !       THIS ROUTINE IS CALLED FROM 'SEQUEN' AND 'DATAGM' AND PRINTS THE CONTENTS OF THE
;    6649  1     !       ERROR-LOG PACKET
;    6650  1     !-
;    6651  1
;    6652  2         begin
;    6653  2
;    6654  2         local
;    6655  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;    6656  2             REASON : word,
;    6657  2             DISK_NUM : byte,
;    6658  2             ELOG_CODE : byte,
;    6659  2             ELOG_SUB : word;
;    6660  2
;    6661  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                  ! ERROR LOG PACKET'S ADDRESS
;    6662  2         REASON = .ELOG_ADDR [EL_FORMAT];                               ! FORMAT
;    6663  2         DISK_NUM = .ELOG_ADDR [EL_DK_NUM];                             ! DISK NUMBER
;    6664  2         ELOG_CODE = .ELOG_ADDR [EL_CODE];                              ! CODE
;    6665  2         ELOG_SUB = .ELOG_ADDR [EL_SUBCODE];                            ! SUBCODE
;    6666  2         PRINTB (ELG_00);                                               ! ERROR-LOG MESSAGE RECEIVED
;    6667  2
;    6668  2         if (.REASON eql FORMAT_CNTR) or
;    6669  3            (.REASON eql FORMAT_HOST)
;    6670  2         then
;    6671  3             PRINTB (.ELG_FMT [.REASON])                                ! PRINT BASIC REASON
;    6672  2         else
;    6673  2             PRINTB (.ELG_FMT [.REASON], .DISK_NUM);                    ! PRINT BASIC REASON WITH DISK NUMBER
;    6674  2
;    6675  2         if (.ELOG_CODE gtru 0) and
;    6676  3            (.ELOG_CODE lequ 11)
;    6677  2         then
;    6678  3             begin
;    6679  3             PRINTX (ASTERISK);
;    6680  3             PRINTX (.ERR_COD [.ELOG_CODE - 1]);                        ! CODE
;    6681  3             end
;    6682  2         else
;    6683  2
;    6684  2             if .ELOG_CODE eql ST_DIA                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    6685  2             then
;    6686  3                 begin
;    6687  3                 PRINTX (ASTERISK);
;    6688  3                 PRINTX (.ERR_COD [12]);
;    6689  2                 end;
;    6690  2
;    6691  2         if (.ELOG_CODE eql ST_MFE) and
;    6692  3            (.ELOG_SUB lequ 10)
;    6693  2         then
;    6694  3             begin
;    6695  3             PRINTX (CRLF);
;    6696  3             PRINTX (ASTERISK);
;    6697  3             PRINTX (.TBL_MFE [.ELOG_SUB]);                             ! MEDIA FORMAT ERROR
```

```
;   6698  2          end;
;   6699  2
;   6700  2      if (.ELOG_CODE eql ST_DAT) and
;   6701  3         (.ELOG_SUB lequ 15)
;   6702  2      then
;   6703  3          begin
;   6704  3          PRINTX (CRLF);
;   6705  3          PRINTX (ASTERISK);
;   6706  3          PRINTX (.TBL_DAT [.ELOG_SUB]);                          ! DATA ERROR
;   6707  2          end;
;   6708  2
;   6709  2      if (.ELOG_CODE eql ST_HST) and
;   6710  3         (.ELOG_SUB lequ 4)
;   6711  2      then
;   6712  3          begin
;   6713  3          PRINTX (CRLF);
;   6714  3          PRINTX (ASTERISK);
;   6715  3          PRINTX (.TBL_HST [.ELOG_SUB]);                          ! HOST ACCESS ERROR
;   6716  2          end;
;   6717  2
;   6718  2      if (.ELOG_CODE eql ST_CNT) and
;   6719  3         (.ELOG_SUB lequ 3)
;   6720  2      then
;   6721  3          begin
;   6722  3          PRINTX (CRLF);
;   6723  3          PRINTX (ASTERISK);
;   6724  3          PRINTX (.TBL_CNT [.ELOG_SUB]);                          ! CONTROLLER ERROR
;   6725  2          end;
;   6726  2
;   6727  2      if (.ELOG_CODE eql ST_DRV) and
;   6728  3         (.ELOG_SUB lequ 8)
;   6729  2      then
;   6730  3          begin
;   6731  3          PRINTX (CRLF);
;   6732  3          PRINTX (ASTERISK);
;   6733  3          PRINTX (.TBL_DRV [.ELOG_SUB]);                          ! DRIVE ERROR
;   6734  2          end;
;   6735  2
;   6736  2      if .REASON eql FORMAT_XFER                                  ! IF DISK XFER INVOLVED
;   6737  2      then
;   6738  2
;   6739  2          if .ELOG_ADDR [EL_BLOCK_TYPE] eql TYPE_LBN              ! PRINT PBN OR RBN
;   6740  2          then
;   6741  3              PRINTX (EX_PBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK])
;   6742  2          else
;   6743  2              PRINTX (EX_RBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK]);
;   6744  2
;   6745  2      EMS_TIM ();                                                 ! TIME
;   6746  2      EMS_BLK ((.ELOG_ADDR + 2), ((.ELOG_ADDR [EL_MSGLEN] + 1) / 2) + 2); ! PRINTX CONTENTS OF PACKET
;   6747  2      ELOG_ADDR [EL_CONTENTS] = EMPTY;                            ! DECLARE SAVE AREA FREE
;   6748  2
;   6749  1      end;
```

```
                                        .SBTTL  EMS.EL ERROR MESSAGE SUBROUTINES
000000  004137  000000G           EMS.EL::JSR     R1,$SAVE5                       ;                                       6645
000004  005746                            TST     -(SP)
000006  016646  000020                    MOV     20(SP),-(SP)                    ; INDEX,*                               6661
000012  012746  000102                    MOV     #102,-(SP)
000016  004737  000000G                   JSR     PC,BL$MUL
000022  062700  000000G                   ADD     #ELOG.PKT,R0
000026  010001                            MOV     R0,R1                           ; *,ELOG.ADDR
000030  116166  000016  000004            MOVB    16(R1),4(SP)                    ; *(ELOG.ADDR),REASON                   6662
000036  105066  000005                    CLRB    5(SP)                           ; REASON
000042  116105  000012                    MOVB    12(R1),R5                       ; *(ELOG.ADDR),DISK.NUM                 6663
000046  116100  000020                    MOVB    20(R1),R0                       ; *(ELOG.ADDR),*                        6664
000052  042700  177740                    BIC     #177740,R0
000056  105004                            CLRB    R4                              ; ELOG.CODE
000060  050004                            BIS     R0,R4                           ; *,ELOG.CODE
000062  016103  000020                    MOV     20(R1),R3                       ; *(ELOG.ADDR),ELOG.SUB                 6665
000066  006203                            ASR     R3                              ; ELOG.SUB
000070  006203                            ASR     R3                              ; ELOG.SUB
000072  006203                            ASR     R3                              ; ELOG.SUB
000074  006203                            ASR     R3                              ; ELOG.SUB
000076  006203                            ASR     R3                              ; ELOG.SUB
000100  042703  174000                    BIC     #174000,R3                      ; *,ELOG.SUB
000104  012716  000000G                   MOV     #ELG.00,(SP)                    ;                                       6666
000110  012746  000001                    MOV     #1,-(SP)
000114  010600                            MOV     SP,R0                           ; SP,*
000116  104414                            TRAP    14
000120  016602  000006                    MOV     6(SP),R2                        ; REASON,*                              6671
000124  006302                            ASL     R2
000126  005766  000006                    TST     6(SP)                           ; REASON                                6668
000132  001404                            BEQ     1$
000134  026627  000006  000001            CMP     6(SP),#1                        ; REASON,*                              6669
000142  001007                            BNE     2$
000144  016216  000000G           1$:     MOV     ELG.FMT(R2),(SP)                ;                                       6671
000150  012746  000001                    MOV     #1,-(SP)
000154  010600                            MOV     SP,R0                           ; SP,*
000156  104414                            TRAP    14
000160  000411                            BR      3$                              ;                                       6668
000162  005016                    2$:     CLR     (SP)                            ;                                       6673
000164  110516                            MOVB    R5,(SP)                         ; DISK.NUM,*
000166  016246  000000G                   MOV     ELG.FMT(R2),-(SP)
000172  012746  000002                    MOV     #2,-(SP)
000176  010600                            MOV     SP,R0                           ; SP,*
000200  104414                            TRAP    14
000202  005726                            TST     (SP)+
000204  105704                    3$:     TSTB    R4                              ; ELOG.CODE                             6675
000206  001423                            BEQ     4$
000210  120427  000013                    CMPB    R4,#13                          ; ELOG.CODE,*                           6676
000214  101020                            BHI     4$
000216  012716  000000G                   MOV     #ASTERISK,(SP)                  ;                                       6679
000222  012746  000001                    MOV     #1,-(SP)
000226  010600                            MOV     SP,R0                           ; SP,*
000230  104415                            TRAP    15
```

```
000232  005000                          CLR     R0                              ;                               6680
000234  150400                          BISB    R4,R0                           ; ELOG.CODE,*
000236  006300                          ASL     R0
000240  016016  177776G                 MOV     ERR.COD-2(R0),(SP)
000244  012746  000001                  MOV     #1,-(SP)
000250  010600                          MOV     SP,R0                           ; SP,*
000252  104415                          TRAP    15
000254  000417                          BR      5$                              ;                               6678
000256  120427  000037          4$:     CMPB    R4,#37                          ; ELOG.CODE,*                   6684
000262  001015                          BNE     6$
000264  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6687
000270  012746  000001                  MOV     #1,-(SP)
000274  010600                          MOV     SP,R0                           ; SP,*
000276  104415                          TRAP    15
000300  013716  000030G                 MOV     ERR.COD+30,(SP)                 ;                               6688
000304  012746  000001                  MOV     #1,-(SP)
000310  010600                          MOV     SP,R0                           ; SP,*
000312  104415                          TRAP    15
000314  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               6686
000316  120427  000005          6$:     CMPB    R4,#5                           ; ELOG.CODE,*                   6691
000322  001031                          BNE     7$
000324  020327  000012                  CMP     R3,#12                          ; ELOG.SUB,*                    6692
000330  101026                          BHI     7$
000332  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6695
000336  012746  000001                  MOV     #1,-(SP)
000342  010600                          MOV     SP,R0                           ; SP,*
000344  104415                          TRAP    15
000346  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6696
000352  012746  000001                  MOV     #1,-(SP)
000356  010600                          MOV     SP,R0                           ; SP,*
000360  104415                          TRAP    15
000362  010300                          MOV     R3,R0                           ; ELOG.SUB,*                    6697
000364  006300                          ASL     R0
000366  016016  000064'                 MOV     TBL.MFE(R0),(SP)
000372  012746  000001                  MOV     #1,-(SP)
000376  010600                          MOV     SP,R0                           ; SP,*
000400  104415                          TRAP    15
000402  062706  000006                  ADD     #6,SP                           ;                               6694
000406  120427  000010          7$:     CMPB    R4,#10                          ; ELOG.CODE,*                   6700
000412  001031                          BNE     8$
000414  020327  000017                  CMP     R3,#17                          ; ELOG.SUB,*                    6701
000420  101026                          BHI     8$
000422  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6704
000426  012746  000001                  MOV     #1,-(SP)
000432  010600                          MOV     SP,R0                           ; SP,*
000434  104415                          TRAP    15
000436  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6705
000442  012746  000001                  MOV     #1,-(SP)
000446  010600                          MOV     SP,R0                           ; SP,*
000450  104415                          TRAP    15
000452  010300                          MOV     R3,R0                           ; ELOG.SUB,*                    6706
000454  006300                          ASL     R0
000456  016016  000120'                 MOV     TBL.DAT(R0),(SP)
```

```
000462  012746  000001                  MOV     #1,-(SP)
000466  010600                          MOV     SP,R0                   ; SP,*
000470  104415                          TRAP    15
000472  062706  000006                  ADD     #6,SP                   ;                       6703
000476  120427  000011          8$:     CMPB    R4,#11                  ; ELOG.CODE,*           6709
000502  001031                          BNE     9$
000504  020327  000004                  CMP     R3,#4                   ; ELOG.SUB,*            6710
000510  101026                          BHI     9$
000512  012716  000000G                 MOV     #CRLF,(SP)              ;                       6713
000516  012746  000001                  MOV     #1,-(SP)
000522  010600                          MOV     SP,R0                   ; SP,*
000524  104415                          TRAP    15
000526  012716  000000G                 MOV     #ASTERISK,(SP)          ;                       6714
000532  012746  000001                  MOV     #1,-(SP)
000536  010600                          MOV     SP,R0                   ; SP,*
000540  104415                          TRAP    15
000542  010300                          MOV     R3,R0                   ; ELOG.SUB,*            6715
000544  006300                          ASL     R0
000546  016016  000160'                 MOV     TBL.HST(R0),(SP)
000552  012746  000001                  MOV     #1,-(SP)
000556  010600                          MOV     SP,R0                   ; SP,*
000560  104415                          TRAP    15
000562  062706  000006                  ADD     #6,SP                   ;                       6712
000566  120427  000012          9$:     CMPB    R4,#12                  ; ELOG.CODE,*           6718
000572  001031                          BNE     10$
000574  020327  000003                  CMP     R3,#3                   ; ELOG.SUB,*            6719
000600  101026                          BHI     10$
000602  012716  000000G                 MOV     #CRLF,(SP)              ;                       6722
000606  012746  000001                  MOV     #1,-(SP)
000612  010600                          MOV     SP,R0                   ; SP,*
000614  104415                          TRAP    15
000616  012716  000000G                 MOV     #ASTERISK,(SP)          ;                       6723
000622  012746  000001                  MOV     #1,-(SP)
000626  010600                          MOV     SP,R0                   ; SP,*
000630  104415                          TRAP    15
000632  010300                          MOV     R3,R0                   ; ELOG.SUB,*            6724
000634  006300                          ASL     R0
000636  016016  000172'                 MOV     TBL.CNT(R0),(SP)
000642  012746  000001                  MOV     #1,-(SP)
000646  010600                          MOV     SP,R0                   ; SP,*
000650  104415                          TRAP    15
000652  062706  000006                  ADD     #6,SP                   ;                       6721
000656  120427  000013          10$:    CMPB    R4,#13                  ; ELOG.CODE,*           6727
000662  001031                          BNE     11$
000664  020327  000010                  CMP     R3,#10                  ; ELOG.SUB,*            6728
000670  101026                          BHI     11$
000672  012716  000000G                 MOV     #CRLF,(SP)              ;                       6731
000676  012746  000001                  MOV     #1,-(SP)
000702  010600                          MOV     SP,R0                   ; SP,*
000704  104415                          TRAP    15
000706  012716  000000G                 MOV     #ASTERISK,(SP)          ;                       6732
000712  012746  000001                  MOV     #1,-(SP)
000716  010600                          MOV     SP,R0                   ; SP,*
```

```
000720  104415                          TRAP    15
000722  010300                          MOV     R3,R0                   ; ELOG.SUB,*                    6733
000724  006300                          ASL     R0
000726  016016  000202'                 MOV     TBL.DRV(R0),(SP)
000732  012746  000001                  MOV     #1,-(SP)
000736  010600                          MOV     SP,R0                   ; SP,*
000740  104415                          TRAP    15
000742  062706  000006                  ADD     #6,SP                   ;                               6730
000746  026627  000010  000002  11$:    CMP     10(SP),#2               ; REASON,*                      6736
000754  001031                          BNE     14$
000756  032761  170000  000060          BIT     #170000,60(R1)          ; *,*(ELOG.ADDR)                6739
000764  001012                          BNE     12$
000766  016116  000056                  MOV     56(R1),(SP)             ; *(ELOG.ADDR),*                6741
000772  011646                          MOV     (SP),-(SP)
000774  012746  000000G                 MOV     #EX.PBN,-(SP)
001000  012746  000003                  MOV     #3,-(SP)
001004  010600                          MOV     SP,R0                   ; SP,*
001006  104415                          TRAP    15
001010  000411                          BR      13$                     ;                               6739
001012  016116  000056          12$:    MOV     56(R1),(SP)             ; *(ELOG.ADDR),*                6743
001016  011646                          MOV     (SP),-(SP)
001020  012746  000000G                 MOV     #EX.RBN,-(SP)
001024  012746  000003                  MOV     #3,-(SP)
001030  010600                          MOV     SP,R0                   ; SP,*
001032  104415                          TRAP    15
001034  062706  000006          13$:    ADD     #6,SP                   ;                               6739
001040  004737  000000V         14$:    JSR     PC,EMS.TIM              ;                               6745
001044  012716  000002                  MOV     #2,(SP)                 ;                               6746
001050  060116                          ADD     R1,(SP)                 ; ELOG.ADDR,*
001052  016146  000002                  MOV     2(R1),-(SP)             ; *(ELOG.ADDR),*
001056  005216                          INC     (SP)
001060  012746  000002                  MOV     #2,-(SP)
001064  004737  000000G                 JSR     PC,BL$DIV
001070  010066  000002                  MOV     R0,2(SP)
001074  062766  000002  000002          ADD     #2,2(SP)
001102  005726                          TST     (SP)+
001104  004737  011526'                 JSR     PC,EMS.BLK
001110  105061  000001                  CLRB    1(R1)                   ; *(ELOG.ADDR)                  6747
001114  062706  000014                  ADD     #14,SP                  ;                               6645
001120  000207                          RTS     PC

; Routine Size:  297 words,    Routine Base:  $CODE$ + 12434
; Maximum stack depth per invocation:  16 words
```

```
;    6750  1     global routine EMS_CMP (ADDR) : novalue =
;    6751  1
;    6752  1     !+
;    6753  1     ! THIS ROUTINE IS CALLED FROM 'HOST_WRT_CHK' AND PRINTS RELEVENT DATA ON A HOST
;    6754  1     ! COMPARE ERROR
;    6755  1     !-
;    6756  1
;    6757  2         begin
;    6758  2
;    6759  2         local
;    6760  2             ORIG_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    6761  2
;    6762  2         ORIG_ADDR = .ADDR;                                          ! ADDRESS OF THE WRITE RETPKT
;    6763  2         PRINTB (ERR_00, .CDISK);                                    ! "DISK XXX"
;    6764  2         PRINTB (DASH);                                              !
;    6765  2         PRINTB (.ERR_COD [12]);                                     ! " - HOST COMPARE ERROR"
;    6766  2         PRINTX (EX_LBW, .ORIG_ADDR [LBN_LO], .ORIG_ADDR [LBN_LO]);  ! LBN (WRITE)
;    6767  2         PRINTX (EX_LBR, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);      ! LBN (READ)
;    6768  2         PRINTX (EX_CBW, .ORIG_ADDR [CBCNT_LO]);                     ! BYTE COUNT (WRITE)
;    6769  2         PRINTX (EX_BC, .ORIG_ADDR [BCNT_LO]);                       ! BYTE COUNT XMITTED (WRITE)
;    6770  2         PRINTX (EX_CBR, .RP_ADDR [CBCNT_LO]);                       ! BYTE COUNT (READ);
;    6771  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);                         ! BYTE COUNT XMITTED (READ)
;    6772  2         PRINTX (EX_BDW, .ORIG_ADDR [BUFF_1], .ORIG_ADDR [BUFF_0]);  ! BUFFER ADDRESS (WRITE)
;    6773  2         PRINTX (EX_BDR, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! BUFFER ADDRESS (READ)
;    6774  2         EMS_TIM ();                                                 ! TIME
;    6775  1         end;


                                .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  010146          EMS.CMP::
                                MOV     R1,-(SP)                            ;                                   6750
000002  016601  000004          MOV     4(SP),R1                            ; ADDR,ORIG.ADDR                    6762
000006  013746  000000G         MOV     CDISK,-(SP)                         ;                                   6763
000012  012746  000000G         MOV     #ERR.00,-(SP)
000016  012746  000002          MOV     #2,-(SP)
000022  010600                  MOV     SP,R0                               ; SP,*
000024  104414                  TRAP    14
000026  012716  000000G         MOV     #DASH,(SP)                          ;                                   6764
000032  012746  000001          MOV     #1,-(SP)
000036  010600                  MOV     SP,R0                               ; SP,*
000040  104414                  TRAP    14
000042  013716  000030G         MOV     ERR.COD+30,(SP)                     ;                                   6765
000046  012746  000001          MOV     #1,-(SP)
000052  010600                  MOV     SP,R0                               ; SP,*
000054  104414                  TRAP    14
000056  016116  000050          MOV     50(R1),(SP)                         ; *(ORIG.ADDR),*                    6766
000062  011646                  MOV     (SP),-(SP)
000064  012746  000000G         MOV     #EX.LBW,-(SP)
000070  012746  000003          MOV     #3,-(SP)
000074  010600                  MOV     SP,R0                               ; SP,*
000076  104415                  TRAP    15
000100  013700  000000G         MOV     RP.ADDR,R0                          ;                                   6767
000104  016016  000050          MOV     50(R0),(SP)
```

```
000110  011646                          MOV     (SP),-(SP)
000112  012746  000000G                 MOV     @EX.LBR,-(SP)
000116  012746  000003                  MOV     #3,-(SP)
000122  010600                          MOV     SP,R0                   ; SP,*
000124  104415                          TRAP    15
000126  016116  000044                  MOV     44(R1),(SP)             ; *(ORIG.ADDR),*                6768
000132  012746  000000G                 MOV     @EX.CBW,-(SP)
000136  012746  000002                  MOV     #2,-(SP)
000142  010600                          MOV     SP,R0                   ; SP,*
000144  104415                          TRAP    15
000146  016116  000020                  MOV     20(R1),(SP)             ; *(ORIG.ADDR),*                6769
000152  012746  000000G                 MOV     @EX.BC,-(SP)
000156  012746  000002                  MOV     #2,-(SP)
000162  010600                          MOV     SP,R0                   ; SP,*
000164  104415                          TRAP    15
000166  013700  000000G                 MOV     RP.ADDR,R0              ;                               6770
000172  016016  000044                  MOV     44(R0),(SP)
000176  012746  000000G                 MOV     @EX.CBR,-(SP)
000202  012746  000002                  MOV     #2,-(SP)
000206  010600                          MOV     SP,R0                   ; SP,*
000210  104415                          TRAP    15
000212  013700  000000G                 MOV     RP.ADDR,R0              ;                               6771
000216  016016  000020                  MOV     20(R0),(SP)
000222  012746  000000G                 MOV     @EX.BC,-(SP)
000226  012746  000002                  MOV     #2,-(SP)
000232  010600                          MOV     SP,R0                   ; SP,*
000234  104415                          TRAP    15
000236  016116  000024                  MOV     24(R1),(SP)             ; *(ORIG.ADDR),*                6772
000242  016146  000026                  MOV     26(R1),-(SP)            ; *(ORIG.ADDR),*
000246  012746  000000G                 MOV     @EX.BDW,-(SP)
000252  012746  000003                  MOV     #3,-(SP)
000256  010600                          MOV     SP,R0                   ; SP,*
000260  104415                          TRAP    15
000262  013700  000000G                 MOV     RP.ADDR,R0              ;                               6773
000266  016016  000024                  MOV     24(R0),(SP)
000272  016046  000026                  MOV     26(R0),-(SP)
000276  012746  000000G                 MOV     @EX.BDR,-(SP)
000302  012746  000003                  MOV     #3,-(SP)
000306  010600                          MOV     SP,R0                   ; SP,*
000310  104415                          TRAP    15
000312  004737  000000V                 JSR     PC,EMS.TIM              ;                               6774
000316  062706  000062                  ADD     #62,SP                  ;                               6757
000322  012601                          MOV     (SP)+,R1                ;                               6750
000324  000207                          RTS     PC

; Routine Size:  107 words,    Routine Base:  $CODE$ + 13556
; Maximum stack depth per invocation:  28 words
```

```
;   6776  1     global routine EMS_ERR : novalue =
;   6777  1
;   6778  2         begin
;   6779  2
;   6780  2     ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;   6781  2     !
;   6782  2         PRINTB (ERR_00, .CDISK);                        ! "DISK XXX"
;   6783  2         PRINTB (DASH);                                  !
;   6784  2
;   6785  2         if (.ST_CODE gtru 0) and                        ! IF STATUS CODE IS WITHIN RANGE
;   6786  3            (.ST_CODE lequ 11)
;   6787  2         then
;   6788  3             PRINTB (.ERR_COD [.ST_CODE - 1])            ! PRINTB APPROPRIATE MESSAGE
;   6789  2         else
;   6790  2
;   6791  2             if .ST_CODE eql ST_DIA
;   6792  2             then
;   6793  3                 PRINTB (.ERR_COD [11])                  ! MESSAGE FROM INTERNAL DIAGNOSTICS
;   6794  2             else
;   6795  2                 PRINTB (EX_SC, .ST_CODE);               ! JUST PRINT STATUS CODE WHEN NO MATCH
;   6796  2
;   6797  2         EMS_RP ();                                      ! PRINTX OTHER RETPKT FIELDS
;   6798  2
;   6799  1         end;


                                .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G    EMS.ERR::
                            MOV     CDISK,-(SP)                 ;                   6782
000004  012746  000000G            MOV     #ERR.00,-(SP)
000010  012746  000002             MOV     #2,-(SP)
000014  010600                     MOV     SP,R0                       ; SP,*
000016  104414                     TRAP    14
000020  012716  000000G            MOV     #DASH,(SP)                  ;           6783
000024  012746  000001             MOV     #1,-(SP)
000030  010600                     MOV     SP,R0                       ; SP,*
000032  104414                     TRAP    14
000034  013700  000000G            MOV     ST.CODE,R0                  ;           6785
000040  001413                     BEQ     1$
000042  020027  000013             CMP     R0,#13                      ;           6786
000046  101010                     BHI     1$
000050  006300                     ASL     R0                          ;           6788
000052  016016  177776G            MOV     ERR.COD-2(R0),(SP)
000056  012746  000001             MOV     #1,-(SP)
000062  010600                     MOV     SP,R0                       ; SP,*
000064  104414                     TRAP    14
000066  000422                     BR      3$                          ;           6785
000070  020027  000037     1$:     CMP     R0,#37                      ;           6791
000074  001007                     BNE     2$
000076  013716  000026G            MOV     ERR.COD+26,(SP)             ;           6793
000102  012746  000001             MOV     #1,-(SP)
000106  010600                     MOV     SP,R0                       ; SP,*
000110  104414                     TRAP    14
```

```
000112  000410                          BR      3$                              ;                               6791
000114  010016                  2$:     MOV     R0,(SP)                         ;                               6795
000116  012746  000000G                 MOV     #EX.SC,-(SP)
000122  012746  000002                  MOV     #2,-(SP)
000126  010600                          MOV     SP,R0                           ; SP,*
000130  104414                          TRAP    14
000132  005726                          TST     (SP)+
000134  004737  012256'          3$:     JSR     PC,EMS.RP                       ;                               6797
000140  062706  000012                  ADD     #12,SP                          ;                               6778
000144  000207                          RTS     PC                              ;                               6776

; Routine Size:  51 words,      Routine Base:  $CODE$ + 14104
; Maximum stack depth per invocation:  8 words
```

```
;   6800  1     routine EMS_TIM : novalue =
;   6801  1
;   6802  1     !+
;   6803  1     ! THIS ROUTINE PRINTS THE TIME-OF-DAY MESSAGE
;   6804  1     !-
;   6805  1
;   6806  1     PRINTX (EX_TIM, .HOURS, .MINUTES);


                                        .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                  EMS.TIM:CLR     -(SP)                           ;                       6806
000002  113716  000000G                 MOVB    MINUTES,(SP)
000006  005046                          CLR     -(SP)
000010  113716  000000G                 MOVB    HOURS,(SP)
000014  012746  000000G                 MOV     #EX.TIM,-(SP)
000020  012746  000003                  MOV     #3,-(SP)
000024  010600                          MOV     SP,R0                           ; SP,*
000026  104415                          TRAP    15
000030  062706  000010                  ADD     #10,SP
000034  000207                          RTS     PC                              ;                       6800

; Routine Size:  15 words,      Routine Base:  $CODE$ + 14252
; Maximum stack depth per invocation:  6 words
```

```
;   6807  1     BGNMSG (EMS_01);


                                        .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.01::JSR     PC,M$EMS.01                     ;                       6807
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14310
; Maximum stack depth per invocation:  2 words


;   6808  2     PRINTB (EBS_01, MAX_UNITS);                     ! "MORE THAN XX UNITS SPECIFIED"
;   6809  1     ENDMSG;


                                        .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  012746  000004                M$EMS.01:
                                        MOV     #4,-(SP)                        ;                       6808
000004  012746  000000G                         MOV     #EBS.01,-(SP)
000010  012746  000002                          MOV     #2,-(SP)
000014  010600                                  MOV     SP,R0                           ; SP,*
000016  104414                                  TRAP    14
000020  062706  000006                          ADD     #6,SP                           ;                       6807
000024  000207                                  RTS     PC

; Routine Size:  11 words,      Routine Base:  $CODE$ + 14320
; Maximum stack depth per invocation:  5 words
```

```
;   6810  1     BGNMSG (EMS_10);


                                        .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.10::JSR     PC,M$EMS.10                     ;                       6810
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14346
; Maximum stack depth per invocation:  2 words


;   6811  2     PRINTB (EBD_10, .RDRX_ADDR + .OF_RC);           ! "NO RESPONSE AT ADDRESS XXXXXX"
;   6812  1     ENDMSG;


                                        .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G         M$EMS.10:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       6811
000004  063716  000000G                 ADD     OF.RC,(SP)
000010  012746  000000G                 MOV     #EBD.10,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104414                          TRAP    14
000024  062706  000006                  ADD     #6,SP                           ;                       6810
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14356
; Maximum stack depth per invocation:  5 words
```

```
;   6813  1     BGNMSG (EMS_12);


                                        .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.12::JSR     PC,M$EMS.12                     ;                       6813
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14410
; Maximum stack depth per invocation:  2 words


;   6814  2     PRINTB (EBD_12, .RDRX_ADDR);                ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX"
;   6815  1     ENDMSG;


                                        .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.12:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       6814
000004  012746  000000G                 MOV     #EBD.12,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  062706  000006                  ADD     #6,SP                           ;                       6813
000024  000207                          RTS     PC

; Routine Size:  11 words,     Routine Base:  $CODE$ + 14420
; Maximum stack depth per invocation:  5 words
```

```
;   6816  1     BGNMSG (EMS_13);


                                        .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V               EMS.13::JSR     PC,M$EMS.13                     ;                       6816
000004  104423                              TRAP    23
000006  000207                              RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14446
; Maximum stack depth per invocation:  2 words


;   6817  2     PRINTB (EBD_13, .STEP);                 ! "STEP X READ ERROR"
;   6818  2     EMS_SA ();                              ! PRINTX SA CONTENTS
;   6819  1     ENDMSG;


                                        .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G               M$EMS.13:
                                        MOV     STEP,-(SP)                      ;                       6817
000004  012746  000000G                       MOV     #EBD.13,-(SP)
000010  012746  000002                        MOV     #2,-(SP)
000014  010600                                MOV     SP,R0                           ; SP,*
000016  104414                                TRAP    14
000020  004737  010022'                       JSR     PC,EMS.SA                       ;                       6818
000024  062706  000006                        ADD     #6,SP                           ;                       6816
000030  000207                                RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14456
; Maximum stack depth per invocation:  5 words
```

```
;   6820  1     BGNMSG (EMS_14);


                                        .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.14::JSR     PC,M$EMS.14                     ;                       6820
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14510
; Maximum stack depth per invocation:  2 words


;   6821  2     PRINTB (EBD_14, .IRDRX_ADDR);                   ! "BAD SA CODE FROM DEVICE XXXXXX"
;   6822  2     EMS_SA ();                                      ! PRINTX SA REGISTER CONTENTS
;   6823  1     ENDMSG;


                                        .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.14:
                                        MOV     IRDRX.ADDR,-(SP)                ;                       6821
000004  012746  000000G                 MOV     #EBD.14,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  010022'                 JSR     PC,EMS.SA                       ;                       6822
000024  062706  000006                  ADD     #6,SP                           ;                       6820
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14520
; Maximum stack depth per invocation:  5 words
```

```
;   6824  1     BGNMSG (EMS_18);


                                        .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.18::JSR     PC,M$EMS.18                     ;                       6824
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14552
; Maximum stack depth per invocation:  2 words


;   6825  2     PRINTB (EBD_18, .CDISK);                ! "DISK XXX WENT OFFLINE"
;   6826  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   6827  1     ENDMSG;


                                        .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.18:
                                        MOV     CDISK,-(SP)                     ;                       6825
000004  012746  000000G                 MOV     #EBD.18,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  012256'                 JSR     PC,EMS.RP                       ;                       6826
000024  062706  000006                  ADD     #6,SP                           ;                       6824
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14562
; Maximum stack depth per invocation:  5 words
```

```
;   6828  1     BGNMSG (EMS_21);


                                        .SBTTL  EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.21::JSR     PC,M$EMS.21                     ;       6828
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14614
; Maximum stack depth per invocation:  2 words


;   6829  2     EMS_RP1 ();                             ! CONTENTS OF RETURN PACKET
;   6830  1     ENDMSG;


                                        .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  012376'             M$EMS.21:
                                        JSR     PC,EMS.RP1                      ;       6829
000004  000207                            RTS     PC                              ;       6828

; Routine Size:  3 words,      Routine Base:  $CODE$ + 14624
; Maximum stack depth per invocation:  1 word
```

```
;   6831  1     BGNMSG (EMS_22)                         !CONTENTS OF DUP BUFFER          ZZZ


                                          .SBTTL  EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.22::JSR     PC,M$EMS.22                      ;       6831
000004  104423                                  TRAP    23
000006  000207                                  RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14632
; Maximum stack depth per invocation:  2 words


;   6832  2     EMS_DBN ();                                                          !ZZZ
;   6833  1     ENDMSG;                                                              !ZZZ


                                          .SBTTL  M$EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  011330'               M$EMS.22:
                                          JSR     PC,EMS.DBN                       ;       6832
000004  000207                                  RTS     PC                               ;       6831

; Routine Size:  3 words,      Routine Base:  $CODE$ + 14642
; Maximum stack depth per invocation:  1 word
```

```
;   6834  1     BGNMSG (EMS_24);


                                        .SBTTL  EMS.24 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.24::JSR     PC,M$EMS.24                     ;                       6834
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14650
; Maximum stack depth per invocation:  2 words


;   6835  2     PRINTB (EBD_24, .CDISK);                ! "DISK XXX WENT TO THE AVAILABLE STATE"
;   6836  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   6837  1     ENDMSG;


                                        .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.24:
                                        MOV     CDISK,-(SP)                     ;                       6835
000004  012746  000000G                 MOV     @EBD.24,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  012256'                 JSR     PC,EMS.RP                       ;                       6836
000024  062706  000006                  ADD     #6,SP                           ;                       6834
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14660
; Maximum stack depth per invocation:  5 words
```

```
;   6838  1     BGNMSG (EMS_30);


                                        .SBTTL  EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.30::JSR     PC,M$EMS.30                     ;                       6838
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14712
; Maximum stack depth per invocation:  2 words


;   6839  2     EMS_ERR ();                                     ! PRINT ALL RELEVANT DATA ON DETECTING AN ERROR
;   6840  1     ENDMSG;


                                        .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  014104'            M$EMS.30:
                                        JSR     PC,EMS.ERR                      ;                       6839
000004  000207                            RTS     PC                              ;                       6838

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14722
; Maximum stack depth per invocation:  1 word


;   6841  1
;   6842  1     end
;   6843  1
;   6844  0     eludom


;                                   OTS external references
                                            .GLOBL  $SAVE5, $SAVE4, $SAVE3, $SAVE2
                                            .GLOBL  BL$DIV, BL$MOD, BL$MUL


;                                   PSECT SUMMARY
;
;       Psect Name                  Words     Attributes
;         $OWN$                       74       RW  ,  D  ,  LCL,  REL,  CON
;         $CODE$                    3308       RO  ,  I  ,  LCL,  REL,  CON
;         $PLIT$                      12       RO  ,  D  ,  LCL,  REL,  CON


;                               Library Statistics
;
;                                       -------- Symbols --------     Pages       Processing
;       File                            Total    Loaded   Percent     Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAE0.L16;18       405       297        73         21       00:00.1
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAE0.BL1/LIST=ZRQAE0.LS1/OBJECT=ZRQAE0.OB1/SOURCE=PAGE:53

; Size:          3138 code + 6657 data words
; Run Time:        02:48.7
; Elapsed Time:    03:55.8
; Lines/CPU Min:    2434
; Lexemes/CPU-Min: 24408
; Memory Used:  699 pages
; Compilation Complete
```

```
;    0001  0
;    0002  0     module ZRQAM3 (
;    0003  0
;    0004  0     %title 'RD/RX EXERCISER'
;    0005  0                     ident = 'V01.8',
;    0006  0                     addressing_mode (absolute),
;    0007  0                     environment (noeis)
;    0008  0                     ) =
;    0009  0
;    0010  1     begin
;    0011  1
;    0012  1     %sbttl 'DECLARATIONS'
;    0013  1
;    0014  1     library 'ZRQAE0.L16';                   ! RDRX EXERCISER GLOBAL LIBRARY
;    0015  1
;    0016  1     require 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1507  1
;    1508  1     EQUALS;
;    1509  1
;    1510  1     forward routine                         ! ROUTINES APPEAR IN THIS ORDER
;    1511  1         INIT_TEST : novalue,                !    INDENTATION IMPLIES CALLED SUBROUTINE
;    1512  1             DRIVER_INIT : novalue,
;    1513  1                 CTLR_INIT : novalue,
;    1514  1                     INI_CTLR_DAT : novalue,
;    1515  1                     REG_EXIST,
;    1516  1                     VEC_BR_TEST,
;    1517  1                     INT_GEN,
;    1518  1                     HARD_INIT,
;    1519  1                     INI_RRING : novalue,
;    1520  1                     SET_CTLR_CHAR,
;    1521  1                 UNIT_INIT : novalue,
;    1522  1                     DR_ERR : novalue,
;    1523  1             ACCESS : novalue,
;    1524  1         MULTI_DRIVE : novalue,
;    1525  1             MD_INIT : novalue,
;    1526  1                 INIT_IO_BUFF : novalue,
;    1527  1             FATAL_ERROR : novalue,
;    1528  1             QIO_OK,
;    1529  1                 QIO_OUT,
;    1530  1                 QIO_GEN : novalue,
;    1531  1                     GET_RANDOM : novalue,
;    1532  1                     QIO_UNIT : novalue,
;    1533  1                     QIO_FUNC : novalue,
;    1534  1                        DUP : NOVALUE,                          !ZZZ
;    1535  1                            DUPWRTDBN : NOVALUE,                !ZZZ
;    1536  1                            DUPREDDBN : NOVALUE,                !ZZZ
;    1537  1                                DUPCOMMAND : NOVALUE,           !ZZZ
;    1538  1                                DUPIDLE : NOVALUE,              !ZZZ
;    1539  1                     QIO_LBN : novalue,
;    1540  1                     QIO_SIZE : novalue,
;    1541  1                     FILL_BUFF : novalue,
;    1542  1                 PROC_RETPKT : novalue,
;    1543  1                     DIO_RETPKT : NOVALUE,                       !ZZZ
```

```
;    1544  1                        DUP_COMPARE : NOVALUE,                                  !ZZZ
;    1545  1                    IO_RETPKT : novalue,
;    1546  1                        FSET_UPAR : novalue,
;    1547  1                        HARD_ERROR : novalue,
;    1548  1                            ERR_HRD_RTNE : novalue,
;    1549  1                            ERR_HRD_RTNE_APT : novalue,
;    1550  1                        UPD_IO_TALLY : novalue,
;    1551  1                            OVF_CHK : novalue,
;    1552  1                            ROUND_OUTPUT : novalue,
;    1553  1                        HOST_WRT_CHK,
;    1554  1    !                       ERR_HRD_RTNE : novalue,
;    1555  1    !                       ERR_HRD_RTNE_APT : novalue,
;    1556  1                        SWEEP : novalue,
;    1557  1                            RPS_REM,
;    1558  1                    DR_RETPKT : novalue,
;    1559  1            AZINTO : L$ISR novalue,
;    1560  1                AZINT : novalue,
;    1561  1    !               FATAL_ERROR : novalue,
;    1562  1                    POLL_CRING : novalue,
;    1563  1                    POLL_RRING : novalue,
;    1564  1                        DUP_RSP  : NOVALUE,
 !ZZZ
;    1565  1                    DISK_RSP : novalue,
;    1566  1                        SEQUEN : novalue,
;    1567  1                            SCAN_ERRLOG : novalue,
;    1568  1                                ERR_SOFT_RTNE : novalue,
;    1569  1                                ERR_SOFT_RTNE_APT : novalue,
;    1570  1                                SOFT_ERROR : novalue,
;    1571  1                        DATAGM : novalue;
;    1572  1    !                       ERR_SOFT_RTNE : novalue,
;    1573  1    !                       ERR_SOFT_RTNE_APT : novalue,
;    1574  1    !                       SOFT_ERROR : novalue;
;    1575  1
;    1576  1        external
;    1577  1            CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    1578  1                            ! RUN-TIME CONTROLLER STATUS TABLES
;    1579  1            CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1580  1                            ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    1581  1            DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    1582  1                            ! DRIVER CONTROLLER TABLES
;    1583  1            DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1584  1                            ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    1585  1            RDRX_ADDR : ref rdrx field (RC_REG),
;    1586  1                            ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    1587  1            IRDRX_ADDR : ref rdrx field (RC_REG),
;    1588  1                            ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    1589  1            BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                             !ZZZ
;    1590  1                            !BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS    !ZZZ
;    1591  1                            !RANDOM SEEK) MODE                              !ZZZ
;    1592  1            TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    1593  1                            ! STATISTICS TABLES
;    1594  1            T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    1595  1                            ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    1596  1            DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
```

```
;  1597  1                          !INFO FROM RECEIVE AND SEND COMMANDS                ZZZ
;  1598  1          TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TK DIRECTION    ZZZ
;  1599  1          RDM_CNT : WORD,     !NO. OF RANDOM NOS.              KEEP\\          ZZZ
;  1600  1          RANDOM : VECTOR [RDM_LEN, WORD],    !RAND NO TABLE TOGET//HER       ZZZ
;  1601  1          C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;  1602  1                          ! STATISTICS TABLE FOR CONTROLLER ERRORS
;  1603  1          MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;  1604  1                          ! MSCP PACKET POOL
;  1605  1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  1606  1                          ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;  1607  1          PKT_USE : vector [PKT_CNT, byte, signed],
;  1608  1                          ! MSCP PACKET POOL ALLOCATION TABLE
;  1609  1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  1610  1                          ! RETURN PACKET POOL
;  1611  1          RP_USE : vector [RP_CNT, byte, signed],
;  1612  1                          ! RETURN PACKET POOL ALLOCATION TABLE
;  1613  1          RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;  1614  1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;  1615  1                          ! CURRENT RETURN PACKET ADDRESS
;  1616  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;  1617  1                          ! ERROR-LOG PACKET SAVE AREA
;  1618  1          BUFF_ADDR : vector [MAX_BUF_CNT],                  ! TABLE OF I/O BUFFER DESCRIPTORS
;  1619  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],     ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;  1620  1          IODQ : vector [IODQ_LEN, byte],                    ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;  1621  1          IODQ_IN : word,                                    ! I/O DONE QUEUE IN POINTER
;  1622  1          IODQ_OUT : word,                                   ! I/O DONE QUEUE OUT POINTER
;  1623  1          ENTRY_REASON : byte,                               ! CURRENT OPERATOR COMMAND
;  1624  1          EOP_FLAG : byte,                                   ! END-OF-PASS FLAG
;  1625  1          DUP_FLAGS : WORD,                                  !DUP FLAGS                  ZZZ
;  1626  1          CCTLR : word,                                      ! NUMBER OF "CURRENT" CONTROLLER
;  1627  1          CDISK : word,                                      ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  1628  1          CUOFF : word,                                      ! CURRENT UNIT CST OFFSET
;  1629  1          CTLR_CNT : word,                                   ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;  1630  1          DUR : vector [MAX_UNITS, byte],                    ! DROP UNIT REASON
;  1631  1          QIO : vector [MAX_CTLR, byte],                     ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;  1632  1          FREE_MEM_ADDR,                                     ! START OF FREE MEMORY
;  1633  1          BYTS_PER_QIO : word,                               ! SIZE (BYTES) OF AN I/O BUFFER
;  1634  1          ST_CODE : word,                                    ! CURRENT STATUS CODE
;  1635  1          SB_CODE : word,                                    ! CURRENT SUB-CODE
;  1636  1          STEP : word,                                       ! CURRENT STEP IN HARD_INIT
;  1637  1          OF_RC : signed word,                               ! OFFSET (0 OR 2) TO READ IP OR SA
;  1638  1          SA_REG : word,                                     ! STORAGE FOR SA REGISTER READS AND WRITES
;  1639  1          CMD_TIME : word,                                   ! COMMAND TIMEOUT VALUE (IN SECONDS)
;  1640  1          NEX : word,                                        ! NON-EXISTENT MEMORY TRAP INDICATOR
;  1641  1          CRN_LOW : word,                                    ! COMMAND REF NUMBER OF LAST COMMAND SENT
;  1642  1          CRN_HIGH : word,                                   ! COMMAND REF NUMBER (HI ORDER)
;  1643  1          CREDIT_BAL : word,                                 ! CREDIT BALANCE
;  1644  1          NEXT_PKT_USE : byte,                               ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;  1645  1          HOURS : byte,                                      ! TIME OF DAY (HOURS)
;  1646  1          MINUTES : byte,                                    ! TIME OF DAY (MINUTES)
;  1647  1          CLK_TICKS : word,                                  ! TIME OF DAY (LINE-CLOCK TICKS)
;  1648  1          CLK_PRESENT : byte,                                ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  1649  1          HOE_FLAG : byte,                                   ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
```

```
;   1650  1        FORCED_ERROR : byte,                          ! "FORCED ERROR" DETECTED IN LAST READ
;   1651  1        FER_LBN : word,                               ! LBN OF THE "FORCED ERROR" BLOCK
;   1652  1        FER_BC : word,                                ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;   1653  1        INIT_OCCURED : byte,                          ! EXERCISER INITIALIZATION COMPLETE
;   1654  1        ADDR_VECT_OK : byte,                          ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;   1655  1        S_PATTERN : WORD,                 !PATTERN WRITTEN TO DBNS          ZZZ
;   1656  1        S_DUPPKT : WORD,                  !DBN BYTE COUNTER                 ZZZ
;   1657  1        P_INDEX : SIGNED WORD,            !CURRENT MESSAGE PACKET INDEX     ZZZ
;   1658  1        DBM12,
;   1659  1        DBM18,
;   1660  1        DBM19,
;   1661  1        DBM20,
;   1662  1        DBM21,
;   1663  1        DBM22,
;   1664  1        DBM23,
;   1665  1        DBM25,
;   1666  1        DBM26,
;   1667  1        DBM27,
;   1668  1        DBM29,
;   1669  1        DBM108,
;   1670  1        DBM109,
;   1671  1        DBM111,
;   1672  1        DBM112,
;   1673  1        DBM120,
;   1674  1        DBM121,
;   1675  1        EH_0,                                                   !ZZZ
;   1676  1        EH_1,                                                   !ZZZ
;   1677  1        EH_2,                                                   !ZZZ
;   1678  1        EH_3,                                                   !ZZZ
;   1679  1        EH_4,                                                   !ZZZ
;   1680  1        EH_5,                                                   !ZZZ
;   1681  1        EH_6,                                                   !ZZZ
;   1682  1        EH_7,                                                   !ZZZ
;   1683  1        EH_8,                                                   !ZZZ
;   1684  1        EH_9,                                                   !ZZZ
;   1685  1        EH_10,                                                  !ZZZ
;   1686  1        EH_12,                                                  !ZZZ
;   1687  1        EH_13,                                                  !ZZZ
;   1688  1        MSG_02,
;   1689  1        MSG_03,
;   1690  1        EGS_02,
;   1691  1        EGD_10,
;   1692  1        EGD_11,
;   1693  1        EGD_12,
;   1694  1        EGD_13,
;   1695  1        EGD_14,
;   1696  1        EGD_15,
;   1697  1        EGD_16,
;   1698  1        EGD_17,
;   1699  1        EGD_18,
;   1700  1        EGD_19,
;   1701  1        EGD_20,
;   1702  1        EGD_21,
```

```
;    1703  1          EGD_22,
;    1704  1          EGD_23,
;    1705  1          EGD_24,
;    1706  1          EGH_30,
;    1707  1          DF_MSG,
;    1708  1          HRD_MSG,
;    1709  1          SFT_MSG,
;    1710  1          HRD_SUB,
;    1711  1          CRLF,
;    1712  1          SWP_ERROR : word,                          ! HARD ERROR LIMIT FOR DROPPING UNIT
;    1713  1          SWP_XFER : word,                           ! TRANSFER LIMIT FOR DROPPING UNIT
;    1714  1          SWP_FLAGS : word,                          ! FLAGS (SEE DOCUMENTATION)
;    1715  1          DUPROUND  : WORD,                          !DUP TESTING RATIO                    ZZZ
;    1716  1          SWP_RAT : word,                            ! RD51/52 OPERATION RATIO
;    1717  1          SWP_DPAT : word,                           ! DATA PATTERN NUMBER
;    1718  1          SWP_UCNT : word,                           ! USER DATA PATTERN COUNT
;    1719  1          SWP_TIME : word,                           ! TIME OF DAY
;    1720  1          SWP_UDPAT : vector [MAX_UDP_CNT, word],    ! USER DATA PATTERN
;    1721  1          L$LUN,
;    1722  1          L$UNIT;
;    1723  1
;    1724  1      psect
;    1725  1          own = $GGG$(read, nowrite, execute, local, concatenate);
;    1726  1
;    1727  1      own
;    1728  1          COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
;    1729  1                              ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
;    1730  1      !!ZZZ    BST : vector [MAX_UNITS, word, signed],
;    1731  1                              ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
;    1732  1          DPST : vector [MAX_UNITS, byte],           ! DATA PATTERN SEQUENCE TABLE
;    1733  1          MAX_LBN : vector [MAX_UNITS, word],        ! LARGEST LBN ALLOWED
;    1734  1          STORAGE : vector [MAX_UNITS, word],        ! DUMMY STORAGE
;    1735  1          ICOM_ADDR : ref block [COMM_LEN, word] field (COM_FIELDS),
;    1736  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S COMMUNICATION AREA
;    1737  1          ICST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1738  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    1739  1          IDCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1740  1                              ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    1741  1          INT_ADDR : vector [MAX_CTLR] initial (AZINTO %(, AZINT1, AZINT2, AZINT3)%),
;    1742  1                              ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
;    1743  1      !!ZZZ    RDM_CNT : word initial (RDM_LEN),          ! NUMBER OF RANDOM NUMBERS  \ KEEP
;    1744  1      !!ZZZ    RANDOM : vector [RDM_LEN, word],           ! RANDOM NUMBER TABLE       / TOGETHER
;    1745  1          ICTLR : word,                              ! INTERRUPTING CONTROLLING NUMBER
;    1746  1          RW_BALANCE : WORD INITIAL (3),             ! FLAGS TOO MANY READS IN RD/WR RATIO      ZZZ
;    1747  1          RD_COUNT : WORD INITIAL (0),               ! NUMBER OF WINCHESTER UNITS               ZZZ
;    1748  1          MX1 : signed word,                         ! MSCP PKT INDEX FOR FIRST QIO
;    1749  1          MX2 : signed word,                         ! MSCP PKT INDEX FOR SECOND QIO
;    1750  1          MAD1 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1751  1                              ! ADDRESS OF MSCP PACKET FOR FIRST QIO
;    1752  1          MAD2 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1753  1                              ! ADDRESS OF MSCP PACKET FOR SECOND QIO
;    1754  1          LAST_PKT : blockvector [MAX_CTLR, LAST_PKT_LEN, word] field (LAST_PKT_FIELDS),
;    1755  1                              ! SAVE AREA FOR INFO ABOUT LAST RESPONSE PACKET
```

```
;  1756  1          PAT02 : vector [2] initial (1,                          ! PATTERN 2
;  1757  1                  %o'000000'),
;  1758  1          PAT03 : vector [2] initial (1,                          ! PATTERN 3
;  1759  1                  %o'177777'),
;  1760  1          PAT04 : vector [2] initial (1,                          ! PATTERN 4
;  1761  1                  %o'105613'),
;  1762  1          PAT05 : vector [2] initial (1,                          ! PATTERN 5
;  1763  1                  %o'031463'),
;  1764  1          PAT06 : vector [2] initial (1,                          ! PATTERN 6
;  1765  1                  %o'030221'),
;  1766  1          PAT07 : vector [17] initial (16,                        ! PATTERN 7
;  1767  1                  %o'000001', %o'000003', %o'000007', %o'000017',
;  1768  1                  %o'000037', %o'000077', %o'000177', %o'000377',
;  1769  1                  %o'000777', %o'001777', %o'003777', %o'007777',
;  1770  1                  %o'017777', %o'037777', %o'077777', %o'177777'),
;  1771  1          PAT08 : vector [17] initial (16,                        ! PATTERN 8
;  1772  1                  %o'177776', %o'177774', %o'177770', %o'177760',
;  1773  1                  %o'177740', %o'177700', %o'177600', %o'177400',
;  1774  1                  %o'177000', %o'176000', %o'174000', %o'170000',
;  1775  1                  %o'160000', %o'140000', %o'100000', %o'000000'),
;  1776  1          PAT09 : vector [17] initial (16,                        ! PATTERN 9
;  1777  1                  rep 3 of (%o'000000'), rep 3 of (%o'177777'),
;  1778  1                  rep 2 of (%o'000000'), rep 2 of (%o'177777'),
;  1779  1                  %o'000000', %o'177777', %o'000000', %o'177777',
;  1780  1                  %o'000000', %o'177777'),
;  1781  1          PAT10 : vector [2] initial (1,                          ! PATTERN 10
;  1782  1                  %o'133331'),
;  1783  1          PAT11 : vector [17] initial (16,                        ! PATTERN 11
;  1784  1                  rep 3 of (%o'052525'), rep 3 of (%o'125252'),
;  1785  1                  rep 2 of (%o'052525'), rep 2 of (%o'125252'),
;  1786  1                  %o'052525', %o'125252', %o'052525', %o'125252',
;  1787  1                  %o'052525', %o'125252'),
;  1788  1          PAT12 : vector [21] initial (20,                        ! PATTERN 12
;  1789  1                  rep 3 of (%o'026455'), rep 3 of (%o'151322'),
;  1790  1                  rep 2 of (%o'026455'), rep 2 of (%o'151322'),
;  1791  1                  rep 2 of (%o'026455'),
;  1792  1                  %o'151322', %o'026455', %o'151322', %o'026455',
;  1793  1                  %o'151322', %o'026455', %o'151322', %o'026455'),
;  1794  1          PAT13 : vector [2] initial (1,                          ! PATTERN 13
;  1795  1                  %o'066666'),
;  1796  1          PAT14 : vector [17] initial (16,                        ! PATTERN 14
;  1797  1                  %o'000001', %o'000002', %o'000004', %o'000010',
;  1798  1                  %o'000020', %o'000040', %o'000100', %o'000200',
;  1799  1                  %o'000400', %o'001000', %o'002000', %o'004000',
;  1800  1                  %o'010000', %o'020000', %o'040000', %o'100000'),
;  1801  1          PAT15 : vector [17] initial (16,                        ! PATTERN 15
;  1802  1                  %o'177776', %o'177775', %o'177773', %o'177767',
;  1803  1                  %o'177757', %o'177737', %o'177677', %o'177577',
;  1804  1                  %o'177377', %o'176777', %o'175777', %o'173777',
;  1805  1                  %o'167777', %o'157777', %o'137777', %o'077777'),
;  1806  1          PAT16 : vector [17] initial (16,                        ! PATTERN 16
;  1807  1                  rep 3 of (%o'133331'), rep 3 of (%o'155554'),
;  1808  1                  rep 2 of (%o'133331'), rep 2 of (%o'155554'),
```

```
;    1809  1                %o'133331', %o'155554', %o'133331', %o'155554',
;    1810  1                %o'133331', %o'155554'),
;    1811  1            PAT17 : vector [22] initial (21,                        ! PATTERN 17
;    1812  1                %o'000000', rep 2 of (%o'106466'),
;    1813  1                rep 3 of (%o'071311'), rep 4 of (%o'106466'),
;    1814  1                rep 5 of (%o'071311'), rep 6 of (%o'106466')),
;    1815  1            PAT18 : vector [22] initial (21,                        ! PATTERN 18
;    1816  1                %o'106466', %o'000000', %o'071311',
;    1817  1                rep 3 of (%o'106466'), rep 4 of (%o'071311'),
;    1818  1                rep 5 of (%o'106466'), rep 6 of (%o'071311')),
;    1819  1            PAT19 : vector [22] initial (21,                        ! PATTERN 19
;    1820  1                %o'000000', rep 2 of (%o'134631'),
;    1821  1                rep 3 of (%o'043146'), rep 4 of (%o'134631'),
;    1822  1                rep 5 of (%o'043146'), rep 6 of (%o'134631')),
;    1823  1            PAT20 : vector [22] initial (21,                        ! PATTERN 20
;    1824  1                %o'134631', %o'000000', %o'043146',
;    1825  1                rep 3 of (%o'134631'), rep 4 of (%o'043146'),
;    1826  1                rep 5 of (%o'134631'), rep 6 of (%o'043146')),
;    1827  1            PAT21 : vector [2] initial (1,                          ! PATTERN 21
;    1828  1                %o'000000'),                                        !    (LBN)
;    1829  1            DPA_TBL : vector [DP_CNT] initial                       ! DATA PATTERN ADDRESS TABLE
;    1830  1                (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
;    1831  1                PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
;    1832  1                PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
;    1833  1                PAT18, PAT19, PAT20, PAT21),
;    1834  1            BST_CNT : word initial (0),             ! CURRENT SEQUENTIAL BLOCK COUNT
;    1835  1            BST_DEV : word initial (0),             ! CURRENT SEQUENTIAL BLOCK DEVICE
;    1836  1            CURRENT_VECTOR : word,                  ! CURRENT DEVICE'S VECTOR ADDRESS
;    1837  1            BRLEVEL : word,                         ! CURRENT DEVICE'S BR LEVEL
;    1838  1            DUOFF   : WORD,                         !DUP OFFSET INTO CST                    ZZZ
;    1839  1            DRS_START,                              ! START OF THE SUPERVISOR
;    1840  1            APT_MODE : byte initial (byte (FALSE)),     ! FLAG SET IF EXERCISER RUNNING UNDER APT
;    1841  1            MAIL_BOX_TESTNUM,                       ! ADDRESS OF TEST NUMBER LOCATION IN APT MAIL-BOX
;    1842  1            MAIL_BOX_SUBTST,                        ! ADDRESS OF SUB-TEST NUMBER LOCATION IN APT MAIL-BOX
;    1843  1            COMPARE_DATA : byte,                    ! FLAG CLEARED TO BYPASS HOST COMPARES
;    1844  1            DRS_FLAGS: word,                        ! FLAGS USED IN START/RESTART OF THE EXERCISER
;    1845  1            RD_MAX_SEQ_CNT : word,                  ! COUNT USED IN SEQUENTIAL ACCESS OPERATIONS
;    1846  1            RX_MAX_SEQ_CNT : word;                  !
;    1847  1
;    1848  1        external routine
;    1849  1            NEX_TRAP : L$ISR novalue,
;    1850  1            TIME : L$ISR novalue,
;    1851  1            SET_CPAR : novalue,
;    1852  1            SET_UPAR : novalue,
;    1853  1            OUT_IODQ,
;    1854  1            IN_IODQ : novalue,
;    1855  1            GET_PKT,
;    1856  1            PUT_PKT : novalue,
;    1857  1            GET_RETPKT,
;    1858  1            PUT_RETPKT : novalue,
;    1859  1            GET_IO_BUFF : novalue,
;    1860  1            PUT_IO_BUFF : novalue,
;    1861  1            PUTA_BUFF : novalue,
```

```
;   1862  1          SEND,
;   1863  1          WAIT : novalue,
;   1864  1          MODULAS,                                        !ZZZ
;   1865  1          DROP_CTLR : novalue,
;   1866  1          DRV_CTLERR : novalue,
;   1867  1          EMS_RP1 : novalue,
;   1868  1          EMS_EL : novalue,
;   1869  1          EMS_CMP : novalue,
;   1870  1          EMS_ERR : novalue,
;   1871  1          EMS_10 : novalue,
;   1872  1          EMS_12 : novalue,
;   1873  1          EMS_13 : novalue,
;   1874  1          EMS_14 : novalue,
;   1875  1          EMS_18 : novalue,
;   1876  1          EMS_21 : novalue,
;   1877  1          EMS_22 : NOVALUE,                               !ZZZ
;   1878  1          EMS_24 : novalue,
;   1879  1          EMS_30 : novalue;
```

```
;    1880  1     %sbttl 'TEST SECTION'
;    1881  1
;    1882  1
;    1883  1     !+
;    1884  1     !       THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RDRX EXERCISER.
;    1885  1     !       THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO A NUMBER OF
;    1886  1     !       SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;    1887  1     !-
;    1888  1
;    1889  1
;    1890  3 .   BGNTST;
;    1891  3
;    1892  3     local
;    1893  3         DUMMY_0 : word,
;    1894  3         DUMMY_1 : word;
;    1895  3
;    1896  3
;    1897  3
;    1898  3     EOP_FLAG = TRUE;                                        ! ASSUME NO UNIT AVAILABLE
;    1899  3     COMPARE_DATA = TRUE;                                    ! ALLOW HOST COMAPRES IF ASKED FOR
;    1900  3     DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);              !CLEAR DUP INIT FLAG     ZZZ
;    1901  3     HOE_FLAG = FALSE;                                       ! ASSUME 'HOE' FLAG NOT SET
;    1902  3     FORCED_ERROR = FALSE;                                   ! INITIALIZE "FORCED ERROR" FLAG
;    1903  3
;    1904  3
;    1905  3     incr I from 0 to PKT_CNT - 1 do                         ! INITIALIZE PACKET AREA
;    1906  4         begin
;    1907  4
;    1908  4         incr J from 0 to PKT_LEN - 1 do
;    1909  4             MSCP_PKT [.I, .J, 0, 16, 0] = 0;
;    1910  4
;    1911  4         MSCP_PKT [.I, RSP_RECEIVED] = FALSE;
;    1912  3         end;
;    1913  3
;    1914  3     incr I from 0 to RP_CNT - 1 do                          ! INITIALIZE RESPONSE SAVE AREA
;    1915  3         incr J from 0 to RP_LEN - 1 do
;    1916  3             RETPKT [.I, .J, 0, 16, 0] = 0;
;    1917  3
;    1918  3     incr I from 0 to EP_CNT do                              ! INITIALIZE ERROR-LOG SAVE AREA
;    1919  4         begin
;    1920  4
;    1921  4         incr J from 0 to EP_LEN - 1 do
;    1922  4             ELOG_PKT [.I, .J, 0, 16, 0] = 0;
;    1923  4
;    1924  4         ELOG_PKT [.I, EL_CONTENTS] = EMPTY;
;    1925  3         end;
;    1926  3
;    1927  4     if BIT_TST (SWP_FLAGS, SWF_CWC)                         ! NO SIMULTANEOUS CNTR/HOST WRIE CHECKS
;    1928  3     then
;    1929  3         SWP_FLAGS = .SWP_FLAGS and (not SWF_HWC);
;    1930  3
;    1931  4     if BIT_TST (SWP_FLAGS, SWF_RDM)                         ! NO SIMULTANEOUS RANDOM/SEQUENTIAL SELECTS
;    1932  3     then
```

```
;    1933  3          SWP_FLAGS = .SWP_FLAGS and (not SWF_SEQ);
;    1934  3
;    1935  3      if not .INIT_OCCURED
;    1936  3      then
;    1937  4          begin
;    1938  4          DRS_START = .FREE_MEM_ADDR + 2 + (..FREE_MEM_ADDR * 2);          ! START OF SUPERVISOR
;    1939  4
;    1940  4
;    1941  4      !+
;    1942  4      ! CAUTION...     THE FOLLOWING CODE IS 'KLUGED' TO DETERMINE IF THE EXERCISER IS RUNNING UNDER THE
;    1943  4      !                APT SUPERVISOR, AND IS DEPENDENT FOR IT'S SUCCESS ON THE KNOWLEDGE OF THE ACTUAL
;    1944  4      !                APT SUPERVISOR AND THE ADDRESS OF THE MAIL-BOX WITHIN THE SUPERVISOR.
;    1945  4      !-
;    1946  4
;    1947  4
;    1948  4
;    1949  4          if (..DRS_START eql %o'167') and                              ! APT DRS STARTS WITH A JMP INSTRUCTION
;    1950  5             (not MANUAL)
;    1951  4          then
;    1952  5              begin
;    1953  5              APT_MODE = TRUE;
;    1954  5              MAIL_BOX_TESTNUM = .DRS_START + %o'62' + %o'6';          ! APT MAIL-BOX IS OFFSET AT OCTAL 62 FROM
;    1955  5              MAIL_BOX_SUBTST = .DRS_START + %o'62' + %o'4';           ! BEGINNING OF SUPERVISOR
;    1956  4              end;
;    1957  4
;    1958  4
;    1959  4          NEX = FALSE;                                                 ! CHECK IF LINE-CLOCK PRESENT
;    1960  4          CLK_PRESENT = FALSE;                                         !
;    1961  4          SETVEC (4, NEX_TRAP, PRI07);                                 ! SET TRAP CATCHER ADDRESS
;    1962  4          DUMMY_0 = .LINE_CLOCK;                                       ! TRY TO ADDRESS THE CLOCK
;    1963  4          DUMMY_1 = 0;                                                 ! DUMMY INSTRUCTION
;    1964  4          CLRVEC (4);                                                  ! RETURN LOC 4 TO THE SUPERVISOR
;    1965  4
;    1966  4
;    1967  4          if not .NEX
;    1968  4          then
;    1969  5              begin
;    1970  5              CLK_PRESENT = TRUE;                                      ! SET FLAG IF CLOCK PRESENT
;    1971  5              CLK_TICKS = 0;                                           ! INITIALIZE THE LINE-CLOCK TICK COUNT
;    1972  5              HOURS = .SWP_TIME / 100;                                 ! TIME OF DAY (HOURS)
;    1973  5              MINUTES = (.SWP_TIME mod 100) + 1;                       ! TIME OF DAY (MINUTES)
;    1974  5
;    1975  5              while .MINUTES gequ 60 do                                ! NORMALIZE MINUTES
;    1976  6                  begin
;    1977  6                  MINUTES = .MINUTES - 60;
;    1978  6                  HOURS = .HOURS + 1;
;    1979  5                  end;
;    1980  5
;    1981  5              HOURS = .HOURS mod 24;                                   ! NORMALIZE HOURS
;    1982  4              end;
;    1983  4
;    1984  3          end;
;    1985  3
```

```
;    1986  3
;    1987  3     if .CLK_PRESENT
;    1988  3     then
;    1989  4         begin
;    1990  4         SETVEC (%o'100', TIME, PRI06);                          ! LINE-CLOCK VECTOR
;    1991  4         LINE_CLOCK = BIT6;                                      ! START THE CLOCK
;    1992  3         end;
;    1993  3
;    1994  3     RFLAGS (DRS_FLAGS);                                         ! READ DRS FLAGS INTO LOC DRS_FLAGS
;    1995  3
;    1996  3     if BIT_TST (DRS_FLAGS, HOE) eql HOE
;    1997  3     then
;    1998  3         HOE_FLAG = TRUE;                                        ! SET FLAG IF 'HOE' SET
;    1999  3
;    2000  3
;    2001  3     INIT_TEST ();                                               ! INITIALIZE TEST ENVIRONMENT
;    2002  3
;    2003  3     incr CTLR from 0 to (MAX_CTLR - 1) do                       ! FOR EVERY CONTROLLER
;    2004  3
;    2005  3         if (.CST [.CTLR, STATE] eql ONLINE) and                 ! IF CONTROLLER ONLINE
;    2006  3            (.DCT [.CTLR, STAT] eql ONLINE) and
;    2007  4            (.CST [.CTLR, U_CNT] gequ 0)
;    2008  3         then
;    2009  3             incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + 4) by UNIT_SIZE do
;    2010  3
;    2011  3                 if .CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE   ! IF AT LEAST ONE UNIT ALIVE
;    2012  3                 then
;    2013  4                     begin
;    2014  4                     EOP_FLAG = FALSE;                           ! NOT END OF PASS
;    2015  4                     exitloop;
;    2016  3                     end;
;    2017  3
;    2018  3     if not .EOP_FLAG
;    2019  3     then
;    2020  3         MULTI_DRIVE ();                                         ! RUN MULTI-DRIVE TEST
;    2021  1     ENDTST;


                                        .TITLE  ZRQAM3 RD/RX EXERCISER
                                        .IDENT  /V01.8/
                                        .ENABL  AMA


000000                                  .PSECT  $GGG$,  RO
000000                          COMM.AREA:
                                        .BLKW   24
000050                          DPST:   .BLKW   2
000054                          MAX.LBN:.BLKW   4
000064                          STORAGE:.BLKW   4
000074                          ICOM.ADDR:
                                        .BLKW   1
000076                          ICST.ADDR:
                                        .BLKW   1
```

```
000100                          IDCT.ADDR:
                                        .BLKW   1
000102  000000V                 INT.ADDR:
                                        .WORD   AZINTO
000104                          ICTLR:  .BLKW   1
000106  000003                  RW.BALANCE:
                                        .WORD   3
000110  000000                  RD.COUNT:
                                        .WORD   0
000112                          MX1:    .BLKW   1
000114                          MX2:    .BLKW   1
000116                          MAD1:   .BLKW   1
000120                          MAD2:   .BLKW   1
000122                          LAST.PKT:
                                        .BLKW   3
000130  000001                  PAT02:  .WORD   1
000132  000000                          .WORD   0
000134  000001                  PAT03:  .WORD   1
000136  177777                          .WORD   -1
000140  000001                  PAT04:  .WORD   1
000142  105613                          .WORD   -72165
000144  000001                  PAT05:  .WORD   1
000146  031463                          .WORD   31463
000150  000001                  PAT06:  .WORD   1
000152  030221                          .WORD   30221
000154  000020                  PAT07:  .WORD   20
000156  000001                          .WORD   1
000160  000003                          .WORD   3
000162  000007                          .WORD   7
000164  000017                          .WORD   17
000166  000037                          .WORD   37
000170  000077                          .WORD   77
000172  000177                          .WORD   177
000174  000377                          .WORD   377
000176  000777                          .WORD   777
000200  001777                          .WORD   1777
000202  003777                          .WORD   3777
000204  007777                          .WORD   7777
000206  017777                          .WORD   17777
000210  037777                          .WORD   37777
000212  077777                          .WORD   77777
000214  177777                          .WORD   -1
000216  000020                  PAT08:  .WORD   20
000220  177776                          .WORD   -2
000222  177774                          .WORD   -4
000224  177770                          .WORD   -10
000226  177760                          .WORD   -20
000230  177740                          .WORD   -40
000232  177700                          .WORD   -100
000234  177600                          .WORD   -200
000236  177400                          .WORD   -400
000240  177000                          .WORD   -1000
000242  176000                          .WORD   -2000
```

```
000244  174000                          .WORD   -4000
000246  170000                          .WORD   -10000
000250  160000                          .WORD   -20000
000252  140000                          .WORD   -40000
000254  100000                          .WORD   -100000
000256  000000                          .WORD   0
000260  000020                  PAT09:  .WORD   20
000262  000000                          .WORD   0
000264  000000                          .WORD   0
000266  000000                          .WORD   0
000270  177777                          .WORD   -1
000272  177777                          .WORD   -1
000274  177777                          .WORD   -1
000276  000000                          .WORD   0
000300  000000                          .WORD   0
000302  177777                          .WORD   -1
000304  177777                          .WORD   -1
000306  000000                          .WORD   0
000310  177777                          .WORD   -1
000312  000000                          .WORD   0
000314  177777                          .WORD   -1
000316  000000                          .WORD   0
000320  177777                          .WORD   -1
000322  000001                  PAT10:  .WORD   1
000324  133331                          .WORD   -44447
000326  000020                  PAT11:  .WORD   20
000330  052525                          .WORD   52525
000332  052525                          .WORD   52525
000334  052525                          .WORD   52525
000336  125252                          .WORD   -52526
000340  125252                          .WORD   -52526
000342  125252                          .WORD   -52526
000344  052525                          .WORD   52525
000346  052525                          .WORD   52525
000350  125252                          .WORD   -52526
000352  125252                          .WORD   -52526
000354  052525                          .WORD   52525
000356  125252                          .WORD   -52526
000360  052525                          .WORD   52525
000362  125252                          .WORD   -52526
000364  052525                          .WORD   52525
000366  125252                          .WORD   -52526
000370  000024                  PAT12:  .WORD   24
000372  026455                          .WORD   26455
000374  026455                          .WORD   26455
000376  026455                          .WORD   26455
000400  151322                          .WORD   -26456
000402  151322                          .WORD   -26456
000404  151322                          .WORD   -26456
000406  026455                          .WORD   26455
000410  026455                          .WORD   26455
000412  151322                          .WORD   -26456
000414  151322                          .WORD   -26456
```

```
000416  026455                          .WORD   26455
000420  026455                          .WORD   26455
000422  151322                          .WORD   -26456
000424  026455                          .WORD   26455
000426  151322                          .WORD   -26456
000430  026455                          .WORD   26455
000432  151322                          .WORD   -26456
000434  026455                          .WORD   26455
000436  151322                          .WORD   -26456
000440  026455                          .WORD   26455
000442  000001                  PAT13:  .WORD   1
000444  066666                          .WORD   66666
000446  000020                  PAT14:  .WORD   20
000450  000001                          .WORD   1
000452  000002                          .WORD   2
000454  000004                          .WORD   4
000456  000010                          .WORD   10
000460  000020                          .WORD   20
000462  000040                          .WORD   40
000464  000100                          .WORD   100
000466  000200                          .WORD   200
000470  000400                          .WORD   400
000472  001000                          .WORD   1000
000474  002000                          .WORD   2000
000476  004000                          .WORD   4000
000500  010000                          .WORD   10000
000502  020000                          .WORD   20000
000504  040000                          .WORD   40000
000506  100000                          .WORD   -100000
000510  000020                  PAT15:  .WORD   20
000512  177776                          .WORD   -2
000514  177775                          .WORD   -3
000516  177773                          .WORD   -5
000520  177767                          .WORD   -11
000522  177757                          .WORD   -21
000524  177737                          .WORD   -41
000526  177677                          .WORD   -101
000530  177577                          .WORD   -201
000532  177377                          .WORD   -401
000534  176777                          .WORD   -1001
000536  175777                          .WORD   -2001
000540  173777                          .WORD   -4001
000542  167777                          .WORD   -10001
000544  157777                          .WORD   -20001
000546  137777                          .WORD   -40001
000550  077777                          .WORD   77777
000552  000020                  PAT16:  .WORD   20
000554  133331                          .WORD   -44447
000556  133331                          .WORD   -44447
000560  133331                          .WORD   -44447
000562  155554                          .WORD   -22224
000564  155554                          .WORD   -22224
000566  155554                          .WORD   -22224
```

```
000570  133331                          .WORD   -44447
000572  133331                          .WORD   -44447
000574  155554                          .WORD   -22224
000576  155554                          .WORD   -22224
000600  133331                          .WORD   -44447
000602  155554                          .WORD   -22224
000604  133331                          .WORD   -44447
000606  155554                          .WORD   -22224
000610  133331                          .WORD   -44447
000612  155554                          .WORD   -22224
000614  000025                  PAT17:  .WORD   25
000616  000000                          .WORD   0
000620  106466                          .WORD   -71312
000622  106466                          .WORD   -71312
000624  071311                          .WORD   71311
000626  071311                          .WORD   71311
000630  071311                          .WORD   71311
000632  106466                          .WORD   -71312
000634  106466                          .WORD   -71312
000636  106466                          .WORD   -71312
000640  106466                          .WORD   -71312
000642  071311                          .WORD   71311
000644  071311                          .WORD   71311
000646  071311                          .WORD   71311
000650  071311                          .WORD   71311
000652  071311                          .WORD   71311
000654  106466                          .WORD   -71312
000656  106466                          .WORD   -71312
000660  106466                          .WORD   -71312
000662  106466                          .WORD   -71312
000664  106466                          .WORD   -71312
000666  106466                          .WORD   -71312
000670  000025                  PAT18:  .WORD   25
000672  106466                          .WORD   -71312
000674  000000                          .WORD   0
000676  071311                          .WORD   71311
000700  106466                          .WORD   -71312
000702  106466                          .WORD   -71312
000704  106466                          .WORD   -71312
000706  071311                          .WORD   71311
000710  071311                          .WORD   71311
000712  071311                          .WORD   71311
000714  071311                          .WORD   71311
000716  106466                          .WORD   -71312
000720  106466                          .WORD   -71312
000722  106466                          .WORD   -71312
000724  106466                          .WORD   -71312
000726  106466                          .WORD   -71312
000730  071311                          .WORD   71311
000732  071311                          .WORD   71311
000734  071311                          .WORD   71311
000736  071311                          .WORD   71311
000740  071311                          .WORD   71311
```

```
000742  071311                          .WORD   71311
000744  000025          PAT19:          .WORD   25
000746  000000                          .WORD   0
000750  134631                          .WORD   -43147
000752  134631                          .WORD   -43147
000754  043146                          .WORD   43146
000756  043146                          .WORD   43146
000760  043146                          .WORD   43146
000762  134631                          .WORD   -43147
000764  134631                          .WORD   -43147
000766  134631                          .WORD   -43147
000770  134631                          .WORD   -43147
000772  043146                          .WORD   43146
000774  043146                          .WORD   43146
000776  043146                          .WORD   43146
001000  043146                          .WORD   43146
001002  043146                          .WORD   43146
001004  134631                          .WORD   -43147
001006  134631                          .WORD   -43147
001010  134631                          .WORD   -43147
001012  134631                          .WORD   -43147
001014  134631                          .WORD   -43147
001016  134631                          .WORD   -43147
001020  000025          PAT20:          .WORD   25
001022  134631                          .WORD   -43147
001024  000000                          .WORD   0
001026  043146                          .WORD   43146
001030  134631                          .WORD   -43147
001032  134631                          .WORD   -43147
001034  134631                          .WORD   -43147
001036  043146                          .WORD   43146
001040  043146                          .WORD   43146
001042  043146                          .WORD   43146
001044  043146                          .WORD   43146
001046  134631                          .WORD   -43147
001050  134631                          .WORD   -43147
001052  134631                          .WORD   -43147
001054  134631                          .WORD   -43147
001056  134631                          .WORD   -43147
001060  043146                          .WORD   43146
001062  043146                          .WORD   43146
001064  043146                          .WORD   43146
001066  043146                          .WORD   43146
001070  043146                          .WORD   43146
001072  043146                          .WORD   43146
001074  000001          PAT21:          .WORD   1
001076  000000                          .WORD   0
001100  000000G         DPA.TBL:        .WORD   RDM.CNT
001102  000130'                         .WORD   PAT02
001104  000134'                         .WORD   PAT03
001106  000140'                         .WORD   PAT04
001110  000144'                         .WORD   PAT05
001112  000150'                         .WORD   PAT06
```

```
001114  000154'                         .WORD   PAT07
001116  000216'                         .WORD   PAT08
001120  000260'                         .WORD   PAT09
001122  000322'                         .WORD   PAT10
001124  000326'                         .WORD   PAT11
001126  000370'                         .WORD   PAT12
001130  000442'                         .WORD   PAT13
001132  000446'                         .WORD   PAT14
001134  000510'                         .WORD   PAT15
001136  000552'                         .WORD   PAT16
001140  000614'                         .WORD   PAT17
001142  000670'                         .WORD   PAT18
001144  000744'                         .WORD   PAT19
001146  001020'                         .WORD   PAT20
001150  001074'                         .WORD   PAT21
001152  000000                  BST.CNT:.WORD   0
001154  000000                  BST.DEV:.WORD   0
001156                          CURRENT.VECTOR:
                                        .BLKW   1
001160                          BRLEVEL:.BLKW   1
001162                          DUOFF:  .BLKW   1
001164                          DRS.START:
                                        .BLKW   1
001166                          APT.MODE:
001166  000                             .BYTE   0
                                        .EVEN
001170                          MAIL.BOX.TESTNUM:
                                        .BLKW   1
001172                          MAIL.BOX.SUBTST:
                                        .BLKW   1
001174                          COMPARE.DATA:
                                        .BLKB   1
                                        .EVEN
001176                          DRS.FLAGS:
                                        .BLKW   1
001200                          RD.MAX.SEQ.CNT:
                                        .BLKW   1
001202                          RX.MAX.SEQ.CNT:
                                        .BLKW   1

                                .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                                .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                                .GLOBL  DUPPKT, TRK.SGN, RDM.CNT, RANDOM
                                .GLOBL  C.ERR.TBL, MSCP.PKT, IPKT.ADDR
                                .GLOBL  PKT.USE, RETPKT, RP.USE, RP.INDX
                                .GLOBL  RP.ADDR, ELOG.PKT, BUFF.ADDR, BUFF.OWN
                                .GLOBL  IODQ, IODQ.IN, IODQ.OUT, ENTRY.REASON
                                .GLOBL  EOP.FLAG, DUP.FLAGS, CCTLR, CDISK
                                .GLOBL  CUOFF, CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
                                .GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
                                .GLOBL  STEP, OF.RC, SA.REG, CMD.TIME
                                .GLOBL  NEX, CRN.LOW, CRN.HIGH, CREDIT.BAL
```

```
                .GLOBL  NEXT.PKT.USE, HOURS, MINUTES, CLK.TICKS
                .GLOBL  CLK.PRESENT, HOE.FLAG, FORCED.ERROR
                .GLOBL  FER.LBN, FER.BC, INIT.OCCURED
                .GLOBL  ADDR.VECT.OK, S.PATTERN, S.DUPPKT
                .GLOBL  P.INDEX, DBM12, DBM18, DBM19, DBM20
                .GLOBL  DBM21, DBM22, DBM23, DBM25, DBM26
                .GLOBL  DBM27, DBM29, DBM108, DBM109, DBM111
                .GLOBL  DBM112, DBM120, DBM121, EH.0, EH.1
                .GLOBL  EH.2, EH.3, EH.4, EH.5, EH.6, EH.7
                .GLOBL  EH.8, EH.9, EH.10, EH.12, EH.13
                .GLOBL  MSG.02, MSG.03, EGS.02, EGD.10
                .GLOBL  EGD.11, EGD.12, EGD.13, EGD.14
                .GLOBL  EGD.15, EGD.16, EGD.17, EGD.18
                .GLOBL  EGD.19, EGD.20, EGD.21, EGD.22
                .GLOBL  EGD.23, EGD.24, EGH.30, DF.MSG
                .GLOBL  HRD.MSG, SFT.MSG, HRD.SUB, CRLF
                .GLOBL  SWP.ERROR, SWP.XFER, SWP.FLAGS
                .GLOBL  DUPROUND, SWP.RAT, SWP.DPAT, SWP.UCNT
                .GLOBL  SWP.TIME, SWP.UDPAT, L$LUN, L$UNIT
                .GLOBL  NEX.TRAP, TIME, SET.CPAR, SET.UPAR
                .GLOBL  OUT.IODQ, IN.IODQ, GET.PKT, PUT.PKT
                .GLOBL  GET.RETPKT, PUT.RETPKT, GET.IO.BUFF
                .GLOBL  PUT.IO.BUFF, PUTA.BUFF, SEND, WAIT
                .GLOBL  MODULAS, DROP.CTLR, DRV.CTLERR
                .GLOBL  EMS.RP1, EMS.EL, EMS.CMP, EMS.ERR
                .GLOBL  EMS.10, EMS.12, EMS.13, EMS.14
                .GLOBL  EMS.18, EMS.21, EMS.22, EMS.24
                .GLOBL  EMS.30

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=           1000
000400          BIT8=           400
000200          BIT7=           200
000100          BIT6=           100
000040          BIT5=           40
000020          BIT4=           20
000010          BIT3=           10
```

```
000004                  BIT2==          4
000002                  BIT1==          2
000001                  BIT0==          1
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE==   36
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000340                  PRI07==         340
000300                  PRI06==         300
000240                  PRI05==         240
000200                  PRI04==         200
000140                  PRI03==         140
000100                  PRI02==         100
000040                  PRI01==         40
000000                  PRI00==         0
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


                        .SBTTL  $T1 TEST SECTION
000000                  .PSECT  $CODE$,  RO

000000  004137  000000G       $T1:    JSR     R1,$SAVE3                       ;                       1879
000004  112737  000001  000000G       MOVB    #1,EOP.FLAG                     ;                       1898
000012  112737  000001  001174'       MOVB    #1,COMPARE.DATA                 ;                       1899
000020  042737  000002  000000G       BIC     #2,DUP.FLAGS                    ;                       1900
000026  105037  000000G               CLRB    HOE.FLAG                        ;                       1901
000032  105037  000000G               CLRB    FORCED.ERROR                    ;                       1902
000036  005002                        CLR     R2                              ; I                     1905
000040  010246                1$:     MOV     R2,-(SP)                        ; I,*                   1909
000042  012746  000043                MOV     #43,-(SP)
000046  004737  000000G               JSR     PC,BL$MUL
000052  005001                        CLR     R1                              ; J                     1908
000054  010003                2$:     MOV     R0,R3                           ;                       1909
000056  060103                        ADD     R1,R3                           ; J,*
000060  006303                        ASL     R3
000062  005063  000000G               CLR     MSCP.PKT(R3)
000066  005201                        INC     R1                              ; J                     1908
000070  020127  000042                CMP     R1,#42                          ; J,*
000074  003767                        BLE     2$
```

```
000076  010216                          MOV     R2,(SP)                 ; I,*                   1911
000100  012746  000106                  MOV     #106,-(SP)
000104  004737  000000G                 JSR     PC,BL$MUL
000110  105060  000005G                 CLRB    MSCP.PKT+5(R0)
000114  062706  000006                  ADD     #6,SP                   ;                       1906
000120  005202                          INC     R2                      ; I                     1905
000122  020227  000013                  CMP     R2,#13                  ; I,*
000126  003744                          BLE     1$
000130  005002                          CLR     R2                      ; I                     1914
000132  005001                  3$:     CLR     R1                      ; J                     1915
000134  010200                  4$:     MOV     R2,R0                   ; I,*                   1916
000136  060100                          ADD     R1,R0                   ; J,*
000140  006300                          ASL     R0
000142  005060  000000G                 CLR     RETPKT(R0)
000146  005201                          INC     R1                      ; J                     1915
000150  020127  000025                  CMP     R1,#25                  ; J,*
000154  003767                          BLE     4$
000156  062702  000026                  ADD     #26,R2                  ; *,I                   1914
000162  020227  000232                  CMP     R2,#232                 ; I,*
000166  003761                          BLE     3$
000170  005002                          CLR     R2                      ; I                     1918
000172  010246                  5$:     MOV     R2,-(SP)                ; I,*                   1922
000174  012746  000041                  MOV     #41,-(SP)
000200  004737  000000G                 JSR     PC,BL$MUL
000204  005001                          CLR     R1                      ; J                     1921
000206  010003                  6$:     MOV     R0,R3                   ;                       1922
000210  060103                          ADD     R1,R3                   ; J,*
000212  006303                          ASL     R3
000214  005063  000000G                 CLR     ELOG.PKT(R3)
000220  005201                          INC     R1                      ; J                     1921
000222  020127  000040                  CMP     R1,#40                  ; J,*
000226  003767                          BLE     6$
000230  010216                          MOV     R2,(SP)                 ; I,*                   1924
000232  012746  000102                  MOV     #102,-(SP)
000236  004737  000000G                 JSR     PC,BL$MUL
000242  105060  000001G                 CLRB    ELOG.PKT+1(R0)
000246  062706  000006                  ADD     #6,SP                   ;                       1919
000252  005202                          INC     R2                      ; I                     1918
000254  020227  000014                  CMP     R2,#14                  ; I,*
000260  003744                          BLE     5$
000262  032737  000020  000000G         BIT     #20,SWP.FLAGS           ;                       1927
000270  001403                          BEQ     7$
000272  042737  000040  000000G         BIC     #40,SWP.FLAGS           ;                       1929
000300  032737  000002  000000G 7$:     BIT     #2,SWP.FLAGS            ;                       1931
000306  001403                          BEQ     8$
000310  042737  001000  000000G         BIC     #1000,SWP.FLAGS         ;                       1933
000316  132737  000001  000000G 8$:     BITB    #1,INIT.OCCURED         ;                       1935
000324  001145                          BNE     13$
000326  017700  000000G                 MOV     @FREE.MEM.ADDR,R0       ;                       1938
000332  006300                          ASL     R0
000334  063700  000000G                 ADD     FREE.MEM.ADDR,R0
000340  010037  001164'                 MOV     R0,DRS.START
000344  062737  000002  001164'         ADD     #2,DRS.START
```

```
000352  027727  001164' 000167          CMP     @DRS.START,#167                 ;                       1949
000360  001021                          BNE     9$
000362  104450                          TRAP    50                              ;                       1950
000364  103417                          BCS     9$
000366  112737  000001  001166'        MOVB    #1,APT.MODE                     ;                       1953
000374  013737  001164' 001170'        MOV     DRS.START,MAIL.BOX.TESTNUM      ;                       1954
000402  062737  000070  001170'        ADD     #70,MAIL.BOX.TESTNUM
000410  013737  001164' 001172'        MOV     DRS.START,MAIL.BOX.SUBTST       ;                       1955
000416  062737  000066  001172'        ADD     #66,MAIL.BOX.SUBTST
000424  005037  000000G          9$:    CLR     NEX                             ;                       1959
000430  105037  000000G                  CLRB    CLK.PRESENT                     ;                       1960
000434  012746  000340                  MOV     #340,-(SP)                      ;                       1961
000440  012746  000000G                  MOV     #NEX.TRAP,-(SP)
000444  012746  000004                  MOV     #4,-(SP)
000450  012746  000003                  MOV     #3,-(SP)
000454  104437                          TRAP    37
000456  012700  000004                  MOV     #4,R0                           ;                       1964
000462  104436                          TRAP    36
000464  032737  000001  000000G        BIT     #1,NEX                          ;                       1967
000472  001060                          BNE     12$
000474  112737  000001  000000G        MOVB    #1,CLK.PRESENT                  ;                       1970
000502  005037  000000G                  CLR     CLK.TICKS                       ;                       1971
000506  013716  000000G                  MOV     SWP.TIME,(SP)                   ;                       1972
000512  012746  000144                  MOV     #144,-(SP)
000516  004737  000000G                  JSR     PC,BL$DIV
000522  110037  000000G                  MOVB    R0,HOURS
000526  013716  000000G                  MOV     SWP.TIME,(SP)                   ;                       1973
000532  012746  000144                  MOV     #144,-(SP)
000536  004737  000000G                  JSR     PC,BL$MOD
000542  010001                          MOV     R0,R1
000544  005201                          INC     R1
000546  110137  000000G                  MOVB    R1,MINUTES
000552  123727  000000G 000074   10$:    CMPB    MINUTES,#74                     ;                       1975
000560  103412                          BLO     11$
000562  005000                          CLR     R0                              ;                       1977
000564  153700  000000G                  BISB    MINUTES,R0
000570  162700  000074                  SUB     #74,R0
000574  110037  000000G                  MOVB    R0,MINUTES
000600  105237  000000G                  INCB    HOURS                           ;                       1978
000604  000762                          BR      10$                             ;                       1975
000606  005016                  11$:    CLR     (SP)                            ;                       1981
000610  113716  000000G                  MOVB    HOURS,(SP)
000614  012746  000030                  MOV     #30,-(SP)
000620  004737  000000G                  JSR     PC,BL$MOD
000624  110037  000000G                  MOVB    R0,HOURS
000630  062706  000006                  ADD     #6,SP                           ;                       1969
000634  062706  000010          12$:    ADD     #10,SP                          ;                       1937
000640  132737  000001  000000G  13$:    BITB    #1,CLK.PRESENT                  ;                       1987
000646  001416                          BEQ     14$
000650  012746  000300                  MOV     #300,-(SP)                      ;                       1990
000654  012746  000000G                  MOV     #TIME,-(SP)
000660  012746  000100                  MOV     #100,-(SP)
000664  012746  000003                  MOV     #3,-(SP)
```

```
000670  104437                          TRAP    37
000672  012737  000100  177546        MOV     #100,@#177546                   ;                       1991
000700  062706  000010                  ADD     #10,SP                          ;                       1989
000704  104421                  14$:    TRAP    21                              ;                       1994
000706  010037  001176'                MOV     R0,DRS.FLAGS
000712  042700  077777                  BIC     #77777,R0                       ;                       1996
000716  020027  100000                  CMP     R0,#-100000
000722  001003                          BNE     15$
000724  012700  000001                  MOV     #1,R0
000730  000401                          BR      16$
000732  005000                  15$:    CLR     R0
000734  020027  100000          16$:    CMP     R0,#-100000
000740  001003                          BNE     17$
000742  112737  000001  000000G       MOVB    #1,HOE.FLAG                     ;                       1998
000750  004737  000000V          17$:    JSR     PC,INIT.TEST                    ;                       2001
000754  005002                          CLR     R2                              ; CTLR                  2003
000756  010246                  18$:    MOV     R2,-(SP)                        ; CTLR,*                2005
000760  012746  000126                  MOV     #126,-(SP)
000764  004737  000000G                  JSR     PC,BL$MUL
000770  022626                          CMP     (SP)+,(SP)+
000772  005760  000002G                  TST     CST+2(R0)
000776  100040                          BPL     22$
001000  010246                          MOV     R2,-(SP)                        ; CTLR,*                2006
001002  012746  000022                  MOV     #22,-(SP)
001006  004737  000000G                  JSR     PC,BL$MUL
001012  022626                          CMP     (SP)+,(SP)+
001014  005760  000000G                  TST     DCT(R0)
001020  100027                          BPL     22$
001022  010246                          MOV     R2,-(SP)                        ; CTLR,*                2011
001024  012746  000053                  MOV     #53,-(SP)
001030  004737  000000G                  JSR     PC,BL$MUL
001034  012701  000003                  MOV     #3,R1                           ; *,OFFSET              2009
001040  010003                  19$:    MOV     R0,R3                           ;                       2011
001042  060103                          ADD     R1,R3                           ; OFFSET,*
001044  006303                          ASL     R3
001046  032763  020000  000000G         BIT     #20000,CST(R3)
001054  001403                          BEQ     20$
001056  105037  000000G                  CLRB    EOP.FLAG                        ;                       2014
001062  000405                          BR      21$                             ;                       2013
001064  062701  000012          20$:    ADD     #12,R1                          ; *,OFFSET              2009
001070  020127  000042                  CMP     R1,#42                          ; OFFSET,*
001074  003761                          BLE     19$
001076  022626                  21$:    CMP     (SP)+,(SP)+
001100  005202                  22$:    INC     R2                              ; CTLR                  2003
001102  000243                          .WORD CLV!CLC

001104  003724                          BLE     18$
001106  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       2018
001114  001002                          BNE     23$
001116  004737  000000V                  JSR     PC,MULTI.DRIVE                  ;                       2020
001122  000207                  23$:    RTS     PC                              ;                       1879

; Routine Size:  298 words,    Routine Base:  $CODE$ + 0000
```

```
; Maximum stack depth per invocation:  12 words



                                        .SBTTL  T1 TEST SECTION
000000  004737  000000'          T1::
000000                           1$:    JSR     PC,$T1                  ;                2020
000004  104466                         TRAP    66
000006  006000                         ROR     R0
000010  103773                         BLO     1$
000012  000207                         RTS     PC

; Routine Size:  6 words,      Routine Base:  $CODE$ + 1124
; Maximum stack depth per invocation:  2 words
```

```
;    2022  1     %sbttl 'INITIALIZATION TEST ROUTINES'
;    2023  1
;    2024  1     GLOBAL routine INIT_TEST : novalue =
;    2025  1
;    2026  1     !+
;    2027  1     !       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
;    2028  1     !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
;    2029  1     !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE TEST
;    2030  1     !       OR THE DM EXERCISER.
;    2031  1     !
;    2032  1     !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
;    2033  1     !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
;    2034  1     !       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
;    2035  1     !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
;    2036  1     !       EACH DISK VIA ROUTINE "ACCESS". THE INITIALIZATION TEST IS DEEMED A
;    2037  1     !       SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH DISK CAN BE ACCESSED.
;    2038  1     !-
;    2039  1
;    2040  2         begin
;    2041  2         DRIVER_INIT ();                         ! INIT DRIVER DATA AND DEVICES
;    2042  2
;    2043  2         incr CTLR from 0 to (MAX_CTLR - 1) do   ! FOR EACH CONTROLLER
;    2044  3             begin
;    2045  3             SET_CPAR (.CTLR);                   ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    2046  3
;    2047  3             if .CST_ADDR [STATE] eql ONLINE     ! IF CONTROLLER IS STILL ALIVE
;    2048  3             then                                ! FOR EACH DISK
;    2049  3
;    2050  3                 incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;    2051  3
;    2052  3                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    2053  3                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    2054  4                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2055  3                 then
;    2056  4                     begin
;    2057  4                     SET_UPAR (.OFFSET);         ! SET UP UNIT-RELATED DATA ITEMS
;    2058  5                     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)          !ZZZ
;    2059  4                        THEN ACCESS ();                                 !ZZZ
;    2060  4                                             !SKIP IF DUP CAUSED INIT ZZZ
;    2061  4
;    2062  3                     end;                      ! IF UNIT IS PRESENT AND ONLINE
;    2063  3
;    2064  2             end;                              ! CONTROLLER LOOP
;    2065  2
;    2066  1         end;                                  ! ROUTINE INIT_TEST


                                .SBTTL  INIT.TEST INITIALIZATION TEST ROUTINES
000000  004137  000000G     INIT.TEST::
                                JSR     R1,$SAVE2                       ;                       2024
000004  004737  000000V             JSR     PC,DRIVER.INIT                  ;                       2041
000010  005002                      CLR     R2                              ; CTLR                  2043
000012  010246              1$:     MOV     R2,-(SP)                        ; CTLR,*                2045
```

```
000014  004737  000000G                        JSR     PC,SET.CPAR
000020  013700  000000G                        MOV     CST.ADDR,R0                     ;                               2047
000024  005760  000002                         TST     2(R0)
000030  100035                                 BPL     4$
000032  012701  000003                         MOV     #3,R1                           ; *,OFFSET                      2050
000036  010100                         2$:     MOV     R1,R0                           ; OFFSET,*                      2052
000040  006300                                 ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  032710  040000                         BIT     #40000,(R0)
000052  001417                                 BEQ     3$
000054  032710  020000                         BIT     #20000,(R0)                     ;                               2053
000060  001414                                 BEQ     3$
000062  032710  010000                         BIT     #10000,(R0)                     ;                               2054
000066  001011                                 BNE     3$
000070  010116                                 MOV     R1,(SP)                         ; OFFSET,*                      2057
000072  004737  000000G                        JSR     PC,SET.UPAR
000076  032737  000002  000000G                BIT     #2,DUP.FLAGS                    ;                               2058
000104  001002                                 BNE     3$
000106  004737  000000V                        JSR     PC,ACCESS                       ;                               2059
000112  062701  000012                 3$:     ADD     #12,R1                          ; *,OFFSET                      2050
000116  020127  000041                         CMP     R1,#41                          ; OFFSET,*
000122  003745                                 BLE     2$
000124  005726                         4$:     TST     (SP)+                           ;                               2044
000126  005202                                 INC     R2                              ; CTLR                          2043
000130  000243                                 .WORD CLV!CLC

000132  003727                                 BLE     1$
000134  000207                                 RTS     PC                              ;                               2024

; Routine Size:  47 words,      Routine Base:  $CODE$ + 1140
; Maximum stack depth per invocation:  5 words
```

```
;   2067  1     GLOBAL routine DRIVER_INIT : novalue =
;   2068  1
;   2069  1     !+
;   2070  1     !       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
;   2071  1     !       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
;   2072  1     !       DRIVER DATA, AND TO BRING EACH RDRX CONTROLLER AND UNIT (DISK)
;   2073  1     !       ONLINE.
;   2074  1     !-
;   2075  1
;   2076  2         begin
;   2077  2
;   2078  2         local
;   2079  2             PKT_ADDR;
;   2080  2
;   2081  2         PKT_ADDR = MSCP_PKT + 10;                                   ! ADDR (TEXT + 0) OF 1ST MSCP PKT
;   2082  2         NEXT_PKT_USE = 0;                                           ! NEXT PACKET TO ALLOCATE
;   2083  2
;   2084  2         incr COUNT from 0 to (PKT_CNT - 1) do                       ! FOR EACH MSCP PACKET
;   2085  3             begin
;   2086  3             PKT_USE [.COUNT] = -1;                                  ! MARK PACKET FREE
;   2087  3             MSCP_PKT [.COUNT, PKT_LO] = .PKT_ADDR;                  ! LOAD ADDR INTO BUFFER DESCRIPTOR
;   2088  3             MSCP_PKT [.COUNT, PKT_HI] = 0;
;   2089  3             MSCP_PKT [.COUNT, CONNID] = CID_DISK;                   ! SET CONNECTION ID TO MSCP ID
;   2090  3             PKT_ADDR = .PKT_ADDR + (PKT_LEN  *  2);                 ! ADVANCE ADDR TO NEXT PACKET
;   2091  2             end;
;   2092  2
;   2093  2         incr CTLR from 0 to (MAX_CTLR - 1) do                       ! FOR EACH CONTROLLER
;   2094  2
;   2095  2             if .CST [.CTLR, IP_ADDR] neqa 0                         ! IF CONTROLLER IS PRESENT
;   2096  2             then
;   2097  3                 begin
;   2098  3                 SET_CPAR (.CTLR);                                   ! CURRENT CONTROLLER PARAMETERS
;   2099  3                 CURRENT_VECTOR = .CST_ADDR [VEC_ADDR];              ! CURRENT CONTROLLER'S VECTOR
;   2100  3                 BRLEVEL = .CST_ADDR [BR_LEV] + 5;                   ! SET CURRENT CONTROLLER'S BR LEVEL
;   2101  3                 CTLR_INIT ();                                       ! INIT DEVICE AND CTLR DATA
;   2102  3
;   2103  3                 if .DCT_ADDR [STAT] eql ONLINE                      ! IF CONTROLLER IS STILL ALIVE
;   2104  3                 then                                                ! FOR EACH DIAK UNIT
;   2105  3
;   2106  3                     incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;   2107  3
;   2108  3                         if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRFSENT) and  ! IF UNIT EXISTS
;   2109  4                            (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;   2110  3                         then
;   2111  4                             begin
;   2112  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_0] = %o'40';      ! BLANK DEVICE NAME
;   2113  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_1] = %o'40';      !
;   2114  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_2] = %o'40';      !
;   2115  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_3] = %o'40';      !
;   2116  4                             SET_UPAR (.OFFSET);                                     ! SET UP UNIT-RELATED DATA ITEMS
;   2117  4                             UNIT_INIT ();                                           ! BRING UNIT ONLINE
;   2118  3                             end;                                                    ! IF UNIT EXISTS
;   2119  3
```

```
;   2120  2                 end;                                          ! IF CONTROLLER IS PRESENT
;   2121  2
;   2122  1             end;                                              ! ROUTINE DRIVER_INIT

                                        .SBTTL  DRIVER.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G          DRIVER.INIT::
                                        JSR     R1,$SAVE3                       ;                               2067
000004  012702  000012G                 MOV     #MSCP.PKT+12,R2                 ; *,PKT.ADDR                    2081
000010  105037  000000G                 CLRB    NEXT.PKT.USE                    ;                               2082
000014  005001                          CLR     R1                              ; COUNT                         2084
000016  112761  000377  000000G  1$:    MOVB    #377,PKT.USE(R1)                ; *,*(COUNT)                    2086
000024  010146                          MOV     R1,-(SP)                        ; COUNT,*                       2087
000026  012746  000106                  MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  010260  000000G                 MOV     R2,MSCP.PKT(R0)                 ; PKT.ADDR,*
000042  005060  000002G                 CLR     MSCP.PKT+2(R0)                  ;                               2088
000046  105060  000011G                 CLRB    MSCP.PKT+11(R0)                 ;                               2089
000052  062702  000106                  ADD     #106,R2                         ; *,PKT.ADDR                    2090
000056  022626                          CMP     (SP)+,(SP)+                     ;                               2085
000060  005201                          INC     R1                              ; COUNT                         2084
000062  020127  000013                  CMP     R1,#13                          ; COUNT,*
000066  003753                          BLE     1$
000070  005003                          CLR     R3                              ; CTLR                          2093
000072  010346                  2$:     MOV     R3,-(SP)                        ; CTLR,*                        2095
000074  012746  000126                  MOV     #126,-(SP)
000100  004737  000000G                 JSR     PC,BL$MUL
000104  022626                          CMP     (SP)+,(SP)+
000106  005760  000000G                 TST     CST(R0)
000112  001503                          BEQ     6$
000114  010346                          MOV     R3,-(SP)                        ; CTLR,*                        2098
000116  004737  000000G                 JSR     PC,SET.CPAR
000122  013700  000000G                 MOV     CST.ADDR,R0                     ;                               2099
000126  016037  000002  001156'         MOV     2(R0),CURRENT.VECTOR
000134  042737  177000  001156'         BIC     #177000,CURRENT.VECTOR
000142  005016                          CLR     (SP)                            ;                               2100
000144  116016  000004                  MOVB    4(R0),(SP)
000150  012746  000005                  MOV     #5,-(SP)
000154  004737  000000G                 JSR     PC,BL$SHF
000160  010037  001160'                 MOV     R0,BRLEVEL
000164  004737  000000V                 JSR     PC,CTLR.INIT                    ;                               2101
000170  005777  000000G                 TST     @DCT.ADDR                       ;                               2103
000174  100051                          BPL     5$
000176  012701  000003                  MOV     #3,R1                           ; *,OFFSET                      2106
000202  013702  000000G          3$:    MOV     CST.ADDR,R2                     ;                               2108
000206  010100                          MOV     R1,R0                           ; OFFSET,*
000210  006300                          ASL     R0
000212  060200                          ADD     R2,R0
000214  032710  040000                  BIT     #40000,(R0)
000220  001432                          BEQ     4$
000222  032710  010000                  BIT     #10000,(R0)                     ;                               2109
000226  001027                          BNE     4$
000230  010100                          MOV     R1,R0                           ; OFFSET,*                      2112
```

F5


```
000232  006300                          ASL     R0
000234  060200                          ADD     R2,R0
000236  112760  000040  000012          MOVB    #40,12(R0)
000244  112760  000040  000013          MOVB    #40,13(R0)                      ;                               2113
000252  010100                          MOV     R1,R0                           ; OFFSET,*                      2114
000254  006300                          ASL     R0
000256  060200                          ADD     R2,R0
000260  112760  000040  000014          MOVB    #40,14(R0)
000266  112760  000040  000015          MOVB    #40,15(R0)                      ;                               2115
000274  010116                          MOV     R1,(SP)                         ; OFFSET,*                      2116
000276  004737  000000G                 JSR     PC,SET.UPAR
000302  004737  000000V                 JSR     PC,UNIT.INIT                    ;                               2117
000306  062701  000012          4$:     ADD     #12,R1                          ; *,OFFSET                      2106
000312  020127  000041                  CMP     R1,#41                          ; OFFSET,*
000316  003731                          BLE     3$
000320  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               2097
000322  005203                  6$:     INC     R3                              ; CTLR                          2093
000324  000243                          .WORD CLV!CLC

000326  003661                          BLE     2$
000330  000207                          RTS     PC                              ;                               2067

; Routine Size:  109 words,    Routine Base:  $CODE$ + 1276
; Maximum stack depth per invocation:  7 words
```

```
;    2123  1     GLOBAL routine CTLR_INIT : novalue =
;    2124  1
;    2125  1     !+
;    2126  1     !       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;    2127  1     !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
;    2128  1     !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;    2129  1     !
;    2130  1     !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;    2131  1     !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
;    2132  1     !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;    2133  1     !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;    2134  1     !            ADDRESS AND INTERRUPT REQUEST LEVEL,
;    2135  1     !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;    2136  1     !
;    2137  1     !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;    2138  1     !       DEVICE ARE DROPPED.
;    2139  1     !-
;    2140  1
;    2141  2         begin
;    2142  2
;    2143  2         local
;    2144  2             RESULT : byte;
;    2145  2
;    2146  2         INI_CTLR_DAT ();                                        ! INITIALIZE CONTROLLER DATA
;    2147  2         SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4);    ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;    2148  2         DCT_ADDR [IG_INT] = TRUE;                               ! SET "IGNORE INTERRUPT" BIT
;    2149  2         L.LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];            ! GET FIRST UNIT NUMBER OF CONTROLLER
;    2150  2                                                                 ! (USED BY DRS FOR DEVICE-FATAL CTLR ERRORS)
;    2151  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2152  2     THEN                                    !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2153  2
;    2154  2         if REG_EXIST () eql FAILURE                             ! REGISTER EXISTENCE TEST
;    2155  2         then
;    2156  3             begin
;    2157  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2158  3             return;
;    2159  2             end;
;    2160  2
;    2161  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2162  2     THEN                                    !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2163  2
;    2164  2         if VEC_BR_TEST () eql FAILURE                           ! VECTOR ADDR AND BR LEVEL TEST
;    2165  2         then
;    2166  3             begin
;    2167  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2168  3             return;
;    2169  2             end;
;    2170  2
;    2171  2         RESULT = HARD_INIT ();                                  ! ATTEMPT HARD DEVICE INIT
;    2172  2         DCT_ADDR [IG_INT] = FALSE;                              ! CLAER "IGNORE INTERRUPT" BIT
;    2173  2
;    2174  2         if .RESULT eql SUCCESS                                  ! IF HARD INIT WAS SUCCESSFUL
;    2175  2         then
```

```
;  2176  3          begin
;  2177  3          ADDR_VECT_OK = TRUE;                    ! ADDRESS/VECTOR TEST PASSED
;  2178  3          INI_RRING ();                           ! INITIALIZE RESPONSE RING
;  2179  3          WRT_RDRX (RCSA, RC_ALL, SA_GO);         ! SET "GO" BIT (START CTLR POLLING)
;  2180  3
;  2181  3          if SET_CTLR_CHAR () eql SUCCESS         ! SET CONTROLLER CHARACTERISTICS
;  2182  3          then
;  2183  4              begin
;  2184  4              DCT_ADDR [STAT] = ONLINE;           ! MARK CONTROLLER ONLINE IN "DRIVER"
;  2185  4              CST_ADDR [STATE] = ONLINE;          ! MARK CONTROLLER ONLINE IN "PROGRAM"
;  2186  3              end;
;  2187  3          end
;  2188  3
;  2189  2      else                                        ! HARD INIT FAILED
;  2190  3          begin
;  2191  3          DROP_CTLR (.CCTLR, DU_INIT);            ! DROP ALL CONTROLLER'S UNITS
;  2192  2          end;
;  2193  2
;  2194  1      end;                                        ! ROUTINE CTLR_INIT


                                .SBTTL  CTLR.INIT INITIALIZATION TEST ROUTINES
000000  010146                 CTLR.INIT::
                                MOV     R1,-(SP)                        ;                       2123
000002  004737  000000V         JSR     PC,INI.CTLR.DAT                 ;                       2146
000006  012746  000200          MOV     #200,-(SP)                      ;                       2147
000012  013700  000000G         MOV     CCTLR,R0
000016  006300                  ASL     R0
000020  016046  000102'          MOV     INT.ADDR(R0),-(SP)
000024  013746  001156'          MOV     CURRENT.VECTOR,-(SP)
000030  012746  000003          MOV     #3,-(SP)
000034  104437                  TRAP    37
000036  052777  040000  000000G BIS     #40000,@DCT.ADDR                ;                       2148
000044  013700  000000G         MOV     CST.ADDR,R0                     ;                       2149
000050  016001  000006          MOV     6(R0),R1
000054  000301                  SWAB    R1
000056  042701  177760          BIC     #177760,R1
000062  010137  000000G         MOV     R1,L$LUN
000066  032737  000002  000000G BIT     #2,DUP.FLAGS                    ;                       2151
000074  001025                  BNE     2$
000076  004737  000000V         JSR     PC,REG.EXIST                    ;                       2154
000102  005700                  TST     R0
000104  001410                  BEQ     1$                              ;                       2157
000106  032737  000002  000000G BIT     #2,DUP.FLAGS                    ;                       2161
000114  001015                  BNE     2$
000116  004737  000000V         JSR     PC,VEC.BR.TEST                  ;                       2164
000122  005700                  TST     R0
000124  001011                  BNE     2$
000126  013716  000000G  1$:     MOV     CCTLR,(SP)                      ;                       2167
000132  012746  000002          MOV     #2,-(SP)
000136  004737  000000G         JSR     PC,DROP.CTLR
000142  062706  000012          ADD     #12,SP                          ;                       2168
000146  000453                  BR      5$                              ;                       2166
```

```
000150  004737  000000V                    2$:     JSR     PC,HARD.INIT                    ;                       2171
000154  110001                                      MOVB    R0,R1                           ; *,RESULT
000156  042777  040000  000000G                    BIC     #40000,@DCT.ADDR                ;                       2172
000164  120127  000001                            CMPB    R1,#1                           ; RESULT,*              2174
000170  001031                                    BNE     3$
000172  112737  000001  000000G                    MOVB    #1,ADDR.VECT.OK                 ;                       2177
000200  004737  000000V                            JSR     PC,INI.RRING                    ;                       2178
000204  012701  000001                            MOV     #1,R1                           ; *,RC.REG              2179
000210  013700  000000G                            MOV     RDRX.ADDR,R0
000214  010160  000002                            MOV     R1,2(R0)                        ; RC.REG,*
000220  004737  000000V                            JSR     PC,SET.CTLR.CHAR                ;                       2181
000224  020027  000001                            CMP     R0,#1
000230  001020                                    BNE     4$
000232  052777  100000  000000G                    BIS     #100000,@DCT.ADDR               ;                       2184
000240  013700  000000G                            MOV     CST.ADDR,R0                     ;                       2185
000244  052760  100000  000002                    BIS     #100000,2(R0)
000252  000407                                    BR      4$                              ;                       2174
000254  013716  000000G                    3$:     MOV     CCTLR,(SP)                      ;                       2191
000260  012746  000002                            MOV     #2,-(SP)
000264  004737  000000G                            JSR     PC,DROP.CTLR
000270  005726                                    TST     (SP)+                           ;                       2190
000272  062706  000010                    4$:     ADD     #10,SP                          ;                       2141
000276  012601                            5$:     MOV     (SP)+,R1                        ;                       2123
000300  000207                                    RTS     PC

; Routine Size:  97 words,      Routine Base:  $CODE$ + 1630
; Maximum stack depth per invocation:  7 words
```

```
;   2195  1     GLOBAL routine INI_CTLR_DAT : novalue =
;   2196  1
;   2197  1     !+
;   2198  1     !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER-RELATED
;   2199  1     !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;   2200  1     !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;   2201  1     !
;   2202  1     !       IMPLICIT INPUTS:
;   2203  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;   2204  1     !               DCT_ADDR - ADDRESS OF CURENT CONTROLLER'S DCT
;   2205  1     !-
;   2206  1
;   2207  2         begin
;   2208  2         DCT_ADDR [WORD0] = 0;                                          ! CLEAR FIRST DCT WORD
;   2209  2         DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;   2210  2         DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4);  ! LAST SLOT IN RESPONSE RING
;   2211  2         DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                    ! START OF COMMAND RING
;   2212  2         DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4);  ! LAST SLOT IN COMMAND RING
;   2213  2         DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG];                       ! FIRST RRING SLOT TO POLL
;   2214  2         DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;   2215  1         end;


                                .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G      INI.CTLR.DAT::
                                JSR     R1,$SAVE2                       ;                       2195
000004  013701  000000G              MOV     DCT.ADDR,R1                     ;                       2208
000010  005011                       CLR     (R1)
000012  012702  000004               MOV     #4,R2                           ;                       2209
000016  060102                       ADD     R1,R2
000020  013746  000000G              MOV     CCTLR,-(SP)
000024  012746  000050               MOV     #50,-(SP)
000030  004737  000000G              JSR     PC,BL$MUL
000034  062700  000010'             ADD     #COMM.AREA+10,R0
000040  010012                       MOV     R0,(R2)
000042  010061  000006               MOV     R0,6(R1)                        ;                       2210
000046  062761  000014  000006       ADD     #14,6(R1)
000054  012700  000010               MOV     #10,R0                          ;                       2211
000060  060100                       ADD     R1,R0
000062  016110  000006               MOV     6(R1),(R0)
000066  062710  000004               ADD     #4,(R0)
000072  011061  000012               MOV     (R0),12(R1)                     ;                       2212
000076  062761  000014  000012       ADD     #14,12(R1)
000104  011261  000014               MOV     (R2),14(R1)                     ;                       2213
000110  011061  000020               MOV     (R0),20(R1)                     ;                       2214
000114  011061  000016               MOV     (R0),16(R1)
000120  022626                       CMP     (SP)+,(SP)+                     ;                       2207
000122  000207                       RTS     PC                              ;                       2195

; Routine Size:  42 words,      Routine Base:  $CODE$ + 2132
; Maximum stack depth per invocation:  6 words
```

```
;    2216  1     GLOBAL routine REG_EXIST =
;    2217  1     !+
;    2218  1     !       THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
;    2219  1     !       THE PRESENCE OF AN RDRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
;    2220  1     !       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
;    2221  1     !       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
;    2222  1     !       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
;    2223  1     !       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
;    2224  1     !       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
;    2225  1     !-
;    2226  2         begin
;    2227  2
;    2228  2         local
;    2229  2             DUMMY_0 : word,                         ! TEMP FOR READING SA AND IP
;    2230  2             DUMMY_1 : word;                         !
;    2231  2
;    2232  2         if .ENTRY_REASON eql NEW_PASS
;    2233  2         then
;    2234  2             return SUCCESS;                         ! SKIP TEST FOR NEXT PASS
;    2235  2
;    2236  2         OF_RC = 2;                                  ! SET UP TO READ SA FIRST
;    2237  2
;    2238  2         do
;    2239  3             begin
;    2240  3             NEX = FALSE;                            ! SET TO "TRAP NOT RECEIVED"
;    2241  3             SETVEC (4, NEX_TRAP, PRI07);            ! SET LOCATION 4 TRAP VECTOR ADDRESS
;    2242  3             DUMMY_0 = .(.RDRX_ADDR + .OF_RC);       ! READ REGISTER (THEN TRAP OR CONTINUE)
;    2243  3             DUMMY_1 = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
;    2244  3                                                     !       (TRAP RETURNS TO NEXT INSTRUCTION)
;    2245  3             CLRVEC (4);                             ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
;    2246  3
;    2247  3             if .NEX                                 ! IF NEX TRAP OCCURRED
;    2248  3             then
;    2249  4                 begin
;    2250  4                 C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2251  4
;    2252  4                 if .APT_MODE
;    2253  4                 then
;    2254  5                     begin
;    2255  5                     .MAIL_BOX_TESTNUM = 1;
;    2256  5                     .MAIL_BOX_SUBTST = 0;
;    2257  4                     end;
;    2258  4
;    2259  4                 ERRDF (10, EGD_10, EMS_10);         ! REGISTER EXISTENCE TEST FAILED
;    2260  4                 SETPRI (PRI00);                     ! LOWER PRIORITY
;    2261  4                 return FAILURE;
;    2262  4                 end
;    2263  3             else
;    2264  3                 OF_RC = .OF_RC - 2;                 ! SET UP FOR IP REG OR QUIT
;    2265  3
;    2266  3             end
;    2267  2         until .OF_RC lss 0;
;    2268  2
```

```
;  2269  2        return SUCCESS;
;  2270  1        end;

                                        .SBTTL  REG.EXIST INITIALIZATION TEST ROUTINES
000000  C04137  000000G          REG.EXIST::
                                        JSR     R1,$SAVE2                       ;                       2216
000004  123727  000000G 000005          CMPB    ENTRY.REASON,#5                 ;                       2232
000012  001472                          BEQ     4$                              ;                       2234
000014  012737  000002  000000G         MOV     #2,OF.RC                        ;                       2236
000022  005037  000000G          1$:    CLR     NEX                             ;                       2240
000026  012746  000340                  MOV     #340,-(SP)                      ;                       2241
000032  012746  000000G                 MOV     #NEX.TRAP,-(SP)
000036  012746  000004                  MOV     #4,-(SP)
000042  012746  000003                  MOV     #3,-(SP)
000046  104437                          TRAP    37
000050  013700  000000G                 MOV     RDRX.ADDR,R0                    ;                       2242
000054  063700  000000G                 ADD     OF.RC,R0
000060  011001                          MOV     (R0),R1                         ; *,DUMMY.0
000062  005002                          CLR     R2                              ; DUMMY.1               2243
000064  012700  000004                  MOV     #4,R0                           ;                       2245
000070  104436                          TRAP    36
000072  032737  000001  000000G         BIT     #1,NEX                          ;                       2247
000100  001427                          BEQ     3$
000102  013700  000000G                 MOV     CCTLR,R0                        ;                       2250
000106  006300                          ASL     R0
000110  105260  C00000G                 INCB    C.ERR.TBL(R0)
000114  032737  000001  001166'         BIT     #1,APT.MODE                     ;                       2252
000122  001405                          BEQ     2$
000124  012777  000001  001170'         MOV     #1,@MAIL.BOX.TESTNUM            ;                       2255
000132  005077  001172'                 CLR     @MAIL.BOX.SUBTST                ;                       2256
000136  104455                  2$:     TRAP    55                              ;                       2259
000140  000012                          .WORD   12
000142  000000G                         .WORD   EGD.10
000144  000000G                         .WORD   EMS.10
000146  005000                          CLR     R0                              ;                       2260
000150  104441                          TRAP    41
000152  062706  000010                  ADD     #10,SP                          ;                       2261
000156  000413                          BR      5$                              ;                       2249
000160  162737  000002  000000G  3$:    SUB     #2,OF.RC                        ;                       2264
000166  062706  000010                  ADD     #10,SP                          ;                       2239
000172  005737  000000G                 TST     OF.RC                           ;                       2267
000176  002311                          BGE     1$
000200  012700  000001          4$:     MOV     #1,R0                           ;                       2226
000204  000207                          RTS     PC
000206  005000                  5$:     CLR     R0                              ;                       2216
000210  000207                          RTS     PC

; Routine Size:  69 words,      Routine Base:  $CODE$ + 2256
; Maximum stack depth per invocation:  9 words
```

```
;    2271  1     GLOBAL routine VEC_BR_TEST =
;    2272  1
;    2273  1     !+
;    2274  1     !       THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RDRX VECTOR ADDRESS GIVEN
;    2275  1     !       BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;    2276  1     !       USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;    2277  1     !       OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;    2278  1     !       AN RDRX INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;    2279  1     !       THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;    2280  1     !       "FUNCTIONAL TEST STARTED" IS PRINTED BEFORE THE TEST, AND
;    2281  1     !       "EXERCISER STARTED" IS PRINTED AT ITS SUCCESSFUL CONCLUSION. IF
;    2282  1     !       EITHER "FUNCTIONAL TEST ..." OR "EXERCISER ..." DOES NOT APPEAR, THEN
;    2283  1     !       PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP IS LIKELY TO OCCUR. AT
;    2284  1     !       THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;    2285  1     !
;    2286  1     !       IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;    2287  1     !       PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;    2288  1     !       USER. A FORCED INTERRUPT SHOULD NOT OCCURR. THEN, BY LOWERING THE
;    2289  1     !       PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;    2290  1     !-
;    2291  1
;    2292  2         begin
;    2293  2
;    2294  2         if .ENTRY_REASON eql NEW_PASS
;    2295  2         then
;    2296  3             begin
;    2297  3             SETPRI (PRI00);                         ! LOWER PRIORITY
;    2298  3             return SUCCESS;                         ! SKIP TEST IF NEXT PASS
;    2299  2             end;
;    2300  2
;    2301  2         PRINTF (MSG_02);                            ! "FUNCTIONAL TEST STARTED"
;    2302  2
;    2303  2         if INT_GEN () eql FALSE                     ! FORCE AN INTERRUPT
;    2304  2         then
;    2305  3             begin                                   ! IF INTERRUPT DID NOT OCCUR
;    2306  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2307  3
;    2308  3             if .APT_MODE
;    2309  3             then
;    2310  4                 begin
;    2311  4                 .MAIL_BOX_TESTNUM = 1;
;    2312  4                 .MAIL_BOX_SUBTST = 0;
;    2313  3                 end;
;    2314  3
;    2315  3             ERRDF (11, EGD_11, 0);                  ! VECTOR TEST FAILED
;    2316  3             return FAILURE;
;    2317  3             end
;    2318  2         else
;    2319  3             begin                                   ! INTERRUPT DID OCCUR
;    2320  3             PRINTF (MSG_03);                        ! "EXERCISER STARTED"
;    2321  3             SETPRI (.BRLEVEL);                      ! SET PRIORITY TO ASSUMED BR LEVEL
;    2322  3
;    2323  3             if INT_GEN () eql FALSE                 ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
```

```
;    2324  3              then
;    2325  4                  begin                           ! IF INTERRUPT DID NOT OCCUR
;    2326  4                  SETPRI (.BRLEVEL - %o'40');     ! LOWER PRIORITY BY 1
;    2327  4                  DELAY (1);                      ! WAIT
;    2328  4
;    2329  4                  if .DCT_ADDR [SA_SAVE] neq 0    ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON-ZERO)
;    2330  4                  then
;    2331  5                      begin
;    2332  5                      SETPRI (PRI00);             ! RESTORE PROCESSOR PRIORITY TO 0
;    2333  5                      return SUCCESS;             ! ONLY SUCCESSFUL EXIT POINT
;    2334  4                      end;
;    2335  4
;    2336  3                  end;
;    2337  3
;    2338  2              end;
;    2339  2
;    2340  2          SETPRI (PRI00);                         ! COME HERE ONLY FOR BR TEST FAILURE
;    2341  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2342  2
;    2343  2          if .APT_MODE
;    2344  2          then
;    2345  3              begin
;    2346  3              .MAIL_BOX_TESTNUM = 1;
;    2347  3              .MAIL_BOX_SUBTST = 0;
;    2348  2              end;
;    2349  2
;    2350  2          ERRDF (12, EGD_12, EMS_12);
;    2351  2          return FAILURE;
;    2352  1          end;


                                        .GLOBL  L$DLY


                                        .SBTTL  VEC.BR.TEST INITIALIZATION TEST ROUTINES
000000  010146                  VEC.BR.TEST::
                                        MOV     R1,-(SP)                        ;                       2271
000002  005746                          TST     -(SP)
000004  123727  000000G 000005         CMPB    ENTRY.REASON,#5                 ;                       2294
000012  001003                          BNE     1$
000014  005000                          CLR     R0                              ;                       2297
000016  104441                          TRAP    41
000020  000504                          BR      8$                              ;                       2296
000022  012746  000000G         1$:     MOV     #MSG.02,-(SP)                   ;                       2301
000026  012746  000001                  MOV     #1,-(SP)
000032  010600                          MOV     SP,R0                           ; SP,*
000034  104417                          TRAP    17
000036  004737  000000V                 JSR     PC,INT.GEN                      ;                       2303
000042  005700                          TST     R0
000044  001023                          BNE     3$
000046  013700  000000G                 MOV     CCTLR,R0                        ;                       2306
000052  006300                          ASL     R0
000054  105260  000000G                 INCB    C.ERR.TBL(R0)
```

```
000060  032737  000001  001166'          BIT     #1,APT.MODE                     ;                       2308
000066  001405                           BEQ     2$
000070  012777  000001  001170'          MOV     #1,@MAIL.BOX.TESTNUM            ;                       2311
000076  005077  001172'                  CLR     @MAIL.BOX.SUBTST                ;                       2312
000102  104455                  2$:     TRAP    55                              ;                       2315
000104  000013                           .WORD   13
000106  000000G                          .WORD   EGD.11
000110  000000                           .WORD   0
000112  000477                           BR      11$                             ;                       2316
000114  012716  000000G           3$:     MOV     #MSG.03,(SP)                    ;                       2320
000120  012746  000001                   MOV     #1,-(SP)
000124  010600                           MOV     SP,R0                           ; SP,*
000126  104417                           TRAP    17
000130  013700  001160'                  MOV     BRLEVEL,R0                      ;                       2321
000134  104441                           TRAP    41
000136  004737  000000V                  JSR     PC,INT.GEN                      ;                       2323
000142  005700                           TST     R0
000144  001035                           BNE     9$
000146  013700  001160'                  MOV     BRLEVEL,R0                      ;                       2326
000152  162700  000040                   SUB     #40,R0
000156  104441                           TRAP    41
000160  012701  000001                   MOV     #1,R1                           ; *,$$TMP2              2327
000164  001411                  4$:     BEQ     7$
000166  013700  000000G                  MOV     L$DLY,R0                        ; *,$$TMP1
000172  001404                           BEQ     6$
000174  005066  000006          5$:     CLR     6(SP)                           ; $$TMP
000200  005300                           DEC     R0                              ; $$TMP1
000202  001374                           BNE     5$
000204  005301                  6$:     DEC     R1                              ; $$TMP2
000206  000766                           BR      4$
000210  013700  000000G           7$:     MOV     DCT.ADDR,R0                     ;                       2329
000214  005760  000002                   TST     2(R0)
000220  001407                           BEQ     9$
000222  005000                           CLR     R0                              ;                       2332
000224  104441                           TRAP    41
000226  062706  000006                   ADD     #6,SP                           ;                       2333
000232  012700  000001          8$:     MOV     #1,R0                           ;                       2331
000236  000427                           BR      12$
000240  005726                  9$:     TST     (SP)+                           ;                       2319
000242  005000                           CLR     R0                              ;                       2340
000244  104441                           TRAP    41
000246  013700  000000G                  MOV     CCTLR,R0                        ;                       2341
000252  006300                           ASL     R0
000254  105260  000000G                  INCB    C.ERR.TBL(R0)
000260  032737  000001  001166'          BIT     #1,APT.MODE                     ;                       2343
000266  001405                           BEQ     10$
000270  012777  000001  001170'          MOV     #1,@MAIL.BOX.TESTNUM            ;                       2346
000276  005077  001172'                  CLR     @MAIL.BOX.SUBTST                ;                       2347
000302  104455                  10$:    TRAP    55                              ;                       2350
000304  000014                           .WORD   14
000306  000000G                          .WORD   EGD.12
000310  000000G                          .WORD   EMS.12
000312  022626                  11$:    CMP     (SP)+,(SP)+                     ;                       2351
```

```
000314  005000                          CLR     R0                      ;                       2271
000316  005726                  12$:    TST     (SP)+
000320  012601                          MOV     (SP)+,R1
000322  000207                          RTS     PC

; Routine Size:  106 words,    Routine Base:  $CODE$ + 2470
; Maximum stack depth per invocation:  7 words
```

```
;    2353  1     GLOBAL routine INT_GEN =
;    2354  1
;    2355  1     !+
;    2356  1     !       THIS ROUTINE BEGINS AN RDRX INITIALIZATION SEQUENCE, BUT ONLY
;    2357  1     !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RDRX
;    2358  1     !       INTERRUPT (AT THE COMPLETEION OF STEP 1) IN ORDER TO HELP VERIFY THE
;    2359  1     !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;    2360  1     !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;    2361  1     !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;    2362  1     !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;    2363  1     !-
;    2364  1
;    2365  2         begin
;    2366  2
;    2367  2         local
;    2368  2             SA : word;                                  ! STORAGE FOR STEP 1 READ AND WRITE
;    2369  2
;    2370  2         DCT_ADDR [SA_SAVE] = 0;                         ! ZERO OUT SA SAVE WORD IN DCT
;    2371  2         WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;    2372  2         DELAY (2);                                      ! WAIT
;    2373  2         SA = .RDRX_ADDR [RCSA, RC_ALL];                 ! STEP 1 READ
;    2374  2         SA = (WR_RING ^ 8) or (.CURRENT_VECTOR ^ -2) or SA_INT;     ! STEP 1 WRITE VALUE
;    2375  2         WRT_RDRX (RCSA, RC_ALL, .SA);                   ! STEP 1 WRITE
;    2376  2
;    2377  2         incr COUNT from 1 to 8000 do
;    2378  3             begin
;    2379  3             DELAY (1);                                  ! TOTAL DELAY COUNT OF 8,000
;    2380  3
;    2381  3             if .DCT_ADDR [SA_SAVE] neq 0                ! IF SA WAS CHANGED
;    2382  3             then
;    2383  3                 return TRUE;                            ! INTERRUPT OCCURED
;    2384  3
;    2385  3             BREAK;
;    2386  2             end;
;    2387  2
;    2388  2         return FALSE;                                   ! IF INTERRUPT DID NOT OCCUR
;    2389  1         end;


                                        .SBTTL  INT.GEN INITIALIZATION TEST ROUTINES
000000  004137  000000G          INT.GEN::
                                        JSR     R1,$SAVE2                       ;                       2353
000004  024646                          CMP     -(SP),-(SP)
000006  013700  000000G                 MOV     DCT.ADDR,R0                     ;                       2370
000012  005060  000002                  CLR     2(R0)
000016  012700  177777                  MOV     #-1,R0                          ; *,RC.REG              2371
000022  010077  000000G                 MOV     R0,@RDRX.ADDR                   ; RC.REG,*
000026  012701  000002                  MOV     #2,R1                           ; *,$$TMP2              2372
000032  001411                  1$:     BEQ     4$
000034  013700  000000G                 MOV     L$DLY,R0                        ; *,$$TMP1
000040  001404                          BEQ     3$
000042  005066  000002          2$:     CLR     2(SP)                           ; $$TMP
000046  005300                          DEC     R0                              ; $$TMP1
```

```
000050  001374                          BNE     2$
000052  005301                  3$:     DEC     R1                      ; $$TMP2
000054  000766                          BR      1$
000056  013700  000000G          4$:     MOV     RDRX.ADDR,R0            ;                               2373
000062  016016  000002                  MOV     2(R0),(SP)              ; *,RC.REG
000066  013701  001156'                  MOV     CURRENT.VECTOR,R1       ;                               2374
000072  006201                          ASR     R1
000074  006201                          ASR     R1
000076  010102                          MOV     R1,R2                   ; *,SA
000100  052702  111200                  BIS     #111200,R2              ; *,SA
000104  010201                          MOV     R2,R1                   ; SA,RC.REG                     2375
000106  010160  000002                  MOV     R1,2(R0)                ; RC.REG,*
000112  012702  017500                  MOV     #17500,R2               ; *,COUNT                       2377
000116  012701  000001          5$:     MOV     #1,R1                   ; *,$$TMP2                      2379
000122  001411                  6$:     BEQ     9$
000124  013700  000000G                  MOV     L$DLY,R0                ; *,$$TMP1
000130  001404                          BEQ     8$
000132  005066  000002          7$:     CLR     2(SP)                   ; $$TMP
000136  005300                          DEC     R0                      ; $$TMP1
000140  001374                          BNE     7$
000142  005301                  8$:     DEC     R1                      ; $$TMP2
000144  000766                          BR      6$
000146  013700  000000G          9$:     MOV     DCT.ADDR,R0             ;                               2381
000152  005760  000002                  TST     2(R0)
000156  001403                          BEQ     10$
000160  012700  000001                  MOV     #1,R0                   ;                               2383
000164  000404                          BR      11$
000166  104422                  10$:    TRAP    22
000170  005302                          DEC     R2                      ; COUNT                         2377
000172  001351                          BNE     5$
000174  005000                          CLR     R0                      ;                               2365
000176  022626                  11$:    CMP     (SP)+,(SP)+             ;                               2353
000200  000207                          RTS     PC

; Routine Size:  65 words,      Routine Base:  $CODE$ + 3014
; Maximum stack depth per invocation:  7 words
```

```
;    2390  1     GLOBAL routine HARD_INIT =
;    2391  1
;    2392  1     !+
;    2393  1     !       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;    2394  1     !       INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;    2395  1     !       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;    2396  1     !       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;    2397  1     !       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;    2398  1     !-
;    2399  1
;    2400  2         begin
;    2401  2
;    2402  2         local
;    2403  2             IE_VEC : word;                                  ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
;    2404  2                                                             !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;    2405  2
;    2406  2         IE_VEC = .CURRENT_VECTOR + -2;                      ! GET VECTOR ADDR/4 (IE = 0)
;    2407  2
;    2408  2         incr ATTEMPTS from 1 to INI_ATT do
;    2409  3             begin
;    2410  3
;    2411  3             label
;    2412  3                 STEP_1_READ,
;    2413  3                 STEP_2_READ,
;    2414  3                 STEP_3_READ,
;    2415  3                 STEP_4_READ;
;    2416  3
;    2417  3             WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;    2418  3     !
;    2419  3     !       STEP 1 READ
;    2420  3     !
;    2421  3             STEP = 1;
;    2422  3             STEP_1_READ:
;    2423  4                 begin
;    2424  4
;    2425  4                 incr COUNT from 1 to 500 do
;    2426  5                     begin
;    2427  5                     DELAY (1);                              ! TOTAL DELAY COUNT OF 500 FOR STEP 1
;    2428  5                     SA_REG = .RDRX_ADDR [RCSA, RC_ALL];     ! READ SA
;    2429  5
;    2430  5                     if (.SA_REG and S1_MASK) eql SA_S1      ! IF STEP 1 READ IS O.K.
;    2431  5                     then
;    2432  5                         leave STEP_1_READ;
;    2433  5
;    2434  5                     BREAK;
;    2435  4                     end;
;    2436  4
;    2437  4                 exitloop;
;    2438  3                 end;
;    2439  3
;    2440  3     !
;    2441  3     !       STEP 1 WRITE
;    2442  3     !
```

```
;  2443  3              SA_REG = (WR_RING ^ 8) or .IE_VEC;                  ! STEP 1 WRITE VALUE
;  2444  3              WRT_RDRX (RCSA, RC_ALL, .SA_REG);                   ! STEP 1 WRITE
;  2445  3      !
;  2446  3      !       STEP 2 READ
;  2447  3      !
;  2448  3              STEP = .STEP + 1;
;  2449  3              STEP_2_READ:
;  2450  4                  begin
;  2451  4
;  2452  4                  incr COUNT from 1 to 10000 do
;  2453  5                      begin
;  2454  5                      DELAY (1);                                  ! TOTAL DELAY COUNT OF 10,000 FOR STEP 2
;  2455  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];         ! READ SA
;  2456  5
;  2457  6                      if (.SA_REG and S2_MASK) eql (SA_S2 or WR_RING) ! IF STEP 2 READ IS O.K.
;  2458  5                      then
;  2459  5                          leave STEP_2_READ;
;  2460  5
;  2461  5                      BREAK;
;  2462  4                      end;
;  2463  4
;  2464  4                  exitloop;
;  2465  3                  end;
;  2466  3
;  2467  3      !
;  2468  3      !       STEP 2 WRITE
;  2469  3      !
;  2470  3              WRT_RDRX (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]);        ! RINGBASE-LO, PI = 0
;  2471  3      !
;  2472  3      !       STEP 3 READ
;  2473  3      !
;  2474  3              STEP = .STEP + 1;
;  2475  3              STEP_3_READ:
;  2476  4                  begin
;  2477  4
;  2478  4                  incr COUNT from 1 to 10000 do
;  2479  5                      begin
;  2480  5                      DELAY (1);                                  ! TOTAL DELAY COUNT OF 10,000 FOR STEP 3 READ
;  2481  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];         ! READ SA
;  2482  5
;  2483  6                      if (.SA_REG and S3_MASK) eql (SA_S3 or .IE_VEC) ! IF STEP 3 READ IS O.K.
;  2484  5                      then
;  2485  5                          leave STEP_3_READ;
;  2486  5
;  2487  5                      BREAK;
;  2488  4                      end;
;  2489  4
;  2490  4                  exitloop;
;  2491  3                  end;
;  2492  3
;  2493  3      !
;  2494  3      !       STEP 3 WRITE
;  2495  3      !
```

```
;    2496  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! PP, RINGBASE-HI = 0
;    2497  3      !
;    2498  3      !       STEP 4 READ
;    2499  3      !
;    2500  3              STEP = .STEP + 1;
;    2501  3              STEP_4_READ:
;    2502  4                  begin
;    2503  4
;    2504  4                  incr COUNT from 1 to 10000 do
;    2505  5                      begin
;    2506  5                      DELAY (1);                                   ! TOTAL DELAY COUNT OF 10,000 FOR STEP 4 READ
;    2507  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];          ! READ SA
;    2508  5
;    2509  5                      if (.SA_REG and S4_MASK) eql SA_S4           ! IF STEP 4 READ IS O.K.
;    2510  5                      then
;    2511  5                          leave STEP_4_READ;
;    2512  5
;    2513  5                      BREAK;
;    2514  4                      end;
;    2515  4
;    2516  4                  exitloop;
;    2517  3                  end;
;    2518  3      !
;    2519  3      !       STEP 4 WRITE
;    2520  3      !
;    2521  3              CREDIT_BAL = 1;                                      ! START WITH A CREDIT BALANCE = 1
;    2522  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! BURST, LF, GO = 0
;    2523  3              return SUCCESS;                                      ! SUCCESS EXIT POINT
;    2524  3
;    2525  2                  end;                                             ! TRY AGAIN OR GIVE UP
;    2526  2
;    2527  2          CREDIT_BAL = 0;                                          ! NO CREDIT BALANCE
;    2528  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2529  2
;    2530  2          if .APT_MODE
;    2531  2          then
;    2532  3              begin
;    2533  3              .MAIL_BOX_TESTNUM = 1;
;    2534  3              .MAIL_BOX_SUBTST = 0;
;    2535  2              end;
;    2536  2
;    2537  2          ERRDF (13, EGD_13, EMS_13);                              ! INIT SEQUENCE FAILED
;    2538  2          return FAILURE;
;    2539  1          end;                                                     ! ROUTINE HARD_INIT


                                        .SBTTL  HARD.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G          HARD.INIT::
                                        JSR     R1,$SAVE5                    ;                                  2390
000004  162706  000012                  SUB     #12,SP
000010  013704  001156'                 MOV     CURRENT.VECTOR,R4            ; *,IE.VEC                         2406
000014  006204                          ASR     R4                           ; IE.VEC
000016  006204                          ASR     R4                           ; IE.VEC
```

```
000020  012705  000002                        MOV     #2,R5                   ; *,ATTEMPTS                    2408
000024  012700  177777                        MOV     #-1,R0                  ; *,RC.REG                      2417
000030  010077  000000G                       MOV     R0,@RDRX.ADDR           ; RC.REG,*
000034  012737  000001  000000G               MOV     #1,STEP                 ;                               2421
000042  012702  000764                        MOV     #764,R2                 ; *,COUNT                       2425
000046  012701  000001          1$:     MOV     #1,R1                   ; *,$$TMP2                      2427
000052  001411                  2$:     BEQ     5$
000054  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000060  001404                        BEQ     4$
000062  005066  000010          3$:     CLR     10(SP)                  ; $$TMP
000066  005300                        DEC     R0                      ; $$TMP1
000070  001374                        BNE     3$
000072  005301                  4$:     DEC     R1                      ; $$TMP2
000074  000766                        BR      2$
000076  013700  000000G           5$:     MOV     RDRX.ADDR,R0            ;                               2428
000102  016016  000002                        MOV     2(R0),(SP)              ; *,RC.REG
000106  011637  000000G                       MOV     (SP),SA.REG             ; RC.REG,*
000112  011600                        MOV     (SP),R0                 ; SA.REG,*                      2430
000114  042700  001777                        BIC     #1777,R0
000120  020027  004000                        CMP     R0,#4000
000124  001404                        BEQ     6$                      ;                               2432
000126  104422                        TRAP    22
000130  005302                        DEC     R2                      ; COUNT                         2425
000132  001345                        BNE     1$
000134  000532                        BR      18$                     ;                               2409
000136  010437  000000G           6$:     MOV     R4,SA.REG               ; IE.VEC,*                      2443
000142  052737  111000  000000G               BIS     #111000,SA.REG
000150  013701  000000G                       MOV     SA.REG,R1               ; *,RC.REG                      2444
000154  013700  000000G                       MOV     RDRX.ADDR,R0
000160  010160  000002                        MOV     R1,2(R0)                ; RC.REG,*
000164  005237  000000G                       INC     STEP                    ;                               2448
000170  012702  023420                        MOV     #23420,R2               ; *,COUNT                       2452
000174  012701  000001          7$:     MOV     #1,R1                   ; *,$$TMP2                      2454
000200  001411                  8$:     BEQ     11$
000202  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000206  001404                        BEQ     10$
000210  005066  000010          9$:     CLR     10(SP)                  ; $$TMP
000214  005300                        DEC     R0                      ; $$TMP1
000216  001374                        BNE     9$
000220  005301                  10$:    DEC     R1                      ; $$TMP2
000222  000766                        BR      8$
000224  013700  000000G           11$:    MOV     RDRX.ADDR,R0            ;                               2455
000230  016066  000002  000002               MOV     2(R0),2(SP)             ; *,RC.REG
000236  016637  000002  000000G               MOV     2(SP),SA.REG            ; RC.REG,*
000244  016600  000002                        MOV     2(SP),R0                ; SA.REG,*                      2457
000250  042700  003400                        BIC     #3400,R0
000254  020027  010222                        CMP     R0,#10222
000260  001404                        BEQ     12$                     ;                               2459
000262  104422                        TRAP    22
000264  005302                        DEC     R2                      ; COUNT                         2452
000266  001342                        BNE     7$
000270  000537                        BR      26$                     ;                               2409
000272  013700  000000G           12$:    MOV     DCT.ADDR,R0             ;                               2470
```

J6


```
000276  016001  000004                          MOV     4(R0),R1                ; *,RC.REG
000302  013700  000000G                         MOV     RDRX.ADDR,R0
000306  010160  000002                          MOV     R1,2(R0)                ; RC.REG,*
000312  005237  000000G                         INC     STEP                    ;                               2474
000316  010403                                  MOV     R4,R3                   ; IE.VEC,*                      2483
000320  052703  020000                          BIS     #20000,R3
000324  012702  023420                          MOV     #23420,R2               ; *,COUNT                       2478
000330  012701  000001                  13$:    MOV     #1,R1                   ; *,$$TMP2                      2480
000334  001411                          14$:    BEQ     17$
000336  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000342  001404                                  BEQ     16$
000344  005066  000010                  15$:    CLR     10(SP)                  ; $$TMP
000350  005300                                  DEC     R0                      ; $$TMP1
000352  001374                                  BNE     15$
000354  005301                          16$:    DEC     R1                      ; $$TMP2
000356  000766                                  BR      14$
000360  013700  000000G                 17$:    MOV     RDRX.ADDR,R0            ;                               2481
000364  016066  000002  000004                  MOV     2(R0),4(SP)             ; *,RC.REG
000372  016637  000004  000000G                 MOV     4(SP),SA.REG            ; RC.REG,*
000400  016600  000004                          MOV     4(SP),R0                ; SA.REG,*                      2483
000404  042700  003400                          BIC     #3400,R0
000410  020003                                  CMP     R0,R3
000412  001404                                  BEQ     19$                     ;                               2485
000414  104422                                  TRAP    22
000416  005302                                  DEC     R2                      ; COUNT                         2478
000420  001343                                  BNE     13$
000422  000462                          18$:    BR      26$                     ;                               2409
000424  013700  000000G                 19$:    MOV     RDRX.ADDR,R0            ;                               2496
000430  005060  000002                          CLR     2(R0)
000434  005237  000000G                         INC     STEP                    ;                               2500
000440  012703  023420                          MOV     #23420,R3               ; *,COUNT                       2504
000444  012701  000001                  20$:    MOV     #1,R1                   ; *,$$TMP2                      2506
000450  001411                          21$:    BEQ     24$
000452  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000456  001404                                  BEQ     23$
000460  005066  000010                  22$:    CLR     10(SP)                  ; $$TMP
000464  005300                                  DEC     R0                      ; $$TMP1
000466  001374                                  BNE     22$
000470  005301                          23$:    DEC     R1                      ; $$TMP2
000472  000766                                  BR      21$
000474  013700  000000G                 24$:    MOV     RDRX.ADDR,R0            ;                               2507
000500  016066  000002  000006                  MOV     2(R0),6(SP)             ; *,RC.REG
000506  016637  000006  000C00G                 MOV     6(SP),SA.REG            ; RC.REG,*
000514  016600  000006                          MOV     6(SP),R0                ; SA.REG,*                      2509
000520  042700  003777                          BIC     #3777,R0
000524  020027  040000                          CMP     R0,#40000
000530  001404                                  BEQ     25$                     ;                               2511
000532  104422                                  TRAP    22
000534  005303                                  DEC     R3                      ; COUNT                         2504
000536  001342                                  BNE     20$
000540  000413                                  BR      26$                     ;                               2409
000542  012737  000001  000000G         25$:    MOV     #1,CREDIT.BAL           ;                               2521
000550  005001                                  CLR     R1                      ; RC.REG                        2522
```

```
000552  013700  000000G                        MOV     RDRX.ADDR,R0
000556  005060  000002                         CLR     2(R0)
000562  012700  000001                         MOV     #1,R0                   ;                       2409
000566  000425                                 BR      28$
000570  005037  000000G                26$:    CLR     CREDIT.BAL              ;                       2527
000574  013700  000000G                        MOV     CCTLR,R0                ;                       2528
000600  006300                                 ASL     R0
000602  105260  000000G                        INCB    C.ERR.TBL(R0)
000606  032737  000001  001166'               BIT     #1,APT.MODE             ;                       2530
000614  001405                                 BEQ     27$
000616  012777  000001  001170'               MOV     #1,@MAIL.BOX.TESTNUM    ;                       2533
000624  005077  001172'                        CLR     @MAIL.BOX.SUBTST        ;                       2534
000630  104455                         27$:    TRAP    55                      ;                       2537
000632  000015                                 .WORD   15
000634  000000G                                .WORD   EGD.13
000636  000000G                                .WORD   EMS.13
000640  005000                                 CLR     R0                      ;                       2400
000642  062706  000012                 28$:    ADD     #12,SP                  ;                       2390
000646  000207                                 RTS     PC

; Routine Size:  212 words,     Routine Base:  $CODE$ + 3216
; Maximum stack depth per invocation:  13 words
```

```
;    2540  1     GLOBAL routine INI_RRING : novalue =
;    2541  1
;    2542  1     !+
;    2543  1     !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
;    2544  1     !       FILL AN RDRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
;    2545  1     !       (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
;    2546  1     !       RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
;    2547  1     !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
;    2548  1     !       CONTROLLER-OWNED.
;    2549  1     !
;    2550  1     !       IMPLICIT INPUTS:
;    2551  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2552  1     !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    2553  1     !-
;    2554  1
;    2555  2         begin
;    2556  2
;    2557  2         local
;    2558  2             index : word,
;    2559  2             RRING_ADDR;
;    2560  2
;    2561  2         RRING_ADDR = .DCT_ADDR [RR_BEG];                    ! FIRST RESPONSE RING SLOT
;    2562  2
;    2563  2         incr COUNT from 1 to RRING_LEN do
;    2564  3             begin
;    2565  3             index = GET_PKT (.CCTLR);                       ! GET AN MSCP PACKET
;    2566  3             .RRING_ADDR = .MSCP_PKT [.index, PKT_LO];       ! LOAD LO-ORDER BUFF DESC INTO SLOT
;    2567  3             RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO SECOND WORD
;    2568  3             .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];       ! LOAD HI-ORDER BUFF DESC INTO SLOT
;    2569  3             PKT_USE [.index] = .CCTLR;                      ! PACKET IN USE
;    2570  3             .RRING_ADDR = ..RRING_ADDR or ED_OWN or ED_FLAG;! GIVE OWNERSHIP TO CONTRLLER
;    2571  3             RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO NEXT SLOT
;    2572  2             end;
;    2573  2
;    2574  1         end;


                                        .SBTTL  INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G        INI.RRING::
                                JSR     R1,$SAVE4                       ;                       2540
000004  013700  000000G                MOV     DCT.ADDR,R0             ;                       2561
000010  016001  000004                 MOV     4(R0),R1                ; *,RRING.ADDR
000014  013703  000000G                MOV     CCTLR,R3                ;                       2565
000020  012704  000004                 MOV     #4,R4                   ; *,COUNT               2563
000024  010346                 1$:     MOV     R3,-(SP)                ;                       2565
000026  004737  000000G                JSR     PC,GET.PKT
000032  010002                         MOV     R0,R2                   ; *,INDEX
000034  010216                         MOV     R2,(SP)                 ; INDEX,*               2566
000036  012746  000106                 MOV     #106,-(SP)
000042  004737  000000G                JSR     PC,B!$MUL
000046  016021  000000G                MOV     MSCP.PKT(R0),(R1)+      ; *,RRING.ADDR
000052  016011  000002G                MOV     MSCP.PKT+2(R0),(R1)     ; *,RRING.ADDR          2568
000056  013703  000000G                MOV     CCTLR,R3                ;                       2569
```

M6


```
000062  110362  000000G                    MOVB    R3,PKT.USE(R2)          ; *,*(INDEX)
000066  052721  140000                      BIS     #140000,(R1)+           ; *,RRING.ADDR                  2570
000072  022626                             CMP     (SP)+,(SP)+             ;                               2564
000074  005304                             DEC     R4                      ; COUNT                         2563
000076  001352                             BNE     1$
000100  000207                             RTS     PC                      ;                               2540

; Routine Size:  33 words,      Routine Base:  $CODE$ + 4066
; Maximum stack depth per invocation:  8 words
```

```
;    2575  1     GLOBAL routine SET_CTLR_CHAR =
;    2576  1
;    2577  1     !*
;    2578  1     !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RDRX HAS BEEN HARD-
;    2579  1     !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;    2580  1     !       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
;    2581  1     !
;    2582  1     !       IMPLICIT INPUTS:
;    2583  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2584  1     !-
;    2585  1
;    2586  2         begin
;    2587  2
;    2588  2         local
;    2589  2             P_INDEX : word;
;    2590  2
;    2591  2
;    2592  2     ! MISCELLANEOUS INITIALIZATION                                   !ZZZ
;    2593  2                                                                     !ZZZ
;    2594  2     QIO [.CCTLR] = 0;               !INIT NO OF OUTSTANDING QIOS    !ZZZ
;    2595  2     CST [.CCTLR, U_CNT] = 0;        !CLEAR UNITS IN CST TABLE       !ZZZ
;    2596  2     INCR COUNT FROM 0 TO (RP_CNT - 1) DO    !INIT RETURN PACKET POOL !ZZZ
;    2597  2          RP_USE [.COUNT] = -1;                                      !ZZZ
;    2598  2                                                                     !ZZZ
;    2599  2     IODQ_IN = IODQ_OUT = 0;         !INIT I/O DONE QUEUE POINTERS   !ZZZ
;    2600  2
;    2601  2
;    2602  2         P_INDEX = GET_PKT (.CCTLR);                                 ! GET AN MSCP PACKET
;    2603  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                       ! PACKET SIZE
;    2604  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                       ! OPCODE = SET CTLR CHAR
;    2605  2         MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                     ! CONTROLLER FLAGS
;    2606  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = IMM_CMD;                    ! IMMEDIATE COMMAND
;    2607  2
;    2608  2         if SEND (.P_INDEX) eql FAILURE                              ! ATTEMPT SEND
;    2609  2         then
;    2610  3             begin                                                   ! IF SEND WAS UNSUCCESSFUL
;    2611  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2612  3
;    2613  3             if .APT_MODE
;    2614  3             then
;    2615  4                 begin
;    2616  4                 .MAIL_BOX_TESTNUM = 1;
;    2617  4                 .MAIL_BOX_SUBTST = 0;
;    2618  3                 end;
;    2619  3
;    2620  3             ERRDF (20, EGD_20, 0);                                  ! FATAL ERROR
;    2621  3             PUT_PKT (.P_INDEX);                                     ! RETURN PACKET TO POOL
;    2622  3             DROP_CTLR (.CCTLR, DU_CFATAL);                          ! DROP CONTROLLER
;    2623  3             return FAILURE;
;    2624  3             end
;    2625  2         else
;    2626  3             begin                                                   ! IF SEND WAS SUCCESSFUL
;    2627  3
```

```
;  2628  3              do
;  2629  4                  begin
;  2630  4                  WAIT ();                                        ! WAIT FOR RETPKT RESPONSE
;  2631  4                  RP_INDX = OUT_IODQ ();                          ! GET INDEX OF RETPKT
;  2632  4                  RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;  2633  4
;  2634  4                  if .RP_ADDR [MESTYP] neq MT_SEQ                 ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;  2635  4                  then
;  2636  4                      PUT_RETPKT (.RP_INDX);
;  2637  4
;  2638  4                  end
;  2639  3              until (.RP_ADDR [CONID] eql CID_DRIVER) or
;  2640  4                    ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;  2641  3                     ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;  2642  3
;  2643  3              if .RP_ADDR [CONID] eql CID_DRIVER                  ! IF RETPKT IS FROM "DRIVER"
;  2644  3              then
;  2645  4                  begin
;  2646  4                  PRINTF (DBM23);                                 ! "ERROR IN SET_CTLR_CHAR"
;  2647  4                  PUT_RETPKT (.RP_INDX);                          ! RELEASE RETURN PACKET
;  2648  4                  DR_ERR ();                                      ! DROP CONTROLLER
;  2649  4                  return FAILURE;
;  2650  4                  end
;  2651  3              else
;  2652  4                  begin                                           ! ELSE - RETPKT IS FROM DISK MSCP
;  2653  4
;  2654  4                  if (.RP_ADDR [ENDCOD] neq (OP_SCC or OP_END)) or    ! IF WRONG ENDCODE
;  2655  5                     ((.RP_ADDR [C_FLGS] and CF_MASK) neq CF_MASK)    ! OR FLAGS IN ERROR
;  2656  4                  then
;  2657  5                      begin
;  2658  5                      C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  2659  5
;  2660  5                      if .APT_MODE
;  2661  5                      then
;  2662  6                          begin
;  2663  6                          .MAIL_BOX_TESTNUM = 1;
;  2664  6                          .MAIL_BOX_SUBTST = 0;
;  2665  5                          end;
;  2666  5
;  2667  5                      ERRDF (21, EGD_21, EMS_21);                 ! FATAL ERROR
;  2668  5                      DROP_CTLR (.CCTLR, DU_CFATAL);              ! DROP CONTROLLER
;  2669  5                      PUT_RETPKT (.RP_INDX);                      ! RELEASE RETURN PACKET
;  2670  5                      return FAILURE;
;  2671  5                      end
;  2672  4                  else
;  2673  5                      begin                                       ! RETPKT HAS CORRECT ENDCODE
;  2674  5                      CMD_TIME = .RP_ADDR [C_TIME] * 2;
;  2675  5
;  2676  6                      if BIT_TST (SWP_FLAGS, SWF_TRC)
;  2677  5                      then
;  2678  5                          PRINTF (DBM25, .RP_ADDR [C_TIME]);
;  2679  5
;  2680  4                      end;                                        ! RETPKT HAS CORRECT ENDCODE
```

```
;   2681  4
;   2682  3                 end;                                    ! IF RETPKT WAS SENT BY DISK MSCP
;   2683  3
;   2684  3             PUT_RETPKT (.RP_INDX);
;   2685  3             return SUCCESS;                             ! IF SEND WAS SUCCESSFUL
;   2686  2             end;
;   2687  2
;   2688  1         end;                                            ! ROUTINE SET_CTLR_CHAR


                                    .SBTTL  SET.CTLR.CHAR INITIALIZATION TEST ROUTINES
000000  010146                 SET.CTLR.CHAR::
                                    MOV     R1,-(SP)                ;                               2575
000002  013701  000000G             MOV     CCTLR,R1                ;                               2594
000006  105061  000000G             CLRB    QIO(R1)
000012  010146                      MOV     R1,-(SP)                ;                               2595
000014  012746  000126              MOV     #126,-(SP)
000020  004737  000000G             JSR     PC,BL$MUL
000024  105060  000005G             CLRB    CST+5(R0)
000030  005000                      CLR     R0                      ; COUNT                         2596
000032  112760  000377  000000G 1$:  MOVB    #377,RP.USE(R0)         ; *,*(COUNT)                    2597
000040  005200                      INC     R0                      ; COUNT                         2596
000042  020027  000007              CMP     R0,#7                   ; COUNT,*
000046  003771                      BLE     1$
000050  005037  000000G             CLR     IODQ.OUT                ;                               2599
000054  005037  000000G             CLR     IODQ.IN
000060  010116                      MOV     R1,(SP)                 ;                               2602
000062  004737  000000G             JSR     PC,GET.PKT
000066  010001                      MOV     R0,R1                   ; *,P.INDEX
000070  010116                      MOV     R1,(SP)                 ; P.INDEX,*                     2603
000072  012746  000106              MOV     #106,-(SP)
000076  004737  000000G             JSR     PC,BL$MUL
000102  012760  000040  000006G     MOV     #40,MSCP.PKT+6(R0)
000110  112760  000004  000022G     MOVB    #4,MSCP.PKT+22(R0)      ;                               2604
000116  012760  000120  000030G     MOV     #120,MSCP.PKT+30(R0)    ;                               2605
000124  105060  000004G             CLRB    MSCP.PKT+4(R0)          ;                               2606
000130  010116                      MOV     R1,(SP)                 ; P.INDEX,*                     2608
000132  004737  000000G             JSR     PC,SEND
000136  005700                      TST     R0
000140  001036                      BNE     3$
000142  013700  000000G             MOV     CCTLR,R0                ;                               2611
000146  006300                      ASL     R0
000150  105260  000000G             INCB    C.ERR.TBL(R0)
000154  032737  000001  001166'     BIT     #1,APT.MODE             ;                               2613
000162  001405                      BEQ     2$
000164  012777  000001  001170'     MOV     #1,@MAIL.BOX.TESTNUM    ;                               2616
000172  005077  001172'             CLR     @MAIL.BOX.SUBTST        ;                               2617
000176  104455                 2$:  TRAP    55                      ;                               2620
000200  000024                      .WORD   24
000202  000000G                     .WORD   EGD.20
000204  000000                      .WORD   0
000206  010116                      MOV     R1,(SP)                 ; P.INDEX,*                     2621
000210  004737  000000G             JSR     PC,PUT.PKT
```

```
000214  013716  000000G                         MOV     CCTLR,(SP)                      ;                               2622
000220  012746  000006                          MOV     #6,-(SP)
000224  004737  000000G                         JSR     PC,DROP.CTLR
000230  005726                                  TST     (SP)+                           ;                               2610
000232  005000                                  CLR     R0                              ;                               2626
000234  000571                                  BR      13$
000236  004737  000000G                 3$:     JSR     PC,WAIT                         ;                               2630
000242  004737  000000G                         JSR     PC,OUT.IODQ                     ;                               2631
000246  010037  000000G                         MOV     R0,RP.INDX
000252  010016                                  MOV     R0,(SP)                         ; RP.INDX,*                     2632
000254  012746  000054                          MOV     #54,-(SP)
000260  004737  000000G                         JSR     PC,BL$MUL
000264  062700  000000G                         ADD     #RETPKT,R0
000270  010037  000000G                         MOV     R0,RP.ADDR
000274  132760  000360  000002                 BITB    #360,2(R0)                      ;                               2634
000302  001404                                  BEQ     4$
000304  013716  000000G                         MOV     RP.INDX,(SP)                    ;                               2636
000310  004737  000000G                         JSR     PC,PUT.RETPKT
000314  005726                          4$:     TST     (SP)+                           ;                               2629
000316  013701  000000G                         MOV     RP.ADDR,R1                      ;                               2639
000322  005000                                  CLR     R0
000324  126127  000003  000003                 CMPB    3(R1),#3
000332  001002                                  BNE     5$
000334  005200                                  INC     R0
000336  000407                                  BR      6$
000340  132761  000360  000002         5$:     BITB    #360,2(R1)                      ;                               2640
000346  001333                                  BNE     3$
000350  105761  000014                          TSTB    14(R1)                          ;                               2641
000354  100330                                  BPL     3$
000356  006000                          6$:     ROR     R0                              ;                               2643
000360  103015                                  BCC     7$
000362  012716  000000G                         MOV     #DBM23,(SP)                     ;                               2646
000366  012746  000001                          MOV     #1,-(SP)
000372  010600                                  MOV     SP,R0                           ; SP,*
000374  104417                                  TRAP    17
000376  013716  000000G                         MOV     RP.INDX,(SP)                    ;                               2647
000402  004737  000000G                         JSR     PC,PUT.RETPKT
000406  004737  000000V                         JSR     PC,DR.ERR                       ;                               2648
000412  000447                                  BR      10$                             ;                               2649
000414  126127  000014  000204         7$:     CMPB    14(R1),#204                     ;                               2654
000422  001007                                  BNE     8$
000424  016100  000022                          MOV     22(R1),R0                       ;                               2655
000430  042700  177657                          BIC     #177657,R0
000434  020027  000120                          CMP     R0,#120
000440  001437                                  BEQ     11$
000442  013700  000000G                 8$:     MOV     CCTLR,R0                        ;                               2658
000446  006300                                  ASL     R0
000450  105260  000000G                         INCB    C.ERR.TBL(R0)
000454  032737  000001  001166'                 BIT     #1,APT.MODE                     ;                               2660
000462  001405                                  BEQ     9$
000464  012777  000001  001170'                 MOV     #1,@MAIL.BOX.TESTNUM            ;                               2663
000472  005077  001172'                         CLR     @MAIL.BOX.SUBTST                ;                               2664
000476  104455                          9$:     TRAP    55                              ;                               2667
```

```
000500  000025                                  .WORD   25
000502  000000G                                 .WORD   EGD.21
000504  000000G                                 .WORD   EMS.21
000506  013716  000000G                         MOV     CCTLR,(SP)                      ;                       2668
000512  012746  000006                          MOV     #6,-(SP)
000516  004737  000000G                         JSR     PC,DROP.CTLR
000522  013716  000000G                         MOV     RP.INDX,(SP)                    ;                       2669
000526  004737  000000G                         JSR     PC,PUT.RETPKT
000532  062706  000010                  10$:    ADD     #10,SP                          ;                       2670
000536  000433                                  BR      14$                             ;                       2657
000540  016137  000024  000000G       11$:    MOV     24(R1),CMD.TIME                 ;                       2674
000546  006337  000000G                         ASL     CMD.TIME
000552  032737  000001  000000G               BIT     #1,SWP.FLAGS                    ;                       2676
000560  001411                                  BEQ     12$
000562  016116  000024                          MOV     24(R1),(SP)                     ;                       2678
000566  012746  000000G                         MOV     #DBM25,-(SP)
000572  012746  000002                          MOV     #2,-(SP)
000576  010600                                  MOV     SP,R0                           ; SP,*
000600  104417                                  TRAP    17
000602  022626                                  CMP     (SP)+,(SP)+
000604  013716  000000G                 12$:    MOV     RP.INDX,(SP)                    ;                       2684
000610  004737  000000G                         JSR     PC,PUT.RETPKT
000614  012700  000001                          MOV     #1,R0                           ;                       2626
000620  062706  000006                  13$:    ADD     #6,SP                           ;                       2608
000624  000401                                  BR      15$                             ;                       2586
000626  005000                          14$:    CLR     R0                              ;                       2575
000630  012601                          15$:    MOV     (SP)+,R1
000632  000207                                  RTS     PC

; Routine Size:  206 words,    Routine Base:  $CODE$ + 4170
; Maximum stack depth per invocation:  8 words
```

```
;    2689  1     routine UNIT_INIT : novalue =
;    2690  1
;    2691  1     !+
;    2692  1     !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
;    2693  1     !       (DISK) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
;    2694  1     !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
;    2695  1     !       MESSAGE, AND TO VERIFY THE RESPONSE.
;    2696  1     !
;    2697  1     !       IMPLICIT INPUTS:
;    2698  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2699  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2700  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    2701  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    2702  1     !-
;    2703  1
;    2704  2         begin
;    2705  2         local
;    2706  2             MAX0_LBNS : WORD UNSIGNED,              ! UNIT'S MAXIMUM LO WORD LBN
;    2707  2             MAX1_LBNS : WORD UNSIGNED;              ! UNIT'S MAXIMUM HI WORD LBN
;    2708  2
;    2709  2         P_INDEX = GET_PKT (.CCTLR);                 ! GET AN MSCP PACKET
;    2710  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;       ! PACKET SIZE
;    2711  2         MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;       ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;    2712  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;       ! OPCODE FOR "ONLINE"
;    2713  2     !ZZZ MSCP_PKT [.P_INDEX, DDPAR] = BIT00;        ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;    2714  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = SEQ_CMD;    ! SEQUENTIAL COMMAND
;    2715  2
;    2716  2         if SEND (.P_INDEX) eql FAILURE              ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;    2717  2         then
;    2718  3             begin
;    2719  3             T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    2720  3
;    2721  3             if .APT_MODE                            !ZZZ
;    2722  3             then
;    2723  4                 begin
;    2724  4                 .MAIL_BOX_TESTNUM = 1;
;    2725  4                 .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    2726  3                 end;
;    2727  3
;    2728  3             CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;    2729  3             ERRDF (22, EGD_22, 0);                  !
;    2730  3             DUR [.L$LUN] = DU_ONLINE;               ! SETUP REASON TO DROP UNIT
;    2731  3             DODU (.L$LUN);                          ! DROP UNIT
;    2732  3             PUT_PKT (.P_INDEX);                     ! RETURN PACKET TO POOL
;    2733  3             end
;    2734  2         else
;    2735  3             begin                                   ! OTHERWISE (SEND WAS SUCCESSFUL)
;    2736  3
;    2737  3             do
;    2738  4                 begin
;    2739  4                 WAIT ();                            ! WAIT FOR RETPKT RESPONSE
;    2740  4                 RP_INDX = OUT_IODQ ();              ! GET INDEX OF RETPKT
;    2741  4                 RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
```

```
;    2742  4
;    2743  4                    if .RP_ADDR [MESTYP] neq MT_SEQ     ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2744  4                    then
;    2745  4                        PUT_RETPKT (.RP_INDX);
;    2746  4
;    2747  4                    end
;    2748  3                until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    2749  4                      ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    2750  3                       ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2751  3
;    2752  3                if .RP_ADDR [CONID] eql CID_DRIVER      ! IF RETPKT IS FROM "DRIVER"
;    2753  3                then
;    2754  4                    begin
;    2755  4                    PRINTF (DBM26);                     ! "ERROR IN UNIT_INIT"
;    2756  4                    DR_ERR ();                          ! DROP CONTROLLER
;    2757  4                    end
;    2758  3                else
;    2759  3
;    2760  4                    if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
;    2761  3                    then
;    2762  4                        begin
;    2763  4                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2764  4
;    2765  4                        if .APT_MODE                    !ZZZ
;    2766  4                        then
;    2767  5                            begin
;    2768  5                            .MAIL_BOX_TESTNUM = 1;
;    2769  5                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    2770  4                            end;
;    2771  4
;    2772  4                        CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2773  4                        ERRDF (23, EGD_23, EMS_21);     ! FATAL ERROR
;    2774  4                        DUR [.L$LUN] = DU_ONLINE;       ! SETUP REASON TO DROP UNIT
;    2775  4                        DODU (.L$LUN);                  ! DROP UNIT
;    2776  4                        end
;    2777  3                    else
;    2778  4                        begin                           ! RETPKT HAS GOOD ENDCODE
;    2779  4                        ST_CODE = .RP_ADDR [STSCOD];    ! GET STATUS CODE
;    2780  4                        SB_CODE = .RP_ADDR [SUBCOD];    ! GET SUB-CODE
;    2781  4
;    2782  4                        CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_0] = .RP_ADDR [NAME_0] + %o'100';                  ! UNIT NAME
ZZZ
;    2783  4                        CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .RP_ADDR [NAME_1_HI] * 16;                    !ZZZ
;    2784  4                        CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] +    !ZZZ
;    2785  4                                                                  .RP_ADDR [NAME_1_LO] + %o'100';                !ZZZ
;    2786  4                        CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_2] = .RP_ADDR [NAME_NUM] / 10 + %o'60';            !ZZZ
;    2787  4                        CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_3] = (.RP_ADDR [NAME_NUM] mod 10) + %o'60';        !ZZZ
;    2788  4
;    2789  4
;    2790  4
;    2791  4                        IF .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] EQL %O'104'         !IF NAME IS _D          !ZZZ
;    2792  4                        THEN                                                                                    !ZZZ
;    2793  4                        CST_ADDR [.CUOFF, D_TYPE] = FIXED                               !ITS FIXED.             !ZZZ
;    2794  4                        ELSE                                                                                    !ZZZ
```

```
;    2795  4                    CST_ADDR [.CUOFF, D_TYPE] = REMOVABLE;                          !OTHERWISE REMOVABLE    !ZZZ
;    2796  4
;    2797  4
;    2798  4
;    2799  4                    if .ST_CODE neq ST_SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
;    2800  4                    then
;    2801  5                        begin
;    2802  5                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2803  5
;    2804  5                        if .APT_MODE                !ZZZ
;    2805  5                        then
;    2806  6                            begin
;    2807  6                            .MAIL_BOX_TESTNUM = 1;
;    2808  6                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    2809  5                            end;
;    2810  5
;    2811  5                        CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2812  5                        ERRDF (15, EGD_15, EMS_30); ! ONLINE FAILED
;    2813  5                        DUR [.L$LUN] = DU_ONLINE;   ! SET UP REASON FOR DROPPING UNIT
;    2814  5                        DODU (.L$LUN);              ! DROP UNIT
;    2815  5                        end
;    2816  4                    else
;    2817  5                        begin                       ! SUCCESFUL OPERATION
;    2818  5
;    2819  5                        MAX0_LBNS = .RP_ADDR [SIZE0];                               ! LOAD LOWER WORD OF UNIT SIZE
;    2820  5                        MAX1_LBNS = .RP_ADDR [SIZE1];                               ! LOAD UPPER WORD OF UNIT SIZE
;    2821  5
;    2822  6                        if (.MAX0_LBNS eql 0)                                       ! THIS SUBTRACTS ONE FROM THE TOTAL
;    2823  5                        then                                                        ! BECAUSE EVERYTHING STARTS AT 0
;    2824  6                            begin                                                   ! THROUGH (MAXIMUM - 1)
;    2825  6                            MAX0_LBNS = %O'177777';
;    2826  6                            MAX1_LBNS = .MAX1_LBNS - 1;
;    2827  6                            end
;    2828  5                        else
;    2829  5                            MAX0_LBNS = .MAX0_LBNS - 1;
;    2830  5
;    2831  5                        if  (.CST_ADDR [.CUOFF + 2, D_BEG1] gtru .MAX1_LBNS) or             ! THIS SECTION CHECKS TO SEE
 IF LBNs LISTED
;    2832  5
;    2833  6                            ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and           ! IN SOFTWARE QUESTIONS WERE
 TO LARGE FOR
;    2834  6                            (.CST_ADDR [.CUOFF + 1, D_BEG0] gtru (.MAX0_LBNS - 1)))         ! DEVICE SPECIFIED
;    2835  6                                                                                        ! note 1 less then max. or diagnosti
c  will error
;    2836  5                        then                                                            ! operator error
;    2837  6                            begin
;    2838  6                            CST_ADDR [.CUOFF + 2, D_BEG1] = 0;
;    2839  6                            CST_ADDR [.CUOFF + 1, D_BEG0] = 0;                          ! change beginning lbn to 0
;    2840  5                            end;
;    2841  5
;    2842  5                         if
;    2843  5                            (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or
;    2844  5
;    2845  6                            ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
;    2846  6                            (.CST_ADDR [.CUOFF + 3, D_END0] gtru .MAX0_LBNS))
;    2847  5                         then
```

```
;    2848  6                          begin
;    2849  6                          CST_ADDR [.CUOFF + 4, D_END1] = .MAX1_LBNS;
;    2850  6                          CST_ADDR [.CUOFF + 3, D_END0] = .MAX0_LBNS;          ! and ending lbn to max_lbn
;    2851  5                          end;
;    2852  5
;    2853  5
;    2854  5
;    2855  7                          if (((.ENTRY_REASON eql RESTART) or                  ! if restart or
;    2856  6                              (.ENTRY_REASON eql START)) and                   ! if continue
;    2857  6
;    2858  6                              (.CRN_LOW leq 8) and                             ! and
;    2859  6                              (.CRN_HIGH eql 0))                               ! first initialization
;    2860  6
;    2861  5                          THEN                                                 ! intialize block numbers
;    2862  6                              begin
;    2863  6                              BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0];      ! LOAD sequential LBN table
;    2864  6                              BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1];      !
;    2865  6                              TRK_SGN [.L$LUN] = 1;                                       ! POSITIVE TRACKING DIRECTIO
N
;    2866  5                              end;
;    2867  5
;    2868  5
;    2869  5                                                                           ! THIS SECTION LOADS TYPE INTO  CST TABLE
;    2870  5      !ZZZ                        selectoneu .RP_ADDR [R_MODEL] of                ! MODEL BYTE TELLS WHAT TYPE OF UNIT
 IN
;    2871  5      !ZZZ                                                                        ! INDENTIFICATION BLOCK
;    2872  5      !ZZZ                            set
;    2873  5      !ZZZ
;    2874  5      !ZZZ                            [%O'6'] :   CST_ADDR [.CUOFF, D_TYPE] = RD_51;      ! RD 51
;    2875  5      !ZZZ                            [%O'7'] :   CST_ADDR [.CUOFF, D_TYPE] = RX_50;      ! RX 50
;    2876  5      !ZZZ                            [%O'10'] :  CST_ADDR [.CUOFF, D_TYPE] = RD_52;      ! RD 52
;    2877  5      !ZZZ
;    2878  5      !ZZZ                            [otherwise] : BEGIN
;    2879  5      !ZZZ                                          ERRDF (25 ,EGD_24 ,EMS_30);           ! ERROR UNKNOWN DEVICE
;    2880  5      !ZZZ                                          END;
;    2881  5      !ZZZ                            tes;
;    2882  5
;    2883  5
;    2884  5
;    2885  5                          if ((.RP_ADDR [U_FLGS] and UF_WPH) eql UF_WPH) and          ! STATUS CODE IS O.K.
;    2886  6                             (.CST_ADDR [.CUOFF, D_PROT] eql UNPROTECTED)
;    2887  5                          then
;    2888  6                              begin
;    2889  6                              T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2890  6
;    2891  6                              if .APT_MODE                        !ZZZ
;    2892  6                              then
;    2893  7                                  begin
;    2894  7                                  .MAIL_BOX_TESTNUM = 1;
;    2895  7                                  .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    2896  6                                  end;
;    2897  6
;    2898  6                              CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2899  6                              ERRDF (16, EGD_16, EMS_30);                             ! WRITE-PROTECT CONFLICT
;    2900  6                              DUR [.L$LUN] = DU_PROTECT;                              ! SET REASON TO DROP UNIT
```

```
;   2901  6                       DODU (.L$LUN);                                  ! DROP UNIT
;   2902  6                       end
;   2903  5                   else
;   2904  6                       begin                                           ! WRITE PROTECT SWITCH IS O.K.
;   2905  6                       CST_ADDR [.CUOFF, D_STAT] = ONLINE;             ! SET ONLINE FLAG
;   2906  6                       CST [.CCTLR, U_CNT] = .CST [.CCTLR, U_CNT] + 1; ! ADD UNIT TO CTRL TABLE
;   2907  5                       end;
;   2908  4                   end;
;   2909  3               end;                                                    ! IF RETPKT HAS CORRECT ENDCODE
;   2910  3
;   2911  3           PUT_RETPKT (.RP_INDX);
;   2912  2           end;                                                        ! IF SEND WAS SUCCESSFUL
;   2913  2
;   2914  1       end;                                                            ! ROUTINE UNIT-INIT


                                        .SBTTL  UNIT.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G                UNIT.INIT:
                                        JSR     R1,$SAVE5                       ;                       2689
000004  024646                          CMP     -(SP),-(SP)
000006  013746  000000G                MOV     CCTLR,-(SP)                     ;                       2709
000012  004737  000000G                JSR     PC,GET.PKT
000016  010037  000000G                MOV     R0,P.INDEX
000022  010016                         MOV     R0,(SP)                         ; P.INDEX,*             2710
000024  012746  000106                 MOV     #106,-(SP)
000030  004737  000000G                JSR     PC,BL$MUL
000034  012760  000044  000006G        MOV     #44,MSCP.PKT+6(R0)
000042  013760  000000G 000016G        MOV     CDISK,MSCP.PKT+16(R0)           ;                       2711
000050  112760  000011  000022G        MOVB    #11,MSCP.PKT+22(R0)             ;                       2712
000056  112760  000001  000004G        MOVB    #1,MSCP.PKT+4(R0)               ;                       2714
000064  013716  000000G                MOV     P.INDEX,(SP)                    ;                       2716
000070  004737  000000G                JSR     PC,SEND
000074  005700                         TST     R0
000076  001054                         BNE     2$
000100  013700  000000G                MOV     T.ADDR,R0                       ;                       2719
000104  105260  000051                 INCB    51(R0)
000110  032737  000001  001166'        BIT     #1,APT.MODE                     ;                       2721
000116  001415                         BEQ     1$
000120  012777  000001  001170'        MOV     #1,@MAIL.BOX.TESTNUM            ;                       2724
000126  013700  000000G                MOV     CUOFF,R0                        ;                       2725
000132  006300                         ASL     R0
000134  063700  000000G                ADD     CST.ADDR,R0
000140  111077  001172'                MOVB    (R0),@MAIL.BOX.SUBTST
000144  042777  177760  001172'        BIC     #177760,@MAIL.BOX.SUBTST
000152  013700  000000G        1$:     MOV     CUOFF,R0                        ;                       2728
000156  006300                         ASL     R0
000160  063700  000000G                ADD     CST.ADDR,R0
000164  052710  010000                 BIS     #10000,(R0)
000170  104455                         TRAP    55                              ;                       2729
000172  000026                         .WORD   26
000174  000000G                        .WORD   EGD.22
000176  000000                         .WORD   0
000200  013700  000000G                MOV     L$LUN,R0                        ;                       2730
```

```
000204  112760  000007  000000G                 MOVB    #7,DUR(R0)
000212  104451                                  TRAP    51                      ;                       2731
000214  013716  000000G                         MOV     P.INDEX,(SP)            ;                       2732
000220  004737  000000G                         JSR     PC,PUT.PKT
000224  000137  006666'                         JMP     26$                     ;                       2716
000230  004737  000000G           2$:           JSR     PC,WAIT                 ;                       2739
000234  004737  000000G                         JSR     PC,OUT.IODQ             ;                       2740
000240  010037  000000G                         MOV     R0,RP.INDX
000244  010016                                  MOV     R0,(SP)                 ; RP.INDX,*             2741
000246  012746  000054                         MOV     #54,-(SP)
000252  004737  000000G                         JSR     PC,BL$MUL
000256  062700  000000G                         ADD     #RETPKT,R0
000262  010037  000000G                         MOV     R0,RP.ADDR
000266  132760  000360  000002                 BITB    #360,2(R0)              ;                       2743
000274  001404                                  BEQ     3$
000276  013716  000000G                         MOV     RP.INDX,(SP)            ;                       2745
000302  004737  000000G                         JSR     PC,PUT.RETPKT
000306  005726                    3$:           TST     (SP)+                   ;                       2738
000310  013702  000000G                         MOV     RP.ADDR,R2              ;                       2748
000314  005000                                  CLR     R0
000316  126227  000003  000003                 CMPB    3(R2),#3
000324  001002                                  BNE     4$
000326  005200                                  INC     R0
000330  000407                                  BR      5$
000332  132762  000360  000002           4$:    BITB    #360,2(R2)              ;                       2749
000340  001333                                  BNE     2$
000342  105762  000014                         TSTB    14(R2)                  ;                       2750
000346  100330                                  BPL     2$
000350  006000                    5$:           ROR     R0                      ;                       2752
000352  103012                                  BCC     6$
000354  012716  000000G                         MOV     #DBM26,(SP)             ;                       2755
000360  012746  000001                         MOV     #1,-(SP)
000364  010600                                  MOV     SP,R0                   ; SP,*
000366  104417                                  TRAP    17
000370  004737  000000V                         JSR     PC,DR.ERR               ;                       2756
000374  005726                                  TST     (SP)+                   ;                       2754
000376  000457                                  BR      8$                      ;                       2752
000400  013705  000000G           6$:           MOV     CST.ADDR,R5             ;                       2772
000404  013766  000000G  000004                 MOV     CUOFF,4(SP)
000412  006366  000004                         ASL     4(SP)
000416  060566  000004                         ADD     R5,4(SP)
000422  126227  000014  000211                 CMPB    14(R2),#211             ;                       2760
000430  001444                                  BEQ     9$
000432  013700  000000G                         MOV     T.ADDR,R0               ;                       2763
000436  105260  000050                         INCB    50(R0)
000442  032737  000001  001166'                 BIT     #1,APT.MODE             ;                       2765
000450  001415                                  BEQ     7$
000452  012777  000001  001170'                 MOV     #1,@MAIL.BOX.TESTNUM    ;                       2768
000460  013700  000000G                         MOV     CUOFF,R0                ;                       2769
000464  006300                                  ASL     R0
000466  063700  000000G                         ADD     CST.ADDR,R0
000472  111077  001172'                         MOVB    (R0),@MAIL.BOX.SUBTST
000476  042777  177760  001172'                 BIC     #177760,@MAIL.BOX.SUBTST
```

```
000504  052776  010000  000004        7$:   BIS     #10000,@4(SP)                   ;                       2772
000512  104455                              TRAP    55                              ;                       2773
000514  000027                              .WORD   27
000516  000000G                             .WORD   EGD.23
000520  000000G                             .WORD   EMS.21
000522  013700  000000G                     MOV     L$LUN,R0                        ;                       2774
000526  112760  000007  000000G             MOVB    #7,DUR(R0)
000534  104451                              TRAP    51                              ;                       2775
000536  000137  006656'              8$:    JMP     25$                             ;                       2760
000542  116237  000016  000000G      9$:    MOVB    16(R2),ST.CODE                  ;                       2779
000550  042737  177740  000000G             BIC     #177740,ST.CODE
000556  016200  000016                      MOV     16(R2),R0                       ;                       2780
000562  006200                              ASR     R0
000564  006200                              ASR     R0
000566  006200                              ASR     R0
000570  006200                              ASR     R0
000572  006200                              ASR     R0
000574  042700  174000                      BIC     #174000,R0
000600  010037  000000G                     MOV     R0,SB.CODE
000604  013701  000000G                     MOV     CUOFF,R1                        ;                       2782
000610  006301                              ASL     R1
000612  060501                              ADD     R5,R1
000614  012703  000012                      MOV     #12,R3
000620  060103                              ADD     R1,R3
000622  116200  000036                      MOVB    36(R2),R0
000626  006200                              ASR     R0
000630  042700  177740                      BIC     #177740,R0
000634  062700  000100                      ADD     #100,R0
000640  110013                              MOVB    R0,(R3)
000642  116200  000036                      MOVB    36(R2),R0                       ;                       2783
000646  042700  177776                      BIC     #177776,R0
000652  006300                              ASL     R0
000654  006300                              ASL     R0
000656  006300                              ASL     R0
000660  006300                              ASL     R0
000662  110063  000001                      MOVB    R0,1(R3)
000666  005000                              CLR     R0                              ;                       2784
000670  156300  000001                      BISB    1(R3),R0
000674  016201  000034                      MOV     34(R2),R1
000700  006201                              ASR     R1
000702  006201                              ASR     R1
000704  006201                              ASR     R1
000706  006201                              ASR     R1
000710  000301                              SWAB    R1
000712  042701  177760                      BIC     #177760,R1
000716  060100                              ADD     R1,R0
000720  010001                              MOV     R0,R1                           ;                       2785
000722  062701  000100                      ADD     #100,R1
000726  110163  000001                      MOVB    R1,1(R3)
000732  013701  000000G                     MOV     CUOFF,R1                        ;                       2786
000736  006301                              ASL     R1
000740  060501                              ADD     R5,R1
000742  116216  000034                      MOVB    34(R2),(SP)
```

```
000746  042716  177700                          BIC     #177700,(SP)
000752  012746  000012                          MOV     #12,-(SP)
000756  004737  000000G                         JSR     PC,BL$DIV
000762  010004                                  MOV     R0,R4
000764  062704  000060                          ADD     #60,R4
000770  110461  000014                          MOVB    R4,14(R1)
000774  116216  000034                          MOVB    34(R2),(SP)                             ;                       2787
001000  042716  177700                          BIC     #177700,(SP)
001004  012746  000012                          MOV     #12,-(SP)
001010  004737  000000G                         JSR     PC,BL$MOD
001014  010004                                  MOV     R0,R4
001016  062704  000060                          ADD     #60,R4
001022  110461  000015                          MOVB    R4,15(R1)
001026  126327  000001  000104                  CMPB    1(R3),#104                              ;                       2791
001034  001004                                  BNE     10$
001036  152776  000020  000010                  BISB    #20,@10(SP)                             ;                       2793
001044  000403                                  BR      11$                                     ;                       2791
001046  142776  000020  000010          10$:    BICB    #20,@10(SP)                             ;                       2795
001054  005737  000000G                 11$:    TST     ST.CODE                                 ;                       2799
001060  001440                                  BEQ     13$
001062  013700  000000G                         MOV     T.ADDR,R0                               ;                       2802
001066  105260  000050                          INCB    50(R0)
001072  032737  000001  001166'                BIT     #1,APT.MODE                             ;                       2804
001100  001411                                  BEQ     12$
001102  012777  000001  001170'                MOV     #1,@MAIL.BOX.TESTNUM                    ;                       2807
001110  117677  000010  001172'                MOVB    @10(SP),@MAIL.BOX.SUBTST                ;                       2808
001116  042777  177760  001172'                BIC     #177760,@MAIL.BOX.SUBTST
001124  052776  010000  000010          12$:    BIS     #10000,@10(SP)                          ;                       2811
001132  104455                                  TRAP    55                                      ;                       2812
001134  000017                                  .WORD   17
001136  000000G                                 .WORD   EGD.15
001140  000000G                                 .WORD   EMS.30
001142  013700  000000G                         MOV     L$LUN,R0                                ;                       2813
001146  112760  000007  000000G                 MOVB    #7,DUR(R0)
001154  104451                                  TRAP    51                                      ;                       2814
001156  000137  006654'                         JMP     24$                                     ;                       2799
001162  016203  000044                  13$:    MOV     44(R2),R3                               ; *,MAX0.LBNS           2819
001166  016204  000046                          MOV     46(R2),R4                               ; *,MAX1.LBNS           2820
001172  005703                                  TST     R3                                      ; MAX0.LBNS             2822
001174  001004                                  BNE     14$
001176  012703  177777                          MOV     #-1,R3                                  ; *,MAX0.LBNS           2825
001202  005304                                  DEC     R4                                      ; MAX1.LBNS             2826
001204  000401                                  BR      15$                                     ;                       2822
001206  005303                          14$:    DEC     R3                                      ; MAX0.LBNS             2829
001210  013701  000000G                 15$:    MOV     CUOFF,R1                                ;                       2831
001214  006301                                  ASL     R1
001216  060501                                  ADD     R5,R1
001220  012766  000004  000012                  MOV     #4,12(SP)
001226  060166  000012                          ADD     R1,12(SP)
001232  027604  000012                          CMP     @12(SP),R4                              ; *,MAX1.LBNS
001236  101012                                  BHI     16$
001240  001021                                  BNE     17$                                     ;                       2833
001242  013701  000000G                         MOV     CUOFF,R1                                ;                       2834
```

```
001246  006301                          ASL     R1
001250  060501                          ADD     R5,R1
001252  010300                          MOV     R3,R0                   ; MAX0.LBNS,*
001254  005300                          DEC     R0
001256  026100  000002                  CMP     2(R1),R0
001262  101410                          BLOS    17$
001264  005076  000012          16$:    CLR     @12(SP)                 ;                       2838
001270  013701  000000G                 MOV     CUOFF,R1                ;                       2839
001274  006301                          ASL     R1
001276  060501                          ADD     R5,R1
001300  005061  000002                  CLR     2(R1)
001304  013701  000000G          17$:    MOV     CUOFF,R1                ;                       2843
001310  006301                          ASL     R1
001312  060501                          ADD     R5,R1
001314  026104  000010                  CMP     10(R1),R4               ; *,MAX1.LBNS
001320  101010                          BHI     18$
001322  001017                          BNE     19$                     ;                       2845
001324  013700  000000G                 MOV     CUOFF,R0                ;                       2846
001330  006300                          ASL     R0
001332  060500                          ADD     R5,R0
001334  026003  000006                  CMP     6(R0),R3                ; *,MAX0.LBNS
001340  101410                          BLOS    19$
001342  010461  000010          18$:    MOV     R4,10(R1)               ; MAX1.LBNS,*           2849
001346  013701  000000G                 MOV     CUOFF,R1                ;                       2850
001352  006301                          ASL     R1
001354  060501                          ADD     R5,R1
001356  010361  000006                  MOV     R3,6(R1)                ; MAX0.LBNS,*
001362  123727  000000G 000002  19$:    CMPB    ENTRY.REASON,#2         ;                       2855
001370  001404                          BEQ     20$
001372  123727  000000G 000001          CMPB    ENTRY.REASON,#1         ;                       2856
001400  001031                          BNE     21$
001402  023727  000000G 000010  20$:    CMP     CRN.LOW,#10             ;                       2858
001410  003025                          BGT     21$
001412  005737  000000G                 TST     CRN.HIGH                ;                       2859
001416  001022                          BNE     21$
001420  013703  000000G                 MOV     L$LUN,R3                ;                       2863
001424  010304                          MOV     R3,R4
001426  006304                          ASL     R4
001430  006304                          ASL     R4
001432  013701  000000G                 MOV     CUOFF,R1
001436  006301                          ASL     R1
001440  060501                          ADD     R5,R1
001442  016164  000002  000000G         MOV     2(R1),BST(R4)
001450  017664  000012  000002G         MOV     @12(SP),BST+2(R4)       ;                       2864
001456  112763  000001  000000G         MOVB    #1,TRK.SGN(R3)          ;                       2865
001464  032762  020000  000022  21$:    BIT     #20000,22(R2)           ;                       2885
001472  001442                          BEQ     23$
001474  005776  000010                  TST     @10(SP)                 ;                       2886
001500  100037                          BPL     23$
001502  013700  000000G                 MOV     T.ADDR,R0               ;                       2889
001506  105260  000050                  INCB    50(R0)
001512  032737  000001  001166'         BIT     #1,APT.MODE             ;                       2891
001520  001411                          BEQ     22$
```

```
001522  012777  000001  001170'              MOV     #1,@MAIL.BOX.TESTNUM            ;                2894
001530  117677  000010  001172'              MOVB    @10(SP),@MAIL.BOX.SUBTST        ;                2895
001536  042777  177760  001172'              BIC     #177760,@MAIL.BOX.SUBTST
001544  052776  010000  000010      22$:     BIS     #10000,@10(SP)                  ;                2898
001552  104455                               TRAP    55                              ;                2899
001554  000020                               .WORD   20
001556  000000G                              .WORD   EGD.16
001560  000000G                              .WORD   EMS.30
001562  013700  000000G                      MOV     L$LUN,R0                        ;                2900
001566  112760  000011  000000G              MOVB    #11,DUR(R0)
001574  104451                               TRAP    51                              ;                2901
001576  000414                               BR      24$                             ;                2885
001600  052776  020000  000010      23$:     BIS     #20000,@10(SP)                  ;                2905
001606  013716  000000G                      MOV     CCTLR,(SP)                      ;                2906
001612  012746  000126                       MOV     #126,-(SP)
001616  004737  000000G                      JSR     PC,BL$MUL
001622  105260  000005G                      INCB    CST+5(R0)
001626  005726                               TST     (SP)+                           ;                2904
001630  022626                       24$:    CMP     (SP)+,(SP)+                     ;                2778
001632  013716  000000G              25$:    MOV     RP.INDX,(SP)                    ;                2911
001636  004737  000000G                      JSR     PC,PUT.RETPKT
001642  062706  000010              26$:    ADD     #10,SP                          ;                2689
001646  000207                               RTS     PC

; Routine Size:  468 words,     Routine Base:  $CODE$ + 5024
; Maximum stack depth per invocation:  14 words
```

```
;    2915  1     GLOBAL routine DR_ERR : novalue =
;    2916  1
;    2917  1     !*
;    2918  1     !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;    2919  1     !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;    2920  1     !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;    2921  1     !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;    2922  1     !       AS AN INITIALIZATION ERROR.
;    2923  1     !
;    2924  1     !       IMPLICIT INPUTS:
;    2925  1     !               RP_ADDR - ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;    2926  1     !                       (I.E., CONNECTION ID = CID_DRIVER)
;    2927  1     !-
;    2928  1
;    2929  2         begin
;    2930  2
;    2931  2         local
;    2932  2             REASON : word initial (DU_TIME);        ! ASSUME COMMAND TIMEOUT
;    2933  2
;    2934  2         if .RP_ADDR [MESTYP] eql MT_FATAL           ! IF FATAL DEVICE ERROR
;    2935  2         then
;    2936  2
;    2937  2         DROP_CTLR (.CCTLR, .REASON);                ! DROP ALL UNITS ON CONTROLLER
;    2938  1         end;


                                    .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                DR.ERR::MOV     R1,-(SP)                        ;                       2915
000002  012701  000012                MOV     #12,R1                          ; #,REASON              2929
000006  013700  000000G               MOV     RP.ADDR,R0                      ;                       2934
000012  116000  000002                MOVB    2(R0),R0
000016  042700  177417                BIC     #177417,R0
000022  020027  000060                CMP     R0,#60
000026  001006                       BNE     1$
000030  013746  000000G               MOV     CCTLR,-(SP)                     ;                       2937
000034  010146                       MOV     R1,-(SP)                        ; REASON,#
000036  004737  000000G               JSR     PC,DROP.CTLR
000042  022626                       CMP     (SP)+,(SP)+
000044  012601                1$:    MOV     (SP)+,R1                        ;                       2915
000046  000207                       RTS     PC

; Routine Size:  20 words,      Routine Base:  $CODE$ + 6674
; Maximum stack depth per invocation:  4 words
```

```
;    2939  1      routine ACCESS : novalue =
;    2940  1
;    2941  1      !+
;    2942  1      !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
;    2943  1      !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;    2944  1      !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
;    2945  1      !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;    2946  1      !
;    2947  1      !       IMPLICIT INPUTS:
;    2948  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    2949  1      !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2950  1      !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    2951  1      !-
;    2952  1
;    2953  2          begin
;    2954  2
;    2955  2      local
;    2956  2              RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
;    2957  2              LBN : word,
;    2958  2              PASS : word initial (1);                ! LOOP PASS COUNT
;    2959  2
;    2960  2          ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB-CODE
;    2961  2          LBN = (((.MAX_LBN [.L$LUN] + 1) ^ -1) and %o'77777') - 1;
;    2962  2                                                      ! START WITH LAST LBN ON TOP SURFACE: [(X+1)/2] -1
;    2963  2
;    2964  2          do
;    2965  3              begin                                   ! LOOP STARTS HERE
;    2966  3              P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
;    2967  3              MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
;    2968  3              MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
;    2969  3              MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
;    2970  3              MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
;    2971  3              MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD;    ! NON-SEQUENTIAL COMMAND
;    2972  3
;    2973  3              if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;    2974  3              then
;    2975  4                  begin
;    2976  4                  PUT_PKT (.P_INDEX);                 ! RETURN PACKET TO POOL
;    2977  4                  PASS = 2;                           ! NO MORE TRIES
;    2978  4                  end
;    2979  3              else
;    2980  4                  begin                               ! IF SEND WAS SUCCESSFUL
;    2981  4
;    2982  4                  do
;    2983  5                      begin
;    2984  5                      WAIT ();                        ! WAIT FOR RESPONSE
;    2985  5                      RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
;    2986  5                      RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;    2987  5
;    2988  5                      if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2989  5                      then
;    2990  5                          PUT_RETPKT (.RP_INDX);
;    2991  5
```

```
;   2992  5                 end
;   2993  4             until (.RP_ADDR [CONID] eql CID_DRIVER) or
;   2994  5                   ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;   2995  4                    ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;   2996  4
;   2997  4             if .RP_ADDR [CONID] eql CID_DRIVER  ! IF RETPKT CAME FROM "DRIVER"
;   2998  4             then
;   2999  4                 PASS = 2                        ! NO MORE TRIES
;   3000  4             else
;   3001  4
;   3002  5                 if .RP_ADDR [ENDCOD] neq (OP_ACC or OP_END)
;   3003  4                 then
;   3004  5                     begin
;   3005  5                     PRINTF (DBM29);             ! "RETPKT HAS BAD ENDCODE"
;   3006  5     !ZZZ            EMSCMD ();
;   3007  5                     end
;   3008  4                 else
;   3009  5                     begin                       ! RETPKT HAS CORRECT ENDCODE
;   3010  5                     ST_CODE = .RP_ADDR [STSCOD];! GET STATUS CODE FROM PACKET
;   3011  5                     SB_CODE = .RP_ADDR [SUBCOD];! GET SUB-CODE FROM PACKET
;   3012  5
;   3013  5                     if .ST_CODE eql ST_SUC      ! IF STATUS CODE INDICATES SUCCESS
;   3014  5                     then
;   3015  6                         begin
;   3016  6                         RESULT = SUCCESS;
;   3017  6                         PASS = 2;               ! NO NEED TO TRY AGAIN
;   3018  5                         end;
;   3019  5
;   3020  4                     end;                        ! IF RETPKT HAS CORRECT ENDCODE
;   3021  4
;   3022  4             PUT_RETPKT (.RP_INDX);
;   3023  3             end;                                ! IF SEND WAS SUCCESSFUL
;   3024  3
;   3025  3         LBN = .LBN + 1;                         ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
;   3026  3         PASS = .PASS + 1;                       ! SECOND PASS
;   3027  3         end                                     ! END OF PASS LOOP
;   3028  2     until .PASS gequ 3;
;   3029  2
;   3030  2     if .RESULT eql FAILURE
;   3031  2     then
;   3032  3         begin
;   3033  3         T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;   3034  3         CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;   3035  3         ERRDF (17, EGD_17, EMS_30);                     ! ACCESS FAILED
;   3036  3         DUR [.L$LUN] = DU_ACCESS;               ! SET REASON TO DROP UNIT
;   3037  3         DODU (.L$LUN);                          ! DROP UNIT
;   3038  2         end;                                    ! IF ACCESS FAILED
;   3039  2
;   3040  1     end;                                        ! ROUTINE ACCESS


                                    .SBTTL  ACCESS INITIALIZATION TEST ROUTINES
000000  004137  000000G         ACCESS: JSR     R1,$SAVE4                       ;                                2939
```

```
000004  005003                          CLR     R3                              ; RESULT                        2953
000006  012702  000001                  MOV     #1,R2                           ; *,PASS
000012  005037  000000G                 CLR     SB.CODE                         ;                               2960
000016  005037  000000G                 CLR     ST.CODE
000022  013700  000000G                 MOV     L$LUN,R0                        ;                               2961
000026  006300                          ASL     R0
000030  016000  000054'                  MOV     MAX.LBN(R0),R0
000034  060200                          ADD     R2,R0
000036  006200                          ASR     R0
000040  010004                          MOV     R0,R4                           ; *,LBN
000042  042704  100000                  BIC     #100000,R4                      ; *,LBN
000046  005304                          DEC     R4                              ; LBN
000050  013746  000000G         1$:     MOV     CCTLR,-(SP)                     ;                               2966
000054  004737  000000G                 JSR     PC,GET.PKT
000060  010037  000000G                 MOV     R0,P.INDEX
000064  010016                          MOV     R0,(SP)                         ; P.INDEX,*                     2967
000066  012746  000106                  MOV     #106,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  013760  000000G  000016G        MOV     CDISK,MSCP.PKT+16(R0)
000104  112760  000020   000022G        MOVB    #20,MSCP.PKT+22(R0)             ;                               2968
000112  012760  001000   000026G        MOV     #1000,MSCP.PKT+26(R0)           ;                               2969
000120  010460  000046G                 MOV     R4,MSCP.PKT+46(R0)              ; LBN,*                         2970
000124  112760  000002   000004G        MOVB    #2,MSCP.PKT+4(R0)               ;                               2971
000132  013716  000000G                 MOV     P.INDEX,(SP)                    ;                               2973
000136  004737  000000G                 JSR     PC,SEND
000142  005700                          TST     R0
000144  001007                          BNE     2$
000146  013716  000000G                 MOV     P.INDEX,(SP)                    ;                               2976
000152  004737  000000G                 JSR     PC,PUT.PKT
000156  012702  000002                  MOV     #2,R2                           ; *,PASS                        2977
000162  000522                          BR      9$                              ;                               2973
000164  004737  000000G         2$:     JSR     PC,WAIT                         ;                               2984
000170  004737  000000G                 JSR     PC,OUT.IODQ                     ;                               2985
000174  010037  000000G                 MOV     R0,RP.INDX
000200  010016                          MOV     R0,(SP)                         ; RP.INDX,*                     2986
000202  012746  000054                  MOV     #54,-(SP)
000206  004737  000000G                 JSR     PC,BL$MUL
000212  062700  000000G                 ADD     #RETPKT,R0
000216  010037  000000G                 MOV     R0,RP.ADDR
000222  132760  000360  000002         BITB    #360,2(R0)                      ;                               2988
000230  001404                          BEQ     3$
000232  013716  000000G                 MOV     RP.INDX,(SP)                    ;                               2990
000236  004737  000000G                 JSR     PC,PUT.RETPKT
000242  005726                  3$:     TST     (SP)+                           ;                               2983
000244  013701  000000G                 MOV     RP.ADDR,R1                      ;                               2993
000250  005000                          CLR     R0
000252  126127  000003  000003         CMPB    3(R1),#3
000260  001002                          BNE     4$
000262  005200                          INC     R0
000264  000407                          BR      5$
000266  132761  000360  000002         BITB    #360,2(R1)                      ;                               2994
000274  001333                          BNE     2$
000276  105761  000014                  TSTB    14(R1)                          ;                               2995
```

```
000302  100330                          BPL     2$
000304  006000                  5$:     ROR     R0                              ;                       2997
000306  103442                          BLO     7$                              ;                       2999
000310  126127  000014  000220          CMPB    14(R1),#220                     ;                       3002
000316  001410                          BEQ     6$
000320  012716  000000G                 MOV     #DBM29,(SP)                     ;                       3005
000324  012746  000001                  MOV     #1,-(SP)
000330  010600                          MOV     SP,R0                           ; SP,*
000332  104417                          TRAP    17
000334  005726                          TST     (SP)+                           ;                       3004
000336  000430                          BR      8$                              ;                       3002
000340  116137  000016  000000G  6$:     MOVB    16(R1),ST.CODE                  ;                       3010
000346  042737  177740  000000G          BIC     #177740,ST.CODE
000354  016100  000016                  MOV     16(R1),R0                       ;                       3011
000360  006200                          ASR     R0
000362  006200                          ASR     R0
000364  006200                          ASR     R0
000366  006200                          ASR     R0
000370  006200                          ASR     R0
000372  042700  174000                  BIC     #174000,R0
000376  010037  000000G                 MOV     R0,SB.CODE
000402  005737  000000G                 TST     ST.CODE                         ;                       3013
000406  001004                          BNE     8$
000410  012703  000001                  MOV     #1,R3                           ; *,RESULT              3016
000414  012702  000002          7$:     MOV     #2,R2                           ; *,PASS                3017
000420  013716  000000G          8$:     MOV     RP.INDX,(SP)                    ;                       3022
000424  004737  000000G                  JSR     PC,PUT.RETPKT
000430  005204                  9$:     INC     R4                              ; LBN                   3025
000432  005202                          INC     R2                              ; PASS                  3026
000434  022626                          CMP     (SP)+,(SP)+                     ;                       2965
000436  020227  000003                  CMP     R2,#3                           ; PASS,*                3028
000442  103602                          BLO     1$
000444  005703                          TST     R3                              ; RESULT                3030
000446  001025                          BNE     10$
000450  013700  000000G                 MOV     T.ADDR,R0                       ;                       3033
000454  105260  000050                  INCB    50(R0)
000460  013700  000000G                 MOV     CUOFF,R0                        ;                       3034
000464  006300                          ASL     R0
000466  063700  000000G                 ADD     CST.ADDR,R0
000472  052710  010000                  BIS     #10000,(R0)
000476  104455                          TRAP    55                              ;                       3035
000500  000021                          .WORD   21
000502  000000G                         .WORD   EGD.17
000504  000000G                         .WORD   EMS.30
000506  013700  000000G                 MOV     L$LUN,R0                        ;                       3036
000512  112760  000010  000000G         MOVB    #10,DUR(R0)
000520  104451                          TRAP    51                              ;                       3037
000522  000207                  10$:    RTS     PC                              ;                       2939

; Routine Size:  170 words,    Routine Base:  $CODE$ + 6744
; Maximum stack depth per invocation:  10 words
```

```
;    3041  1      %sbttl 'MULTI-DRIVE TEST ROUTINES'
;    3042  1
;    3043  1
;    3044  1      GLOBAL routine MULTI_DRIVE : novalue =
;    3045  1
;    3046  1
;    3047  1      !+
;    3048  1      !       THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;    3049  1      !       MULTI-DRIVE TEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;    3050  1      !       USER AN INDICATION OF HOW ONE OR SEVERAL RDRX DRIVES WOULD PERFORM IN
;    3051  1      !       AN OPERATING SYSTEM ENVIRONMENT.
;    3052  1      !
;    3053  1      !       THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
;    3054  1      !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE TEST DATA, THIS ROUTINE
;    3055  1      !       ENTERS A LOOP WHICH ISSUES QIOs TO ALL ACTIVE CONTROLLERS AND PROCESSES
;    3056  1      !       ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;    3057  1      !       DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;    3058  1      !       LOOP OCCURS WHEN QIOs ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;    3059  1      !       QIOS HAVE COMPLETED.
;    3060  1      !-
;    3061  1
;    3062  1
;    3063  2          begin
;    3064  2
;    3065  2          local
;    3066  2              CUR_PRIORITY : word;
;    3067  2
;    3068  2          label
;    3069  2              SEND_COMMANDS;
;    3070  2
;    3071  2          MD_INIT ();                                          ! INIT MULTI-DRIVE TEST DATA
;    3072  2          INIT_OCCURED = TRUE;                                 !
;    3073  2
;    3074  2
;    3075  3          do  begin                                            ! START OF EXECUTIVE LOOP
;    3076  3
;    3077  3              incr CTLR from 0 to (MAX_CTLR - 1) do            ! FOR EACH CONTROLLER
;    3078  4                  begin
;    3079  4                  SET_CPAR (.CTLR);                            ! SET UP CURRENT CONTROLLER PARAMETERS
;    3080  4                  GETPRI (CUR_PRIORITY);                       !
;    3081  4                  SETPRI (PRI04);                              ! NO INTERRUPTS WHEN EXAMINING SA
;    3082  4                  ICTLR = .CCTLR;                              ! FAKE INTERRUPTING CONTROLLER'S NUMBER
;    3083  4                  ICST_ADDR = .CST_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S CST ADDR
;    3084  4                  IDCT_ADDR = .DCT_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S DCT ADDR
;    3085  4                  IRDRX_ADDR = .ICST_ADDR [IP_ADDR];           ! FAKE INTERRUPTING CONTROLLER'S ADDRESS
;    3086  4                  IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];  ! CONTENTS OF THE SA REGISTER
;    3087  4
;    3088  5                  if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)     ! IF SA SHOWS AN ERROR
;    3089  4                  then
;    3090  5                      begin
;    3091  5                      FATAL_ERROR ();                          ! DECLARE FATAL ERROR
;    3092  5                      SETPRI (.CUR_PRIORITY);                  ! LOWER PRIORITY
;    3093  5                      exitloop;                                ! QUIT
```

```
;    3094  5                    end
;    3095  5
;    3096  4                else
;    3097  4                    SETPRI (.CUR_PRIORITY);                          ! IF NO ERROR, CONTINUE
;    3098  4
;    3099  4                if QIO_OK ()                                         ! IF O.K. TO ISSUE QIO(S) TO CONTROLLER
;    3100  4                then
;    3101  4                    SEND_COMMANDS:
;    3102  5                        begin
;    3103  5                        QIO_GEN ();                                  ! GENERATE 1 OR 2 QIOs
;    3104  5
;    3105  5                        if (.MX1 geq 0) and                          ! IF SUCCESS ON FIRST QIO
;    3106  6                           (not .EOP_FLAG)
;    3107  5                        then
;    3108  5
;    3109  5                            if SEND (.MX1) eql SUCCESS               ! ATTEMPT TO SEND IT. IF SUCCESS
;    3110  5                            then
;    3111  6                            BEGIN                                    !ZZZ
;    3112  6                                QIO [.CTLR] = .QIO [.CTLR] + 1;      ! INCR OUTSTANDING QIO COUNT ZZZ
;    3113  6                                RW_BALANCE = .RW_BALANCE + 1;        !INCR RD/WR RATIO COUNT
;    3114  6                            END                                      !ZZZ
;    3115  6
;    3116  5                            else
;    3117  6                                begin
;    3118  6                                PUT_PKT (.MX1);                      ! RETURN PACKET TO POOL
;    3119  6                                leave SEND_COMMANDS;
;    3120  5                                end;
;    3121  5
;    3122  5
;    3123  5                        if (.MX2 geq 0) and                          ! IF SUCCESS ON SECOND QIO
;    3124  6                           (not .EOP_FLAG)
;    3125  5                        then
;    3126  6                            begin
;    3127  6
;    3128  6                            do
;    3129  6                                BREAK                                ! WAIT TILL 1 MORE SLOT AVAILABLE IN CRING
;    3130  6                            until (.DCT_ADDR [CRING_CNT] lssu CRING_LEN);    !
;    3131  6
;    3132  6                            if SEND (.MX2) eql SUCCESS               ! ATTEMPT TO SEND IT.
;    3133  6                            then
;    3134  7                            BEGIN                                    !ZZZ
;    3135  7                                QIO [.CTLR] = .QIO [.CTLR] + 1;      ! IF SUCCESS, INCR OUTSTANDING QIO COUNT
;    3136  7                                RW_BALANCE = 0;                      !INDICATE RD/WT PAIR WAS LAST ISSUED  ZZZ
;    3137  7                            END                                      !ZZZ
;    3138  7
;    3139  6                            else
;    3140  7                                begin
;    3141  7                                PRINTF (DBM121, .CRN_HIGH, .CRN_LOW);
;    3142  7                                COMPARE_DATA = FALSE;                ! NO SENSE IN COMPARING WRITE DATA
;    3143  7                                PUT_PKT (.MX2);                      ! RETURN PACKET TO POOL
;    3144  6                                end;
;    3145  6
;    3146  5                            end;
```

```
;    3147  5
;    3148  4                        end;                                   ! O.K. TO ISSUE QIO(S)
;    3149  3                    end;                                       ! CONTROLLER LOOP
;    3150  3
;    3151  3
;    3152  3                    BREAK;                                     ! LET SUPERVISOR CATCH USER REQUESTS
;    3153  3                    PROC_RETPKT ();                            ! PROCESS ANY RETURN PACKETS
;    3154  3
;    3155  3                    end                                        ! EXECUTIVE PROCESSING LOOP
;    3156  3                until ((not QIO_OUT ()) or
;    3157  4                      ((.DCT_ADDR [CRING_CNT] eql 0) and
;    3158  2                       (.EOP_FLAG)));
;    3159  2
;    3160  2
;    3161  2                DCT_ADDR [IG_INT] = TRUE;                      ! NO FURTHER INTERRUPTS ON THIS CONTROLLER
;    3162  2
;    3163  2
;    3164  1                end;                                           ! EXERCISER


                                        .SBTTL  MULTI.DRIVE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          MULTI.DRIVE::
                                        JSR     R1,$SAVE3                       ;                                       3044
000004  005746                         TST     -(SP)
000006  004737  000000V                 JSR     PC,MD.INIT                      ;                                       3071
000012  112737  000001  000000G        MOVB    #1,INIT.OCCURED                 ;                                       3072
000020  005001                  1$:     CLR     R1                              ; CTLR                                  3077
000022  010146                  2$:     MOV     R1,-(SP)                        ; CTLR,*                                3079
000024  004737  000000G                 JSR     PC,SET.CPAR
000030  104440                         TRAP    40                              ;                                       3080
000032  010003                         MOV     R0,R3                           ; *,CUR.PRIORITY
000034  012700  000200                 MOV     #200,R0                         ;                                       3081
000040  104441                         TRAP    41
000042  013737  000000G 000104'        MOV     CCTLR,ICTLR                     ;                                       3082
000050  013737  000000G 000076'        MOV     CST.ADDR,ICST.ADDR              ;                                       3083
000056  013737  000000G 000100'        MOV     DCT.ADDR,IDCT.ADDR              ;                                       3084
000064  017737  000076' 000000G        MOV     @ICST.ADDR,IRDRX.ADDR           ;                                       3085
000072  013700  000100'                MOV     IDCT.ADDR,R0                    ;                                       3086
000076  013702  000000G                MOV     IRDRX.ADDR,R2
000102  016266  000002  000002        MOV     2(R2),2(SP)                     ; *,RC.REG
000110  016660  000002  000002        MOV     2(SP),2(R0)                     ; RC.REG,*
000116  016600  000002                MOV     2(SP),R0                        ;                                       3088
000122  042700  077777                BIC     #77777,R0
000126  020027  100000                CMP     R0,#-100000
000132  001006                       BNE     3$
000134  004737  000000V               JSR     PC,FATAL.ERROR                  ;                                       3091
000140  010300                       MOV     R3,R0                           ; CUR.PRIORITY,*                        3092
000142  104441                         TRAP    41
000144  005726                         TST     (SP)+                           ;                                       3090
000146  000515                         BR      9$
000150  010300                  3$:    MOV     R3,R0                           ; CUR.PRIORITY,*                        3097
000152  104441                         TRAP    41
000154  004737  000000V               JSR     PC,QIO.OK                       ;                                       3099
```

```
000160  006000                          ROR     R0
000162  103103                          BCC     8$
000164  004737  000000V                 JSR     PC,QIO.GEN              ;                       3103
000170  013700  000112'                 MOV     MX1,R0                  ;                       3105
000174  002424                          BLT     5$
000176  132737  000001  000000G         BITB    #1,EOP.FLAG             ;                       3106
000204  001020                          BNE     5$
000206  010016                          MOV     R0,(SP)                 ;                       3109
000210  004737  000000G                 JSR     PC,SEND
000214  020027  000001                  CMP     R0,#1
000220  001005                          BNE     4$
000222  105261  000000G                 INCB    QIO(R1)                 ; #(CTLR)               3112
000226  005237  000106'                 INC     RW.BALANCE              ;                       3113
000232  000405                          BR      5$                      ;                       3109
000234  013716  000112'         4$:     MOV     MX1,(SP)                ;                       3118
000240  004737  000000G                 JSR     PC,PUT.PKT
000244  000452                          BR      8$                      ;                       3117
000246  005737  000114'         5$:     TST     MX2                     ;                       3123
000252  002447                          BLT     8$
000254  132737  000001  000000G         BITB    #1,EOP.FLAG             ;                       3124
000262  001043                          BNE     8$
000264  104422                  6$:     TRAP    22                      ;                       3128
000266  127727  000000G 000004         CMPB    @DCT.ADDR,#4            ;                       3130
000274  103373                          BHIS    6$
000276  013716  000114'                 MOV     MX2,(SP)                ;                       3132
000302  004737  000000G                 JSR     PC,SEND
000306  020027  000001                  CMP     R0,#1
000312  001005                          BNE     7$
000314  105261  000000G                 INCB    QIO(R1)                 ; #(CTLR)               3135
000320  005037  000106'                 CLR     RW.BALANCE              ;                       3136
000324  000422                          BR      8$                      ;                       3132
000326  013716  000000G         7$:     MOV     CRN.LOW,(SP)            ;                       3141
000332  013746  000000G                 MOV     CRN.HIGH,-(SP)
000336  012746  000000G                 MOV     #DBM121,-(SP)
000342  012746  000003                  MOV     #3,-(SP)
000346  010600                          MOV     SP,R0                   ; SP,#
000350  104417                          TRAP    17
000352  105037  001174'                 CLRB    COMPARE.DATA            ;                       3142
000356  013716  000114'                 MOV     MX2,(SP)                ;                       3143
000362  004737  000000G                 JSR     PC,PUT.PKT
000366  062706  000006                  ADD     #6,SP                   ;                       3140
000372  005726                  8$:     TST     (SP)+                   ;                       3078
000374  005201                          INC     R1                      ; CTLR                  3077
000376  000243                          .WORD CLV!CLC

000400  003610                          BLE     2$
000402  104422                  9$:     TRAP    22                      ;                       3149
000404  004737  000000V                 JSR     PC,PROC.RETPKT          ;                       3153
000410  004737  000000V                 JSR     PC,QIO.OUT              ;                       3156
000414  006000                          ROR     R0
000416  103011                          BCC     12$
000420  105777  000000G                 TSTB    @DCT.ADDR               ;                       3157
000424  001402                          BEQ     11$
```

```
000426  000137  007510'                    10$:    JMP     1$
000432  132737  000001  000000G            11$:    BITB    #1,EOP.FLAG                     ;                       3158
00044C  001772                              BEQ     10$
000442  052777  040000  000000G            12$:    BIS     #40000,@DCT.ADDR                ;                       3161
000450  005726                              TST     (SP)+                                   ;                       3044
000452  000207                              RTS     PC

; Routine Size:  150 words,     Routine Base:  $CODE$ + 7470
; Maximum stack depth per invocation:  11 words


;   3165  1
```

```
;    3166  1     GLOBAL routine MD_INIT : novalue =
;    3167  1
;    3168  1     !+
;    3169  1     !       THIS ROUTINE IS CALLED BY ROUTINE MULTI_DRIVE TO INITIALIZE DATA ITEMS
;    3170  1     !       USED BY THE MULTI-DRIVE TEST.
;    3171  1     !-
;    3172  1
;    3173  2         begin
;    3174  2
;    3175  2     !!ZZZ    local
;    3176  2     !!ZZZ    AVG_XFER_SIZE : word,                          ! SIZE (BYTES) OF AN AVERAGE I/O XFER
;    3177  2     !!ZZZ    QUICK_PASS_CNT : word;                         ! AVG NO. OF I/O OPERATIONS IN A QUICK PASS
;    3178  2
;    3179  2         if not .INIT_OCCURED                               ! IF THIS IS A START
;    3180  2         then
;    3181  2             INIT_IO_BUFF ();                               ! PARTITION FREE MEMORY INTO I/O BUFFERS
;    3182  2
;    3183  2         if (.ENTRY_REASON neq CONT) and                    ! IF START, RESTART, OR PWR FAIL
;    3184  3            (.ENTRY_REASON neq NEW_PASS)
;    3185  2         then
;    3186  2
;    3187  2             incr CTLR from 0 to (MAX_CTLR - 1) do
;    3188  3                 begin
;    3189  3                 SET_CPAR (.CTLR);
;    3190  3
;    3191  4                 INCR DISK FROM (0 + OF_UN) TO (3 * UNIT_SIZE                    !ZZZ
;    3192  3                      + OF_UN) BY UNIT_SIZE DO                                   !ZZZ
;    3193  4                      BEGIN                                                      !ZZZ
;    3194  4                      SET_UPAR (.DISK);                                          !ZZZ
;    3195  4                      DPST [.L$LUN] = DP_CNT;       !INIT DATA PTRN SEQ TABLE!ZZZ
;    3196  3                      END;                                                       !ZZZ
;    3197  3
;    3198  2                 END;                                                           !ZZZ
;    3199  2             INCR COUNT FROM 0 TO (QIO_PER_CTLR * MAX_CTLR - 1) DO   !INIT   !ZZZ
;    3200  2                  BUFF_OWN [.COUNT] = -1;              !I/O BUFF ALLOC TABLE  !ZZZ
;    3201  1             END;                                      !END MD_INIT           !ZZZ


                                    .SBTTL  MD.INIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          MD.INIT::
                                    JSR     R1,$SAVE2                       ;                       3166
000004  132737  000001  000000G            BITB    #1,INIT.OCCURED         ;                       3179
000012  001002                      BNE     1$
000014  004737  000000V                    JSR     PC,INIT.IO.BUFF         ;                       3181
000020  123727  000000G 000003   1$:      CMPB    ENTRY.REASON,#3         ;                       3183
000026  001433                      BEQ     4$
000030  123727  000000G 000005            CMPB    ENTRY.REASON,#5         ;                       3184
000036  001427                      BEQ     4$
000040  005002                      CLR     R2                              ; CTLR                  3187
000042  010246                 2$:  MOV     R2,-(SP)                        ; CTLR,*                3189
000044  004737  000000G            JSR     PC,SET.CPAR
000050  012701  000003            MOV     #3,R1                           ; *,DISK                3191
000054  010116                 3$:  MOV     R1,(SP)                         ; DISK,*                3194
```

```
000056  004737  000000G                        JSR     PC,SET.UPAR
000062  013700  000000G                        MOV     L$LUN,R0                        ;                       3195
000066  112760  000025  000050'                MOVB    #25,DPST(R0)
000074  062701  000012                         ADD     #12,R1                          ; *,DISK                3191
000100  020127  000041                         CMP     R1,#41                          ; DISK,*
000104  003763                                 BLE     3$
000106  005726                                 TST     (SP)+                           ;                       3188
000110  005202                                 INC     R2                              ; CTLR                  3187
000112  000243                                 .WORD CLV!CLC

000114  003752                                 BLE     2$
000116  005000                         4$:     CLR     R0                              ; COUNT                 3199
000120  112760  000377  000000G        5$:     MOVB    #377,BUFF.OWN(R0)               ; *,*(COUNT)            3200
000126  005200                                 INC     R0                              ; COUNT                 3199
000130  020027  000007                         CMP     R0,#7                           ; COUNT,*
000134  003771                                 BLE     5$
000136  000207                                 RTS     PC                              ;                       3166

; Routine Size:  48 words,      Routine Base:  $CODE$ + 10144
; Maximum stack depth per invocation:  5 words


;   3202  1
```

```
;    3203  1     GLOBAL routine INIT_IO_BUFF : novalue =
;    3204  1
;    3205  1     !+
;    3206  1     !       THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE TEST IS FIRST
;    3207  1     !       STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;    3208  1     !       COLLECTION OF I/O BUFFERS. THE SIZE OF EACH I/O BUFFER IS DETERMINED
;    3209  1     !       BY A NUMBER OF FACTORS, INCLUDING THE NUMBER OF UNITS, THE NUMBER OF
;    3210  1     !       CONTROLLERS, AND THE SIZE OF FREE MEMORY.
;    3211  1     !
;    3212  1     !       ONCE THE BUFFER SIZE IS DETERMINED, THE NUMBER OF I/O BUFFERS IS
;    3213  1     !       CALCULATED. FINALLY, THE BUFFER ADDRESS (BUFF_ADDR) TABLE IS LOADED
;    3214  1     !       WITH FIXED BUFFER DESCRIPTORS THAT ARE USED IN THE ALLOCATION AND
;    3215  1     !       DEALLOCATION PROCESS.
;    3216  1     !
;    3217  1     !       IMPLICIT INPUTS:
;    3218  1     !               CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
;    3219  1     !               L$UNIT - THE NUMBER OF UNITS AVAILABLE FOR TESTING
;    3220  1     !               FREE_MEM_ADDR - START OF FREE MEMORY
;    3221  1     !-
;    3222  1
;    3223  2         begin
;    3224  2         BUFF_ADDR [0] = (.FREE_MEM_ADDR + 2 + 1) and %o'177776';                ! START OF READ/WRITE BUFFERS
;    3225  2
;    3226  2         while (.BUFF_ADDR [0] and %o'37') neq 0 do                              ! FORCE FIRST I/O BUFFER TO START
;    3227  2             BUFF_ADDR [0] = .BUFF_ADDR [0] + 2;                                 ! ON EVEN BOUNDARY
;    3228  2
;    3229  2         BYTS_PER_QIO = ((.DRS_START - .BUFF_ADDR [0]) / (QIO_PER_CTLR * MAX_CTLR)) and %o'177740';
;    3230  2                                                                                 ! MAX TRANSFER SIZE
;    3231  2
;    3232  2         if .BYTS_PER_QIO gtru MAX_XFER_SIZE
;    3233  2         then
;    3234  2             BYTS_PER_QIO = MAX_XFER_SIZE;                                       ! ADJUST TRANSFER SIZE LOWER
;    3235  2
;    3236  2         if .BYTS_PER_QIO lssu 32
;    3237  2         then
;    3238  3             begin
;    3239  3             ERRSF (2, EGS_02, 0);                                               ! ERROR IF NOT ENOUGH MEMORY
;    3240  3             DOCLN;
;    3241  2             end;
;    3242  2
;    3243  2         if (QIO_PER_CTLR * MAX_CTLR) gtru 1
;    3244  2         then
;    3245  2
;    3246  2             incr index from 1 to (QIO_PER_CTLR * MAX_CTLR - 1) do               ! INIT REMAINING TABLE ENTRIES
;    3247  2                 BUFF_ADDR [.index] = .BUFF_ADDR [.index - 1] + .BYTS_PER_QIO;   ! FIXED BUFFER ADDRESS
;    3248  2
;    3249  1         end;                                                                    ! ROUTINE INIT_IO_BUFF


                                        .SBTTL  INIT.IO.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          INIT.IO.BUFF::
                                        JSR     R1,$SAVE3                       ;                                       3203
000004  013700  000000G                 MOV     FREE.MEM.ADDR,R0                ;                                       3224
```

```
000010  062700  000003                        ADD     #3,R0
000014  010037  000000G                       MOV     R0,BUFF.ADDR
000020  042737  000001  000000G               BIC     #1,BUFF.ADDR
000026  032737  000037  000000G       1$:     BIT     #37,BUFF.ADDR                   ;                       3226
000034  001404                              BEQ     2$
000036  062737  000002  000000G               ADD     #2,BUFF.ADDR                    ;                       3227
000044  000770                              BR      1$                              ;                       3226
000046  013746  001164'              2$:     MOV     DRS.START,-(SP)                 ;                       3229
000052  163716  000000G                       SUB     BUFF.ADDR,(SP)
000056  012746  000010                        MOV     #10,-(SP)
000062  004737  000000G                       JSR     PC,BL$DIV
000066  010037  000000G                       MOV     R0,BYTS.PER.QIO
000072  042737  000037  000000G               BIC     #37,BYTS.PER.QIO
000100  023727  000000G 001400                CMP     BYTS.PER.QIO,#1400              ;                       3232
000106  101403                              BLOS    3$
000110  012737  001400  000000G               MOV     #1400,BYTS.PER.QIO              ;                       3234
000116  023727  000000G 000040      3$:     CMP     BYTS.PER.QIO,#40                ;                       3236
000124  103005                              BHIS    4$
000126  104454                              TRAP    54                              ;                       3239
000130  000002                              .WORD   2
000132  000000G                             .WORD   EGS.02
000134  000000                              .WORD   0
000136  104444                              TRAP    44
000140  012702  000001              4$:     MOV     #1,R2                           ; *,INDEX               3243
000144  010200                      5$:     MOV     R2,R0                           ; INDEX,*               3247
000146  006300                              ASL     R0
000150  010201                              MOV     R2,R1                           ; INDEX,*
000152  006301                              ASL     R1
000154  016103  177776G                      MOV     BUFF.ADDR-2(R1),R3
000160  063703  000000G                       ADD     BYTS.PER.QIO,R3
000164  010360  000000G                       MOV     R3,BUFF.ADDR(R0)
000170  005202                              INC     R2                              ; INDEX                 3243
000172  020227  000007                      CMP     R2,#7                           ; INDEX,*
000176  003762                              BLE     5$
000200  022626                              CMP     (SP)+,(SP)+                     ;                       3223
000202  000207                              RTS     PC                              ;                       3203

; Routine Size:  66 words,      Routine Base:  $CODE$ + 10304
; Maximum stack depth per invocation:  8 words
```

```
;    3250  1     GLOBAL routine QIO_OK =
;    3251  1
;    3252  1
;    3253  1     !*
;    3254  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
;    3255  1     !       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
;    3256  1     !       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
;    3257  1     !       THE CONTROLLER MEETS 3 REQUIREMENTS:
;    3258  1     !
;    3259  1     !               A. THE CONTROLLER IS ONLINE;
;    3260  1     !               B. THE NUMBER OF OUTSTANDING QIOs IS AT LEAST 2 LESS THAN THE
;    3261  1     !                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
;    3262  1     !               C. THERE IS AT LEAST ONE DISK ONLINE TO THE CONTROLLER.
;    3263  1     !
;    3264  1     !       IF ANY OF THESE TEST FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
;    3265  1     !
;    3266  1     !       IMPLICIT INPUTS:
;    3267  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3268  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3269  1     !-
;    3270  1
;    3271  1
;    3272  1         if (.CST_ADDR [STATE] eql ONLINE) and                   ! IF CONTROLLER IS ONLINE
;    3273  1            (not .EOP_FLAG) and
;    3274  1            ((.QIO [.CCTLR] + 2) lequ QIO_PER_CTLR) and          ! IF OUTSTANDING QIO COUNT IS O.K.
;    3275  2            (.CST_ADDR [U_CNT] neq 0)                            ! IF THERE IS VALID UNI
;    3276  2
;    3277  1         then
;    3278  1             return TRUE                                         ! "TRUE" EXIT POINT
;    3279  1
;    3280  1         else
;    3281  1             return FALSE;                                       ! "FALSE" EXIT POINT


                                        .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  013700  000000G           QIO.OK::MOV     CST.ADDR,R0                     ;                       3272
000004  005760  000002                   TST     2(R0)
000010  100027                           BPL     1$
000012  132737  000001  000000G          BITB    #1,EOP.FLAG                     ;                       3273
000020  001023                           BNE     1$
000022  013700  000000G                  MOV     CCTLR,R0                        ;                       3274
000026  116000  000000G                  MOVB    QIO(R0),R0
000032  042700  177400                   BIC     #177400,R0
000036  062700  000002                   ADD     #2,R0
000042  020027  000010                   CMP     R0,#10
000046  101010                           BHI     1$
000050  013700  000000G                  MOV     CST.ADDR,R0                     ;                       3275
000054  105760  000005                   TSTB    5(R0)
000060  001403                           BEQ     1$
000062  012700  000001                   MOV     #1,R0                           ;                       3281
000066  000207                           RTS     PC
000070  005000                   1$:     CLR     R0
000072  000207                           RTS     PC                              ;                       3250
```

```
; Routine Size:  30 words,      Routine Base:  $CODE$ + 10510
; Maximum stack depth per invocation:  0 words


;   3282  1
```

```
;    3283  1     GLOBAL routine QIO_OUT =
;    3284  1
;    3285  1     !+
;    3286  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
;    3287  1     !       END OF THE MULTI-DRIVE TEST. ITS PURPOSE IS TO EXAMINE THE QIO VECTOR
;    3288  1     !       FOR ANY OUTSTANDING QIOs ON ANY CONTROLLER. A VALUE OF "TRUE" IS
;    3289  1     !       RETURNED IF THERE IS AT LEAST ONE QIO OUTSTANDING ON ANY CONTROLLER.
;    3290  1     !       OTHERWISE, "FALSE" IS RETURNED INDICATING NO OUTSTANDING QIOs.
;    3291  1     !-
;    3292  1
;    3293  2         begin
;    3294  2
;    3295  2         incr CTLR from 0 to (MAX_CTLR - 1) do
;    3296  3             begin
;    3297  3             SET_CPAR (.CTLR);                       ! SET UP CURRENT CONTROLLER PARAMETERS
;    3298  3
;    3299  3             if .CST_ADDR [STATE] eql ONLINE         ! IF CONTROLLER IS ONLINE
;    3300  3             then
;    3301  3                 return TRUE;
;    3302  3
;    3303  2             end;
;    3304  2
;    3305  2         return FALSE;                               ! EXIT - NO CONTROLLERS ONLINE
;    3306  1         end;


                                .SBTTL  QIO.OUT MULTI-DRIVE TEST ROUTINES
000000  010146          QIO.OUT::
                                MOV     R1,-(SP)                        ;                       3283
000002  005001                  CLR     R1                              ; CTLR                  3295
000004  010146          1$:     MOV     R1,-(SP)                        ; CTLR,*                3297
000006  004737  000000G         JSR     PC,SET.CPAR
000012  013700  000000G         MOV     CST.ADDR,R0                     ;                       3299
000016  005760  000002          TST     2(R0)
000022  100004                  BPL     2$
000024  005726                  TST     (SP)+                           ;                       3301
000026  012700  000001          MOV     #1,R0
000032  000405                  BR      3$
000034  005726          2$:     TST     (SP)+                           ;                       3296
000036  005201                  INC     R1                              ; CTLR                  3295
000040  000243                  .WORD CLV!CLC

000042  003760                  BLE     1$
000044  005000                  CLR     R0                              ;                       3293
000046  012601          3$:     MOV     (SP)+,R1                        ;                       3283
000050  000207                  RTS     PC

; Routine Size:  21 words,      Routine Base:  $CODE$ + 10604
; Maximum stack depth per invocation:  3 words
```

```
;    3307  1     GLOBAL routine GIO_GEN : novalue =
;    3308  1
;    3309  1     !+
;    3310  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR AN ONLINE
;    3311  1     !       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
;    3312  1     !       RESPONSIBLE FOR SECURING ONE OR TWO MSCP PACKETS AND LOADING THEM
;    3313  1     !       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
;    3314  1     !       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
;    3315  1     !       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
;    3316  1     !
;    3317  1     !       EACH FIELD OF THE PACKET(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
;    3318  1     !       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
;    3319  1     !       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
;    3320  1     !
;    3321  1     !       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
;    3322  1     !       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
;    3323  1     !       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
;    3324  1     !       GENERATED WITH THE SAME LBN AND BYTE COUNT.
;    3325  1     !
;    3326  1     !       AFTER THE PACKET(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
;    3327  1     !       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
;    3328  1     !       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
;    3329  1     !       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
;    3330  1     !       CONTAIN VALID MSCP PACKET INDECES, OR -1.
;    3331  1     !
;    3332  1     !       IMPLICIT INPUTS:
;    3333  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3334  1     !-
;    3335  1
;    3336  2         begin
;    3337  2         MX2 = -1;                                   ! ASSUME FAILURE IN SECURING 2ND PACKET
;    3338  2
;    3339  2         if (MX1 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 1ST PACKET. IF FAILURE
;    3340  2         then
;    3341  2             return;                                 ! NO POINT IN CONTINUING
;    3342  2
;    3343  2         if (MX2 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 2ND PACKET. IF FAILURE
;    3344  2         then
;    3345  3             begin
;    3346  3             PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
;    3347  3             MX1 = -1;                               ! INDICATE FAILURE
;    3348  3             return;                                 ! DONE
;    3349  2             end;
;    3350  2
;    3351  2         MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);     ! CALCULATE STARTING ADDRESSES
;    3352  2         MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);     !     OF BOTH PACKETS
;    3353  2         GET_RANDOM ();                              ! GENERATE A SET OF RANDOM NUMBERS
;    3354  2         QIO_UNIT ();                                ! LOAD RANDOM UNIT NUMBER INTO PACKETS
;    3355  2
;    3356  2         if .EOP_FLAG                                ! RETURN IF NO UNIT ONLINE
;    3357  2         then
;    3358  2             return;
;    3359  2
```

```
;    3360  2          QIO_FUNC ();                          ! LOAD RANDOM FUNCTION CODE (OPCODE)
;    3361  2          QIO_LBN ();                           ! LOAD LBN (RANDOM OR SEQUENTIAL)
;    3362  2          QIO_SIZE ();                          ! LOAD RANDOM BYTE COUNT
;    3363  2          GET_IO_BUFF (MAD1 [BUF_0]);           ! TRY TO GET AN I/O BUFFER
;    3364  2
;    3365  2          if .MX2 geq 0                         ! IF TWO QIOs ARE TO BE ISSUED
;    3366  2          then
;    3367  3              begin
;    3368  3              GET_IO_BUFF (MAD2 [BUF_0]);       ! TRY TO GET 2ND I/O BUFFER
;    3369  3
;    3370  3              if .MAD2 [BUF_0] eqla 0           ! IF 2ND BUFFER ALLOCATION FAILED
;    3371  3              then
;    3372  4                  begin
;    3373  4
;    3374  4                  if .MAD1 [BUF_0] neqa 0       ! IF 1ST I/O BUFFER WAS ALLOCATED
;    3375  4                  then
;    3376  5                      begin
;    3377  5                      PUT_IO_BUFF (MAD1 [BUF_0]);   ! RETURN 1ST I/O BUFFER TO POOL
;    3378  5                      MAD1 [BUF_0] = 0;             ! MARK IT AS FAILED
;    3379  4                      end;
;    3380  4
;    3381  4                  PUT_PKT (.MX2);               ! RETURN 2ND PACKET TO POOL
;    3382  4                  MX2 = -1;                     ! INDICATE FAILURE
;    3383  3                  end;                          ! IF 2ND I/O BUFFER ALLOCATION FAILED
;    3384  3
;    3385  2              end;                              ! IF TWO QIOs ARE TO BE ISSUED
;    3386  2
;    3387  2          if .MAD1 [BUF_0] eqla 0               ! IF 1ST I/O BUFFER ALLOCATION FAILED
;    3388  2          then
;    3389  3              begin
;    3390  3              PUT_PKT (.MX1);                   ! RETURN 1ST PACKET TO POOL
;    3391  3              MX1 = -1;                         ! INDICATE FAILURE
;    3392  3              end
;    3393  2          else
;    3394  2
;    3395  2              if .MAD1 [OPCODE] eql OP_WRT      ! OTHERWISE, IF 1ST OPCODE IS A WRITE  (ALL IS O.K.)
;    3396  2              then
;    3397  2                  FILL_BUFF ();                 ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;    3398  2
;    3399  1          end;                                  ! ROUTINE QIO_GEN

                                        .SBTTL  QIO.GEN MULTI-DRIVE TEST ROUTINES
000000  012737  177777  000114'    QIO.GEN::
                                        MOV     #-1,MX2                         ;                       3337
000006  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3339
000012  004737  000000G                 JSR     PC,GET.PKT
000016  010037  000112'                 MOV     R0,MX1
000022  005726                          TST     (SP)+
000024  005700                          TST     R0                              ; MX1
000026  002563                          BLT     6$                              ;                       3341
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3343
000034  004737  000000G                 JSR     PC,GET.PKT
```

```
000040  010037  000114'                      MOV     R0,MX2
000044  005726                               TST     (SP)+
000046  005700                               TST     R0                      ; MX2
000050  002011                               BGE     1$
000052  013746  000112'                      MOV     MX1,-(SP)               ;                       3346
000056  004737  000000G                      JSR     PC,PUT.PKT
000062  012737  177777  000112'             MOV     #-1,MX1                 ;                       3347
000070  005726                               TST     (SP)+                   ;                       3348
000072  000207                               RTS     PC                      ;                       3345
000074  013746  000112'              1$:     MOV     MX1,-(SP)               ;                       3351
000100  012746  000106                       MOV     #106,-(SP)
000104  004737  000000G                      JSR     PC,BL$MUL
000110  062700  000000G                      ADD     #MSCP.PKT,R0
000114  010037  000116'                      MOV     R0,MAD1
000120  013716  000114'                      MOV     MX2,(SP)                ;                       3352
000124  012746  000106                       MOV     #106,-(SP)
000130  004737  000000G                      JSR     PC,BL$MUL
000134  062700  000000G                      ADD     #MSCP.PKT,R0
000140  010037  000120'                      MOV     R0,MAD2
000144  004737  000000V                      JSR     PC,GET.RANDOM           ;                       3353
000150  004737  000000V                      JSR     PC,QIO.UNIT             ;                       3354
000154  132737  000001  000000G             BITB    #1,EOP.FLAG             ;                       3356
000162  001103                               BNE     5$                      ;                       3307
000164  004737  000000V                      JSR     PC,QIO.FUNC             ;                       3360
000170  004737  000000V                      JSR     PC,QIO.LBN              ;                       3361
000174  004737  000000V                      JSR     PC,QIO.SIZE             ;                       3362
000200  013716  000116'                      MOV     MAD1,(SP)               ;                       3363
000204  062716  000032                       ADD     #32,(SP)
000210  004737  000000G                      JSR     PC,GET.IO.BUFF
000214  005737  000114'                      TST     MX2                     ;                       3365
000220  002437                               BLT     3$
000222  013716  000120'                      MOV     MAD2,(SP)               ;                       3368
000226  062716  000032                       ADD     #32,(SP)
000232  004737  000000G                      JSR     PC,GET.IO.BUFF
000236  013700  000120'                      MOV     MAD2,R0                 ;                       3370
000242  005760  000032                       TST     32(R0)
000246  001024                               BNE     3$
000250  013700  000116'                      MOV     MAD1,R0                 ;                       3374
000254  062700  000032                       ADD     #32,R0
000260  005710                               TST     (R0)
000262  001407                               BEQ     2$
000264  010016                               MOV     R0,(SP)                 ;                       3377
000266  004737  000000G                      JSR     PC,PUT.IO.BUFF
000272  013700  000116'                      MOV     MAD1,R0                 ;                       3378
000276  005060  000032                       CLR     32(R0)
000302  013716  000114'              2$:     MOV     MX2,(SP)                ;                       3381
000306  004737  000000G                      JSR     PC,PUT.PKT
000312  012737  177777  000114'             MOV     #-1,MX2                 ;                       3382
000320  013700  000116'              3$:     MOV     MAD1,R0                 ;                       3387
000324  005760  000032                       TST     32(R0)
000330  001010                               BNE     4$
000332  013716  000112'                      MOV     MX1,(SP)                ;                       3390
000336  004737  000000G                      JSR     PC,PUT.PKT
```

```
000342  012737  177777  000112'              MOV     #-1,MX1                 ;                3391
000350  000410                               BR      5$                      ;                3387
000352  013700  000116'              4$:     MOV     MAD1,R0                 ;                3395
000356  126027  000022  000042              CMPB    22(R0),#42
000364  001002                               BNE     5$
000366  004737  000000V                      JSR     PC,FILL.BUFF            ;                3397
000372  062706  000006              5$:     ADD     #6,SP                   ;                3336
000376  000207                      6$:     RTS     PC                      ;                3307

; Routine Size:  128 words,     Routine Base:  $CODE$ + 10656
; Maximum stack depth per invocation:  4 words
```

```
;    3400  1     GLOBAL routine GET_RANDOM : novalue =
;    3401  1
;    3402  1     !+
;    3403  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;    3404  1     !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;    3405  1     !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;    3406  1     !       OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;    3407  1     !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;    3408  1     !-
;    3409  1
;    3410  2         begin
;    3411  2
;    3412  2         own
;    3413  2             SEED : word initial (173),
;    3414  2             NEXT_RANDNUM : word initial (245);
;    3415  2
;    3416  2         incr COUNT from 0 to (RDM_LEN - 1) do
;    3417  3             begin
;    3418  3             SEED = (.SEED + .NEXT_RANDNUM + 1) * 4;
;    3419  3             NEXT_RANDNUM = (.NEXT_RANDNUM / 4) + .SEED;
;    3420  3             RANDOM [.COUNT] = .NEXT_RANDNUM;
;    3421  2             end;
;    3422  2
;    3423  1         end;


001204                                  .PSECT  $GGG$,  RO
001204  000255                SEED:     .WORD   255
001206  000365                NEXT.RANDNUM:
                                        .WORD   365


                                        .SBTTL  GET.RANDOM MULTI-DRIVE TEST ROUTINES
011256                                  .PSECT  $CODE$,  RO

000000  004137  000000G       GET.RANDOM::
                                        JSR     R1,$SAVE3                       ;                       3400
000004  013703  001204'                 MOV     SEED,R3                         ;                       3418
000010  013702  001206'                 MOV     NEXT.RANDNUM,R2
000014  005001                          CLR     R1                              ; COUNT                 3416
000016  010200                1$:       MOV     R2,R0                           ;                       3418
000020  060300                          ADD     R3,R0
000022  006300                          ASL     R0
000024  006300                          ASL     R0
000026  010037  001204'                 MOV     R0,SEED
000032  062737  000004  001204'         ADD     #4,SEED
000040  010246                          MOV     R2,-(SP)                        ;                       3419
000042  012746  000004                  MOV     #4,-(SP)
000046  004737  000000G                 JSR     PC,BL$DIV
000052  013703  001204'                 MOV     SEED,R3
000056  060300                          ADD     R3,R0
000060  010037  001206'                 MOV     R0,NEXT.RANDNUM
```

```
000064  010002                          MOV     R0,R2                   ; NEXT.RANDNUM,*                3420
000066  010261  000000G                 MOV     R2,RANDOM(R1)           ; *,*(COUNT)
000072  022626                          CMP     (SP)+,(SP)+             ;                               3417
000074  062701  000002                  ADD     #2,R1                   ; *,COUNT                       3416
000100  020127  000036                  CMP     R1,#36                  ; COUNT,*
000104  003744                          BLE     1$
000106  000207                          RTS     PC                      ;                               3400

; Routine Size:  36 words,      Routine Base:  $CODE$ + 11256
; Maximum stack depth per invocation:  7 words
```

```
;    3424  1     GLOBAL routine QIO_UNIT : novalue =
;    3425  1
;    3426  1     !+
;    3427  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;    3428  1     !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;    3429  1     !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;    3430  1     !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;    3431  1     !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;    3432  1     !
;    3433  1     !       IMPLICIT INPUTS:
;    3434  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3435  1     !
;    3436  1     !       IMPLICIT OUTPUTS:
;    3437  1     !               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
;    3438  1     !                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
;    3439  1     !-
;    3440  1
;    3441  2         begin
;    3442  2
;    3443  2         local
;    3444  2             MOD_COUNT : byte,
;    3445  2             TBL_COUNT : byte,
;    3446  2             SELECT_RD : byte initial (byte (TRUE)),
;    3447  2     !ZZZ    RD_COUNT : word initial (0),
;    3448  2             RX_COUNT : word initial (0);
;    3449  2
;    3450  2     !
;    3451  2     ! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
;    3452  2     ! SELECTED VIA THE SOFTWARE PARAMETERS
;    3453  2     !
;    3454  2     ! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
;    3455  2     ! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
;    3456  2     ! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
;    3457  2     ! MODE
;    3458  2     !
;    3459  2
;    3460  2             RD_COUNT = 0;                                           !ZZZ
;    3461  2             RX_COUNT = 0;                                           !ZZZ
;    3462  2
;    3463  2             incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3464  2
;    3465  2                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3466  2                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3467  3                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3468  2                 then
;    3469  2
;    3470  3                     if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3471  2                     then
;    3472  2                         RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
;    3473  2                     else
;    3474  2                         RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;    3475  2
;    3476  2
```

```
;    3477  2
;    3478  2          if (not BIT_TST (SWP_FLAGS, SWF_RDM)) and           ! NOT RANDOM MODE
;    3479  3             (not BIT_TST (SWP_FLAGS, SWF_SEQ))               ! NOT RANDOM SEQUEUNTIAL MODE
;    3480  2          then
;    3481  2
;    3482  2              if (.BST_CNT neq 0) and
;    3483  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3484  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3485  3                 (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3486  2              then
;    3487  3                  begin                                       ! ALREADY WITHIN DEVICE
;    3488  3                  BST_CNT = .BST_CNT - 1;
;    3489  3                  SET_UPAR (.BST_DEV);
;    3490  3                  MAD1 [DK_NUM] = .CDISK;
;    3491  3                  MAD2 [DK_NUM] = .CDISK;
;    3492  3                  return;
;    3493  3                  end
;    3494  2              else
;    3495  3                  begin                                       ! GET NEW DEVICE
;    3496  3
;    3497  3      !ZZZ        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3498  3      !ZZZ
;    3499  3      !ZZZ            if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3500  3      !ZZZ               (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3501  3      !ZZZ               (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3502  3      !ZZZ            then
;    3503  3      !ZZZ
;    3504  3      !ZZZ                if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3505  3      !ZZZ                then
;    3506  3      !ZZZ                    RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
;    3507  3      !ZZZ                else
;    3508  3      !ZZZ                    RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;    3509  3
;    3510  3                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3511  4                      begin
;    3512  4
;    3513  4                      if (.BST_DEV eql 0) or
;    3514  5                         (.BST_DEV eql ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN))
;    3515  4                      then
;    3516  4                          BST_DEV = OF_UN
;    3517  4                      else
;    3518  4                          BST_DEV = .BST_DEV + UNIT_SIZE;
;    3519  4
;    3520  4                      if (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3521  4                         (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3522  5                         (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3523  4                      then
;    3524  5                          begin
;    3525  5
;    3526  5                          if .CST_ADDR [.BST_DEV + OF_DATA, D_TYPE] eql REMOVABLE
;    3527  5                          then
;    3528  5                              BST_CNT = .RX_MAX_SEQ_CNT / .RX_COUNT
;    3529  5                          else
```

```
;  3530  5                        BST_CNT = .RD_MAX_SEQ_CNT / .RD_COUNT;
;  3531  5
;  3532  5                    ;if .BST_CNT eql 0
;  3533  5                    then
;  3534  5                        BST_CNT = 1;
;  3535  5
;  3536  5                    SET_UPAR (.BST_DEV);
;  3537  5                    MAD1 [DK_NUM] = .CDISK;
;  3538  5                    MAD2 [DK_NUM] = .CDISK;
;  3539  5                    return;
;  3540  4                    end;
;  3541  4
;  3542  3                end;
;  3543  3
;  3544  2            end;
;  3545  2
;  3546  2        !
;  3547  2        ! RANDOM SELECTION OF DRIVES
;  3548  2        !
;  3549  2        !
;  3550  2        ! DETERMINE IF RD51/52s ARE TO BE SELECTED
;  3551  2        !
;  3552  2
;  3553  2            ;if ((.RANDOM [RDM_LEN - 1] and %o'077777') mod 100) gequ .SWP_RAT
;  3554  2            then
;  3555  2                SELECT_RD = FALSE;
;  3556  2
;  3557  2        !
;  3558  2        ! IF RD51/52s SELECTED
;  3559  2        !
;  3560  2        ! COUNT NUMBER OF RD51/52s AVAILABLE
;  3561  2        !
;  3562  2
;  3563  2            if .SELECT_RD
;  3564  2            then
;  3565  3                begin
;  3566  3                MOD_COUNT = 0;                                  ! COUNT THE NUMBER OF RDs UNDER TEST
;  3567  3
;  3568  3                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3569  3
;  3570  3                    if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  3571  3                       (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3572  3                       (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED) and
;  3573  4                       (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  3574  3                    then
;  3575  4                        begin
;  3576  4                        STORAGE [.MOD_COUNT] = .OFFSET;
;  3577  4                        MOD_COUNT = .MOD_COUNT + 1;
;  3578  3                        end;
;  3579  3
;  3580  3        !
;  3581  3        ! SELECT ON OF THE RD51/52s
;  3582  3        !
```

```
;    3583  3
;    3584  3              if .MOD_COUNT neq 0                              ! IF AT LEAST ONE RD51/52 PRESENT
;    3585  3              then
;    3586  4                  begin
;    3587  4                  TBL_COUNT = 0;
;    3588  4
;    3589  4                  do
;    3590  5                      begin
;    3591  5                      SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3592  5                      TBL_COUNT = .TBL_COUNT + 1;
;    3593  5                      end
;    3594  5                  until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3595  5                         (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3596  4                         (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3597  4                        (.TBL_COUNT eql RDM_LEN);
;    3598  4
;    3599  4                  MAD1 [DK_NUM] = .CDISK;
;    3600  4                  MAD2 [DK_NUM] = .CDISK;
;    3601  4                  return;
;    3602  3                  end;
;    3603  3
;    3604  2              end;
;    3605  2
;    3606  2          !
;    3607  2          ! IF NO RD51/52 SELECTED, SELECT AN RX50
;    3608  2          !
;    3609  2          ! COUNT THE NUMBER OF RX50s
;    3610  2          !
;    3611  2
;    3612  2              MOD_COUNT = 0;
;    3613  2
;    3614  2              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3615  2
;    3616  2                  if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3617  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3618  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) and
;    3619  3                     (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3620  2                  then
;    3621  3                      begin
;    3622  3                      STORAGE [.MOD_COUNT] = .OFFSET;
;    3623  3                      MOD_COUNT = .MOD_COUNT + 1;
;    3624  2                      end;
;    3625  2
;    3626  2          !
;    3627  2          ! AND CHOOSE ONE!
;    3628  2          !
;    3629  2
;    3630  2              if .MOD_COUNT neq 0
;    3631  2              then
;    3632  3                  begin
;    3633  3                  TBL_COUNT = 0;
;    3634  3
;    3635  3                  do
```

```
;    3636  4                  begin
;    3637  4                  SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3638  4                  TBL_COUNT = .TBL_COUNT + 1;
;    3639  4                  end
;    3640  4              until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3641  4                     (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3642  3                     (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3643  3                    (.TBL_COUNT eql RDM_LEN);
;    3644  3
;    3645  3              MAD1 [DK_NUM] = .CDISK;
;    3646  3              MAD2 [DK_NUM] = .CDISK;
;    3647  3              return;
;    3648  2              end;
;    3649  2
;    3650  2          !
;    3651  2          ! IF NO UNIT SELECTED SO FAR BY ABOVE METHOD, SELECT ANY ONE AT RANDOM
;    3652  2          !
;    3653  2          ! COUNT ALL UNITS AVAILABLE
;    3654  2          !
;    3655  2
;    3656  2              MOD_COUNT = 0;
;    3657  2
;    3658  2              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3659  2
;    3660  2                  if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3661  2                     (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3662  3                     (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3663  2                  then
;    3664  3                      begin
;    3665  3                      STORAGE [.MOD_COUNT] = .OFFSET;
;    3666  3                      MOD_COUNT = .MOD_COUNT + 1;
;    3667  2                      end;
;    3668  2
;    3669  2          !
;    3670  2          ! SELECT ANY ONE ONE UNIT AT RANDOM
;    3671  2          !
;    3672  2              if .MOD_COUNT neq 0
;    3673  2              then
;    3674  3                  begin
;    3675  3                  TBL_COUNT = 0;
;    3676  3
;    3677  3                  do
;    3678  4                      begin
;    3679  4                      SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3680  4                      TBL_COUNT = .TBL_COUNT + 1;
;    3681  4                      end
;    3682  4                  until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3683  4                         (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3684  3                         (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3685  3                        (.TBL_COUNT eql RDM_LEN);
;    3686  3
;    3687  3                  MAD1 [DK_NUM] = .CDISK;
;    3688  3                  MAD2 [DK_NUM] = .CDISK;
```

```
;    3689  3              return
;    3690  3              end
;    3691  3
;    3692  3      !
;    3693  3      ! DECLARE END-OF-PASS IF NO UNIT ONLINE
;    3694  3      !
;    3695  3
;    3696  2          else
;    3697  2              EOP_FLAG = TRUE;
;    3698  2
;    3699  1          end;                                    ! ROUTINE QIO_UNIT

                                    .SBTTL  QIO.UNIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G        QIO.UNIT::
                                    JSR     R1,$SAVE4                   ;                       3424
000004  112704  000001                MOVB    #1,R4                       ; *,SELECT.RD           3441
000010  005003                        CLR     R3                          ; RX.COUNT
000012  005037  000110'               CLR     RD.COUNT                    ;                       3460
000016  013702  000000G               MOV     CST.ADDR,R2                 ;                       3465
000022  012701  000006                MOV     #6,R1                       ; *,OFFSET              3463
000026  010100              1$:       MOV     R1,R0                       ; OFFSET,*              3465
000030  060200                        ADD     R2,R0
000032  032710  040000                BIT     #40000,(R0)
000036  001415                        BEQ     3$
000040  032710  020000                BIT     #20000,(R0)                 ;                       3466
000044  001412                        BEQ     3$
000046  032710  010000                BIT     #10000,(R0)                 ;                       3467
000052  001007                        BNE     3$
000054  132710  000020                BITB    #20,(R0)                    ;                       3470
000060  001403                        BEQ     2$
000062  005237  000110'               INC     RD.COUNT                    ;                       3472
000066  000401                        BR      3$                          ;                       3470
000070  005203              2$:       INC     R3                          ; RX.COUNT              3474
000072  062701  000024        3$:     ADD     #24,R1                      ; *,OFFSET              3463
000076  020127  000102                CMP     R1,#102                     ; OFFSET,*
000102  003751                        BLE     1$
000104  032737  000002  000000G       BIT     #2,SWP.FLAGS                ;                       3478
000112  001163                        BNE     13$
000114  032737  001000  000000G       BIT     #1000,SWP.FLAGS             ;                       3479
000122  001157                        BNE     13$
000124  005737  001152'               TST     BST.CNT                     ;                       3482
000130  001447                        BEQ     4$
000132  013700  001154'               MOV     BST.DEV,R0                  ;                       3483
000136  006300                        ASL     R0
000140  060200                        ADD     R2,R0
000142  032710  040000                BIT     #40000,(R0)
000146  001440                        BEQ     4$
000150  013700  001154'               MOV     BST.DEV,R0                  ;                       3484
000154  006300                        ASL     R0
000156  060200                        ADD     R2,R0
000160  032710  020000                BIT     #20000,(R0)
000164  001431                        BEQ     4$
```

```
000166  013700  001154'                        MOV     BST.DEV,R0                      ;                       3485
000172  006300                                  ASL     R0
000174  060200                                  ADD     R2,R0
000176  032710  010000                          BIT     #10000,(R0)
000202  001022                                  BNE     4$
000204  005337  001152'                        DEC     BST.CNT                         ;                       3488
000210  013746  001154'                        MOV     BST.DEV,-(SP)                   ;                       3489
000214  004737  000000G                       JSR     PC,SET.UPAR
000220  013700  000116'                        MOV     MAD1,R0                         ;                       3490
000224  013760  000000G 000016                MOV     CDISK,16(R0)
000232  013700  000120'                        MOV     MAD2,R0                         ;                       3491
000236  013760  000000G 000016                MOV     CDISK,16(R0)
000244  005726                                TST     (SP)+                           ;                       3492
000246  000207                                RTS     PC                              ;                       3487
000250  012702  000003              4$:       MOV     #3,R2                           ; *,OFFSET              3510
000254  013700  001154'              5$:       MOV     BST.DEV,R0                      ;                       3513
000260  001403                                BEQ     6$
000262  020027  000041                        CMP     R0,#41                          ;                       3514
000266  001004                                BNE     7$
000270  012737  000003  001154'      6$:       MOV     #3,BST.DEV                      ;                       3516
000276  000403                                BR      8$                              ;                       3513
000300  062737  000012  001154'      7$:       ADD     #12,BST.DEV                     ;                       3518
000306  013700  001154'              8$:       MOV     BST.DEV,R0                      ;                       3520
000312  006300                                ASL     R0
000314  063700  000000G                       ADD     CST.ADDR,R0
000320  032710  040000                        BIT     #40000,(R0)
000324  001451                                BEQ     12$
000326  032710  020000                        BIT     #20000,(R0)                     ;                       3521
000332  001446                                BEQ     12$
000334  032710  010000                        BIT     #10000,(R0)                     ;                       3522
000340  001043                                BNE     12$
000342  032710  000020                        BITB    #20,(R0)                        ;                       3526
000346  001004                                BNE     9$
000350  013746  001202'                        MOV     RX.MAX.SEQ.CNT,-(SP)            ;                       3528
000354  010346                                MOV     R3,-(SP)                        ; RX.COUNT,*
000356  000404                                BR      10$
000360  013746  001200'              9$:       MOV     RD.MAX.SEQ.CNT,-(SP)            ;                       3530
000364  013746  000110'                        MOV     RD.COUNT,-(SP)
000370  004737  000000G             10$:      JSR     PC,BL$DIV
000374  010037  001152'                        MOV     R0,BST.CNT
000400  001003                                BNE     11$                             ;                       3532
000402  012737  000001  001152'              MOV     #1,BST.CNT                      ;                       3534
000410  013716  001154'             11$:       MOV     BST.DEV,(SP)                    ;                       3536
000414  004737  000000G                       JSR     PC,SET.UPAR
000420  013700  000116'                        MOV     MAD1,R0                         ;                       3537
000424  013760  000000G 000016                MOV     CDISK,16(R0)
000432  013700  000120'                        MOV     MAD2,R0                         ;                       3538
000436  013760  000000G 000016                MOV     CDISK,16(R0)
000444  022626                                CMP     (SP)+,(SP)+                     ;                       3539
000446  000207                                RTS     PC                              ;                       3524
000450  062702  000012              12$:      ADD     #12,R2                          ; *,OFFSET              3510
000454  020227  000041                        CMP     R2,#41                          ; OFFSET,*
000460  003675                                BLE     5$
```

```
000462  013746  000036G                13$:    MOV     RANDOM+36,-(SP)                 ;                               3553
000466  042716  100000                        BIC     #100000,(SP)
000472  012746  000144                        MOV     #144,-(SP)
000476  004737  000000G                        JSR     PC,BL$MOD
000502  022626                                CMP     (SP)+,(SP)+
000504  020037  000000G                        CMP     R0,SWP.RAT
000510  103401                                BLO     14$
000512  105004                                CLRB    R4                              ; SELECT.RD                     3555
000514  006004                        14$:    ROR     R4                              ; SELECT.RD                     3563
000516  103105                                BCC     19$
000520  105003                                CLRB    R3                              ; MOD.COUNT                     3566
000522  012701  000003                        MOV     #3,R1                           ; *,OFFSET                      3568
000526  010100                        15$:    MOV     R1,R0                           ; OFFSET,*                      3570
000530  006300                                ASL     R0
000532  063700  000000G                        ADD     CST.ADDR,R0
000536  032710  040000                        BIT     #40000,(R0)
000542  001417                                BEQ     16$
000544  032710  020000                        BIT     #20000,(R0)                     ;                               3571
000550  001414                                BEQ     16$
000552  132710  000020                        BITB    #20,(R0)                        ;                               3572
000556  001411                                BEQ     16$
000560  032710  010000                        BIT     #10000,(R0)                     ;                               3573
000564  001006                                BNE     16$
000566  005000                                CLR     R0                              ;                               3576
000570  150300                                BISB    R3,R0                           ; MOD.COUNT,*
000572  006300                                ASL     R0
000574  010160  000064'                        MOV     R1,STORAGE(R0)                  ; OFFSET,*
000600  105203                                INCB    R3                              ; MOD.COUNT                     3577
000602  062701  000012                16$:    ADD     #12,R1                          ; *,OFFSET                      3568
000606  020127  000041                        CMP     R1,#41                          ; OFFSET,*
000612  003745                                BLE     15$
000614  105703                                TSTB    R3                              ; MOD.COUNT                     3584
000616  001445                                BEQ     19$
000620  105002                                CLRB    R2                              ; TBL.COUNT                     3587
000622  005000                        17$:    CLR     R0                              ;                               3591
000624  150200                                BISB    R2,R0                           ; TBL.COUNT,*
000626  006300                                ASL     R0
000630  016046  000000G                        MOV     RANDOM(R0),-(SP)
000634  042716  100000                        BIC     #100000,(SP)
000640  005046                                CLR     -(SP)
000642  110316                                MOVB    R3,(SP)                         ; MOD.COUNT,*
000644  004737  000000G                        JSR     PC,BL$MOD
000650  006300                                ASL     R0
000652  016016  000064'                        MOV     STORAGE(R0),(SP)
000656  004737  000000G                        JSR     PC,SET.UPAR
000662  105202                                INCB    R2                              ; TBL.COUNT                     3592
000664  022626                                CMP     (SP)+,(SP)+                     ;                               3590
000666  013700  000000G                        MOV     CUOFF,R0                        ;                               3594
000672  006300                                ASL     R0
000674  063700  000000G                        ADD     CST.ADDR,R0
000700  032710  040000                        BIT     #40000,(R0)
000704  001406                                BEQ     18$
000706  032710  020000                        BIT     #20000,(R0)                     ;                               3595
```

```
000712  001403                          BEQ     18$
000714  032710  010000                  BIT     #10000,(R0)             ;                       3596
000720  001510                          BEQ     24$
000722  120227  000020          18$:    CMPB    R2,#20                  ; TBL.COUNT,*           3597
000726  001335                          BNE     17$
000730  000504                          BR      24$
                                                                        ;                       3599
000732  105003                  19$:    CLRB    R3                      ; MOD.COUNT             3612
000734  012701  000003                  MOV     #3,R1                   ; *,OFFSET              3614
000740  010100                  20$:    MOV     R1,R0                   ; OFFSET,*              3616
000742  006300                          ASL     R0
000744  063700  000000G                 ADD     CST.ADDR,R0
000750  032710  040000                  BIT     #40000,(R0)
000754  001417                          BEQ     21$
000756  032710  020000                  BIT     #20000,(R0)             ;                       3617
000762  001414                          BEQ     21$
000764  132710  000020                  BITB    #20,(R0)                ;                       3618
000770  001011                          BNE     21$
000772  032710  010000                  BIT     #10000,(R0)             ;                       3619
000776  001006                          BNE     21$
001000  005000                          CLR     R0                      ;                       3622
001002  150300                          BISB    R3,R0                   ; MOD.COUNT,*
001004  006300                          ASL     R0
001006  010160  000064'                 MOV     R1,STORAGE(R0)          ; OFFSET,*
001012  105203                          INCB    R3                      ; MOD.COUNT             3623
001014  062701  000012          21$:    ADD     #12,R1                  ; *,OFFSET              3614
001020  020127  000041                  CMP     R1,#41                  ; OFFSET,*
001024  003745                          BLE     20$
001026  105703                          TSTB    R3                      ; MOD.COUNT             3630
001030  001445                          BEQ     25$
001032  105002                          CLRB    R2                      ; TBL.COUNT             3633
001034  005000                  22$:    CLR     R0                      ;                       3637
001036  150200                          BISB    R2,R0                   ; TBL.COUNT,*
001040  006300                          ASL     R0
001042  016046  000000G                 MOV     RANDOM(R0),-(SP)
001046  042716  100000                  BIC     #100000,(SP)
001052  005046                          CLR     -(SP)
001054  110316                          MOVB    R3,(SP)                 ; MOD.COUNT,*
001056  004737  000000G                 JSR     PC,BL$MOD
001062  006300                          ASL     R0
001064  016016  000064'                 MOV     STORAGE(R0),(SP)
001070  004737  000000G                 JSR     PC,SET.UPAR
001074  105202                          INCB    R2                      ; TBL.COUNT             3638
001076  022626                          CMP     (SP)+,(SP)+             ;                       3636
001100  013700  000000G                 MOV     CUOFF,R0                ;                       3640
001104  006300                          ASL     R0
001106  063700  000000G                 ADD     CST.ADDR,R0
001112  032710  040000                  BIT     #40000,(R0)
001116  001406                          BEQ     23$
001120  032710  020000                  BIT     #20000,(R0)             ;                       3641
001124  001403                          BEQ     23$
001126  032710  010000                  BIT     #10000,(R0)             ;                       3642
001132  001505                          BEQ     30$
001134  120227  000020          23$:    CMPB    R2,#20                  ; TBL.COUNT,*           3643
```

```
001140  001335                          BNE     22$
001142  000501                  24$:    BR      30$                             ;                               3645
001144  105003                  25$:    CLRB    R3                              ; MOD.COUNT                     3656
001146  012701  000003                  MOV     #3,R1                           ; *,OFFSET                      3658
001152  010100                  26$:    MOV     R1,R0                           ; OFFSET,*                      3660
001154  006300                          ASL     R0
001156  063700  000000G                 ADD     CST.ADDR,R0
001162  032710  040000                  BIT     #40000,(R0)
001166  001414                          BEQ     27$
001170  032710  020000                  BIT     #20000,(R0)                     ;                               3661
001174  001411                          BEQ     27$
001176  032710  010000                  BIT     #10000,(R0)                     ;                               3662
001202  001006                          BNE     27$
001204  005000                          CLR     R0                              ;                               3665
001206  150300                          BISB    R3,R0                           ; MOD.COUNT,*
001210  006300                          ASL     R0
001212  010160  000064'                 MOV     R1,STORAGE(R0)                  ; OFFSET,*
001216  105203                          INCB    R3                              ; MOD.COUNT                     3666
001220  062701  000012          27$:    ADD     #12,R1                          ; *,OFFSET                      3658
001224  020127  000041                  CMP     R1,#41                          ; OFFSET,*
001230  003750                          BLE     26$
001232  105703                          TSTB    R3                              ; MOD.COUNT                     3672
001234  001457                          BEQ     31$
001236  105002                          CLRB    R2                              ; TBL.COUNT                     3675
001240  005000                  28$:    CLR     R0                              ;                               3679
001242  150200                          BISB    R2,R0                           ; TBL.COUNT,*
001244  006300                          ASL     R0
001246  016046  000000G                 MOV     RANDOM(R0),-(SP)
001252  042716  100000                  BIC     #100000,(SP)
001256  005046                          CLR     -(SP)
001260  110316                          MOVB    R3,(SP)                         ; MOD.COUNT,*
001262  004737  000000G                 JSR     PC,BL$MOD
001266  006300                          ASL     R0
001270  016016  000064'                 MOV     STORAGE(R0),(SP)
001274  004737  000000G                 JSR     PC,SET.UPAR
001300  105202                          INCB    R2                              ; TBL.COUNT                     3680
001302  022626                          CMP     (SP)+,(SP)+                     ;                               3678
001304  013700  000000G                 MOV     CUOFF,R0                        ;                               3682
001310  006300                          ASL     R0
001312  063700  000000G                 ADD     CST.ADDR,R0
001316  032710  040000                  BIT     #40000,(R0)
001322  001406                          BEQ     29$
001324  032710  020000                  BIT     #20000,(R0)                     ;                               3683
001330  001403                          BEQ     29$
001332  032710  010000                  BIT     #10000,(R0)                     ;                               3684
001336  001403                          BEQ     30$
001340  120227  000020          29$:    CMPB    R2,#20                          ; TBL.COUNT,*                   3685
001344  001335                          BNE     28$
001346  013700  000116'          30$:    MOV     MAD1,R0                         ;                               3687
001352  013760  000000G 000016          MOV     CDISK,16(R0)
001360  013700  000120'                 MOV     MAD2,R0                         ;                               3688
001364  013760  000000G 000016          MOV     CDISK,16(R0)
001372  000207                          RTS     PC                              ;                               3674
```

```
001374  112737  000001  000000G        31$:    MOVB    #1,EOP.FLAG                     ;                       3697
001402  000207                                 RTS     PC                              ;                       3424

; Routine Size:  386 words,     Routine Base:  $CODE$ + 11366
; Maximum stack depth per invocation:  8 words
```

```
;    3700  1     GLOBAL routine QIO_FUNC : novalue =
;    3701  1
;    3702  1     !+
;    3703  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;    3704  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;    3705  1     !       BY THE FOLLWING ALGORITHM:
;    3706  1     !
;    3707  1     !               IF THE CHOSEN UNIT IS PROTECTED
;    3708  1     !               THEN
;    3709  1     !                       FUNCTION = READ
;    3710  1     !               ELSE (UNPROTECTED)
;    3711  1     !                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;    3712  1     !                       NUMBER
;    3713  1     !
;    3714  1     !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
;    3715  1     !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;    3716  1     !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;    3717  1     !       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
;    3718  1     !
;    3719  1     !
;    3720  1     !       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT   ZZZ
;    3721  1     !       BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN     ZZZ
;    3722  1     !       DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED     ZZZ
;    3723  1     !       WITH THE REGULAR MSCP TESTING OF THE LBNS.                       ZZZ
;    3724  1     !
;    3725  1     !       TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY       ZZZ
;    3726  1     !       EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O's HAVE BEEN DONE.  ZZZ
;    3727  1     !       THE DUMBER OF DUP I/O's IS 'DUPROUND'. THIS GIVES US A 25 TO 1   ZZZ
;    3728  1     !       INTERLEAVE.                                                      ZZZ
;    3729  1     !
;    3730  1     !       THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS   ZZZ
;    3731  1     !       TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON-      ZZZ
;    3732  1     !       TROLLER IS REINITIALIZED, AND QIO_FUNC ROUTINE CONTINUES FROM    ZZZ
;    3733  1     !       WHERE IT LEFT OFF.                                               ZZZ
;    3734  1     !
;    3735  1     !
;    3736  1     !       IMPLICIT INPUTS:
;    3737  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3738  1     !               CUOFF - CURRENT UNIT CST OFFSET
;    3739  1     !
;    3740  1     !       IMPLICIT OUTPUTS:
;    3741  1     !               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
;    3742  1     !-
;    3743  1
;    3744  2         begin
;    3745  2
;    3746  2         local
;    3747  2             FUNC : word;                                          ! OPCODE (READ OR WRITE)
;    3748  2
;    3749  2     DUOFF = .CUOFF;                                  !SAVE IN CASE OTHER CMDS ZZZ
;    3750  2                                                      !LEFT IN QUEUE           ZZZ
;    3751  3     IF ((.CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] LEQ 0) AND !MSCP CNT=0       ZZZ
;    3752  3         (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND         !FIXED DISK       ZZZ
```

```
;    3753  3          (.CST_ADDR [.DUOFF + OF_DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN      ZZZ
;    3754  3                                                                              !ZZZ
;    3755  2      THEN                                                                    !ZZZ
;    3756  3          BEGIN                                                               !ZZZ
;    3757  3          PUT_PKT (.MX2);                          !RETURN 2ND ENVLOPE        !ZZZ
;    3758  3          MX2 = -1;                                !INDICATE FAILURE          ZZZ
;    3759  3          DUP ();                                  !DO DUP TEST               ZZZ
;    3760  3          CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =  !REINIT MSCP FUN-          ZZZ
;    3761  3                  (25 * .DUPROUND);                !CTION COUNTER             ZZZ
;    3762  3                                                                              !ZZZ
;    3763  3      !       THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER   ZZZ
;    3764  3      !       CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.             ZZZ
;    3765  3      !                                                                       ZZZ
;    3766  3          DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;      !SET DUP INIT FLAG         ZZZ
;    3767  3          INIT_TEST ();                            !REINIT CONTROLLER         ZZZ
;    3768  3          DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT); !CLR DUP INIT DLAG       ZZZ
;    3769  3                                                                              !ZZZ
;    3770  3          MX2 = -1;                                !ASSUME NO 2ND ENVELOPE    ZZZ
;    3771  3          IF (MX1 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 1ST ENVELOPE      ZZZ
;    3772  4              OR (.EOP_FLAG)                       !IF FAILURE                ZZZ
;    3773  3          THEN    RETURN;                          !NO POINT TO GO ON         ZZZ
;    3774  3          IF (MX2 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 2ND ENVELOPE      ZZZ
;    3775  4              OR (.EOP_FLAG)                       !IF FAILURE                ZZZ
;    3776  4          THEN    BEGIN                                                       !ZZZ
;    3777  4                  PUT_PKT (.MX1);                  !PUT 1ST BACK IN POOL      ZZZ
;    3778  4                  MX1 = -1;                        !INDICATE FAILURE          ZZZ
;    3779  4                  RETURN;                          !DONE                      ZZZ
;    3780  3                  END;                                                        !ZZZ
;    3781  3                                                                              !ZZZ
;    3782  3                  MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2); !CALC START ADDR   ZZZ
;    3783  3                  MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2); !OF BOTH ENVELOPES ZZZ
;    3784  3                  GET_RANDOM ();                   !GET SET OF RANDOM NOS     ZZZ
;    3785  3                  QIO_UNIT ();                     !PUT RAND UNIT NO IN       ZZZ
;    3786  2                  END;                             !ENVELOPES                 ZZZ
;    3787  2                                                                              !ZZZ
;    3788  2      !       MSCP CODE STARTS HERE                                           ZZZ
;    3789  2      !                                                                       ZZZ
;    3790  2
;    3791  2      CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =      !                          ZZZ
;    3792  2          .CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] -1;  !DECR MSCP FUNCTION CNTR ZZZ
;    3793  2
;    3794  2          MAD2 [OPCODE] = 0;                                                  ! ASSUME 2ND PACKET NOT NEEDED
;    3795  2
;    3796  2          if (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED) and       ! IF "FORCED ERROR" SET IN LAST READ,
;    3797  2             (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and             !       REWRITE SAME BLOCK
;    3798  3             (.FORCED_ERROR)                                                  !
;    3799  2          then
;    3800  2              FUNC = OP_WRT
;    3801  2          else
;    3802  2
;    3803  2              if .CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql PROTECTED           ! IF UNIT IS PROTECTED
;    3804  2              then
;    3805  2                  FUNC = OP_RD                                                ! SET FUNCTION TO READ
```

```
;    3806  2              else
;    3807  2
;    3808  3                  if (.RANDOM [1] and 1)                          ! USE 2ND RANDOM NUMBER TO SELECT
;    3809  2                  then
;    3810  2                      FUNC = OP_RD                                ! READ
;    3811  2                  else
;    3812  2                      FUNC = OP_WRT;                              ! WRITE
;    3813  2
;    3814  2
;    3815  2
;    3816  3          IF (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED)      !I/O's ARE CANCELLED WHEN CMD ZZZ
;    3817  2          THEN                                                           !RING IS FULL. DON'T LET THIS ZZZ
;    3818  2              IF .RW_BALANCE GEQU TOO_MANY_READS                         !UPSET THE BALANCE BETWEEN    ZZZ
;    3819  2              THEN                                                       !THE NUMBER OF READS AND      ZZZ
;    3820  2                  FUNC = OP_WRT;                                         !WRITES.                      ZZZ
;    3821  2
;    3822  2
;    3823  2          if (MAD1 [OPCODE] = .FUNC) eql OP_WRT                      ! LOAD CHOSEN OPCODE. IF WRITE
;    3824  2          then
;    3825  3              begin
;    3826  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                         ! NON-SEQUENTIAL COMMAND
;    3827  3
;    3828  4              if BIT_TST (SWP_FLAGS, SWF_CWC)                        ! IF CONTROLLER DOES WRITE-COMPARES
;    3829  3              then
;    3830  3                  MAD1 [MODIFY] = MD_CMP                             ! ADD COMPARE MODIFIER
;    3831  3              else
;    3832  3
;    3833  4                  if BIT_TST (SWP_FLAGS, SWF_HWC)                    ! IF HOST DOES WRITE-COMPARES
;    3834  3                  then
;    3835  4                      begin
;    3836  4                      MAD1 [MODIFY] = MD_EXP;                        ! SET WRITE AS AN EXPRESS REQUEST
;    3837  4                      MAD2 [OPCODE] = OP_RD;                         ! SET READ OPCODE INTO 2ND MSCP PACKET
;    3838  4                      MAD2 [MODIFY] = MD_EXP;                        ! SET READ AS AN EXPRESS REQUEST TOO
;    3839  4                      MAD2 [CMD_TYPE] = NON_SEQ_CMD;                 ! NON-SEQUENTIAL COMMAND
;    3840  3                      end;
;    3841  3                  end
;    3842  2              else
;    3843  3                  begin
;    3844  3                  MAD1 [CMD_TYPE] = NON_SEQ_CMD;                     ! NON-SEQUENTIAL COMMAND
;    3845  3
;    3846  4                  if BIT_TST (SWP_FLAGS, SWF_CRC)                    ! IF READ-COMPARES - FUNCTION IS READ
;    3847  3                  then
;    3848  3                      MAD1 [MODIFY] = MD_CMP;                        ! ADD COMPARE MODIFIER
;    3849  3
;    3850  2                  end;
;    3851  2
;    3852  2          if .MAD2 [OPCODE] eql 0                                    ! IF NO OPCODE IN 2ND PACKET
;    3853  2          then
;    3854  3              begin
;    3855  3              PUT_PKT (.MX2);                                        ! RETURN 2ND PACKET TO POOL
;    3856  3              MX2 = -1;                                              ! MARK IT UNUSED
;    3857  2              end;
;    3858  2
```

```
;   3859  1          end;                                                    ! ROUTINE QIO_FUNC


                                        .SBTTL  QIO.FUNC MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 QIO.FUNC::
                                        JSR     R1,$SAVE4                       ;                       3700
000004  013737  000000G 001162'                 MOV     CUOFF,DUOFF                     ;                       3749
000012  013702  000000G                         MOV     CST.ADDR,R2                     ;                       3751
000016  013701  001162'                         MOV     DUOFF,R1
000022  010100                                  MOV     R1,R0
000024  006300                                  ASL     R0
000026  060200                                  ADD     R2,R0
000030  005760  000022                          TST     22(R0)
000034  003146                                  BGT     4$
000036  010100                                  MOV     R1,R0                           ;                       3752
000040  006300                                  ASL     R0
000042  060200                                  ADD     R2,R0
000044  132710  000020                          BITB    #20,(R0)
000050  001540                                  BEQ     4$
000052  010100                                  MOV     R1,R0                           ;                       3753
000054  006300                                  ASL     R0
000056  060200                                  ADD     R2,R0
000060  005760  000020                          TST     20(R0)
000064  100532                                  BMI     4$
000066  013746  000114'                         MOV     MX2,-(SP)                       ;                       3757
000072  004737  000000G                         JSR     PC,PUT.PKT
000076  012737  177777  000114'                 MOV     #-1,MX2                         ;                       3758
000104  004737  000000V                         JSR     PC,DUP                          ;                       3759
000110  013701  001162'                         MOV     DUOFF,R1                        ;                       3760
000114  006301                                  ASL     R1
000116  063701  000000G                         ADD     CST.ADDR,R1
000122  013716  000000G                         MOV     DUPROUND,(SP)                   ;                       3761
000126  012746  000031                          MOV     #31,-(SP)
000132  004737  000000G                         JSR     PC,BL$MUL
000136  010061  000022                          MOV     R0,22(R1)
000142  052737  000002  000000G                 BIS     #2,DUP.FLAGS                    ;                       3766
000150  004737  001140'                         JSR     PC,INIT.TEST                    ;                       3767
000154  042737  000002  000000G                 BIC     #2,DUP.FLAGS                    ;                       3768
000162  012737  177777  000114'                 MOV     #-1,MX2                         ;                       3770
000170  013716  000000G                         MOV     CCTLR,(SP)                      ;                       3771
000174  004737  000000G                         JSR     PC,GET.PKT
000200  010037  000112'                         MOV     R0,MX1
000204  002426                                  BLT     2$
000206  132737  000001  000000G                 BITB    #1,EOP.FLAG                     ;                       3772
000214  001022                                  BNE     2$                              ;                       3700
000216  013716  000000G                         MOV     CCTLR,(SP)                      ;                       3774
000222  004737  000000G                         JSR     PC,GET.PKT
000226  010037  000114'                         MOV     R0,MX2
000232  002404                                  BLT     1$
000234  132737  000001  000000G                 BITB    #1,EOP.FLAG                     ;                       3775
000242  001411                                  BEQ     3$
000244  013716  000112'                 1$:     MOV     MX1,(SP)                        ;                       3777
000250  004737  000000G                         JSR     PC,PUT.PKT
```

```
000254  012737  177777  000112'                MOV     #-1,MX1                 ;                       3778
000262  022626                          2$:     CMP     (SP)+,(SP)+             ;                       3779
000264  000207                                  RTS     PC                      ;                       3776
000266  013716  000112'                 3$:     MOV     MX1,(SP)                ;                       3782
000272  012746  000106                          MOV     #106,-(SP)
000276  004737  000000G                         JSR     PC,BL$MUL
000302  062700  000000G                         ADD     #MSCP.PKT,R0
000306  010037  000116'                         MOV     R0,MAD1
000312  013716  000114'                         MOV     MX2,(SP)                ;                       3783
000316  012746  000106                          MOV     #106,-(SP)
000322  004737  000000G                         JSR     PC,BL$MUL
000326  062700  000000G                         ADD     #MSCP.PKT,R0
000332  010037  000120'                         MOV     R0,MAD2
000336  004737  011256'                         JSR     PC,GET.RANDOM           ;                       3784
000342  004737  011366'                         JSR     PC,QIO.UNIT             ;                       3785
000346  062706  000010                          ADD     #10,SP                  ;                       3756
000352  013700  000000G                 4$:     MOV     CUOFF,R0                ;                       3791
000356  006300                                  ASL     R0
000360  063700  000000G                         ADD     CST.ADDR,R0
000364  005360  000022                          DEC     22(R0)                  ;                       3792
000370  013701  000120'                         MOV     MAD2,R1                 ;                       3794
000374  012704  000022                          MOV     #22,R4
000400  060104                                  ADD     R1,R4
000402  105014                                  CLRB    (R4)
000404  013700  000000G                         MOV     CUOFF,R0                ;                       3796
000410  006300                                  ASL     R0
000412  063700  000000G                         ADD     CST.ADDR,R0
000416  005003                                  CLR     R3
000420  005710                                  TST     (R0)
000422  100010                                  BPL     5$
000424  005203                                  INC     R3
000426  132710  000020                          BITB    #20,(R0)                ;                       3797
000432  001404                                  BEQ     5$
000434  132737  000001  000000G                 BITB    #1,FORCED.ERROR         ;                       3798
000442  001012                                  BNE     7$                      ;                       3800
000444  032710  100000                  5$:     BIT     #100000,(R0)            ;                       3803
000450  001404                                  BEQ     6$                      ;                       3805
000452  032737  000001  000002G                 BIT     #1,RANDOM+2             ;                       3808
000460  001403                                  BEQ     7$
000462  012702  000041                  6$:     MOV     #41,R2                  ; *,FUNC                3810
000466  000402                                  BR      8$                      ;                       3808
000470  012702  000042                  7$:     MOV     #42,R2                  ; *,FUNC                3812
000474  006003                          8$:     ROR     R3                      ;                       3816
000476  103006                                  BCC     9$
000500  023727  000106' 000002                  CMP     RW.BALANCE,#2           ;                       3818
000506  103402                                  BLO     9$
000510  012702  000042                          MOV     #42,R2                  ; *,FUNC                3820
000514  013700  000116'                 9$:     MOV     MAD1,R0                 ;                       3823
000520  013703  000000G                         MOV     SWP.FLAGS,R3            ;                       3828
000524  110260  000022                          MOVB    R2,22(R0)               ; FUNC,*                3823
000530  020227  000042                          CMP     R2,#42                  ; FUNC,*
000534  001025                                  BNE     10$
000536  112760  000002  000004                  MOVB    #2,4(R0)                ;                       3826
```

```
000544  032703  000020                          BIT     #20,R3                          ;       3828
000550  001025                                  BNE     11$                             ;       3830
000552  032703  000040                          BIT     #40,R3                          ;       3833
000556  001425                                  BEQ     12$
000560  012760  100000  000024                  MOV     #-100000,24(R0)                 ;       3836
000566  112714  000041                          MOVB    #41,(R4)                        ;       3837
000572  012761  100000  000024                  MOV     #-100000,24(R1)                 ;       3838
000600  112761  000002  000004                  MOVB    #2,4(R1)                        ;       3839
000606  000411                                  BR      12$                             ;       3823
000610  112760  000002  000004          10$:    MOVB    #2,4(R0)                        ;       3844
000616  032703  000004                          BIT     #4,R3                           ;       3846
000622  001403                                  BEQ     12$
000624  012760  040000  000024          11$:    MOV     #40000,24(R0)                   ;       3848
000632  105714                          12$:    TSTB    (R4)                            ;       3852
000634  001010                                  BNE     13$
000636  013746  000114'                          MOV     MX2,-(SP)                       ;       3855
000642  004737  000000G                          JSR     PC,PUT.PKT
000646  012737  177777  000114'                  MOV     #-1,MX2                         ;       3856
000654  005726                                  TST     (SP)+                           ;       3854
000656  000207                          13$:    RTS     PC                              ;       3700

; Routine Size:  216 words,     Routine Base:  $CODE$ + 12772
; Maximum stack depth per invocation:  10 words
```

```
;    3860  1
;    3861  1     GLOBAL ROUTINE DUP : NOVALUE =                                          !ZZZ
;    3862  1     !*                                                                     !ZZZ
;    3863  1     !       THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS. ZZZ
;    3864  1     !       THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,   ZZZ
;    3865  1     !       SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                 ZZZ
;    3866  1     !                                                                       ZZZ
;    3867  1     !       THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ-  ZZZ
;    3868  1     !       COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL      ZZZ
;    3869  1     !       RECORD THE STATISTICS IN THE TALLY TABLES.                       ZZZ
;    3870  1     !                                                                       ZZZ
;    3871  1     !       THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ     ZZZ
;    3872  1     !       DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE       ZZZ
;    3873  1     !       SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.      ZZZ
;    3874  1     !       THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN       ZZZ
;    3875  1     !       BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR ZZZ
;    3876  1     !       SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE  ZZZ
;    3877  1     !       PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE ZZZ
;    3878  1     !       DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST- ZZZ
;    3879  1     !       ED IN THE RCT TABLES.                                            ZZZ
;    3880  1     !                                                                       ZZZ
;    3881  1     !       AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES  ZZZ
;    3882  1     !       WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE     ZZZ
;    3883  1     !       AS BEFORE.                                                       ZZZ
;    3884  1     !                                                                       ZZZ
;    3885  1     !       IMPLICIT INPUTS:                                                ZZZ
;    3886  1     !                       CCTLR - CURRENT CONTROLLER NUMBER               ZZZ
;    3887  1     !                       CST_ADDRS - CONTAINS THE CURRENT CONTROLLER     ZZZ
;    3888  1     !                                   STATUS TABLE                        ZZZ
;    3889  1     !                       CUOFF - CURRENT OFFSET IN CST TABLE FOR         ZZZ
;    3890  1     !                               PARTICULAR DRIVE                        ZZZ
;    3891  1     !                                                                       ZZZ
;    3892  1     !               IMPLICIT OUTPUTS:                                       ZZZ
;    3893  1     !                       S_PATTERN - PATTERN BEING WRITTEN TO DBNS       ZZZ
;    3894  1     !                                                                       ZZZ
;    3895  1     !                                                                      !ZZZ
;    3896  1     !-                                                                     !ZZZ
;    3897  1
```

```
;    3898  1
;    3899  1
;    3900  2     BEGIN
;    3901  2     OWN
;    3902  2             TEMP : WORD;
;    3903  2
;    3904  2     !PRINTX (DBM110);
;    3905  2     !PRINTX (DER10);
;    3906  2
;    3907  2     until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    3908  2           (.EOP_FLAG eql true) do                   ! Make sure all MSCP commands are completed
;    3909  3         begin
;    3910  3         BREAK;                                      ! BREAK FOR ACT
;    3911  3         PROC_RETPKT();                              ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    3912  3         RP_INDX = .RP_INDX + 1;                     ! INCREMENT RP_INDX
;    3913  3         if .RP_INDX geq RP_CNT then (RP_INDX = 0;); ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
;    3914  3         RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;    3915  2         end;
;    3916  2
;    3917  2
;    3918  2     S_PATTERN = .RANDOM [1];                                !OTHER UNIT VARIABLES
;    3919  2
;    3920  2     IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .duproun) GEQ 144     ! TEST TO SEE IF NEXT DBN'S TO LARGE
;    3921  2     THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);           ! CIRCLE ARROUND IF DBN TO LARGE
;    3922  2
;    3923  2     DUPIDLE ();                                             ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
;    3924  2     IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN
;    3925  2
;    3926  2     TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
;    3927  2     INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .duproun) DO                   ! INCREMENT FROM RELATIVE DBN TO DBN + dupro
und
;    3928  3             BEGIN
;    3929  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPWRITE]                            ! IF WRITE FLAG SET IN CST TABLE THE
N WRITE DBN'S
;    3930  3             THEN
;    3931  4                     BEGIN
;    3932  4                     DUPIDLE ();                                     ! MAKE SURE THE CONTROLLER IS IN AN IDLE STA
TE
;    3933  4                     DUPWRTDBN ();                                   ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
;    3934  3                     END;
;    3935  3
;    3936  3             DUPIDLE ();                                             ! MAKE SURE CONTROLLER IN IDLE STATE
;    3937  3             DUPREDDBN ();                                           ! CALL ROUTINE TO HANDLE READING DBN'S
;    3938  3
;    3939  3             CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
;    3940  3
;    3941  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1                  ! ERROR IN DUP REINITIALIZE
;    3942  3             THEN RETURN;                                            ! AND RETURN
;    3943  2             END;
;    3944  1     END;


001210                                  .PSECT  $GGG$,  RO
001210                          TEMP:   .BLKW   1
```

```
                                        .SBTTL  DUP MULTI-DRIVE TEST ROUTINES
013652                                  .PSECT  $CODE$,  RO

000000  004137  000000G                 DUP::   JSR     R1,$SAVE3                       ;                       3861
000004  013700  000000G                 1$:     MOV     RP.ADDR,R0                      ;                       3907
000010  023760  000000G 000004                  CMP     CRN.LOW,4(R0)
000016  001433                                  BEQ     3$
000020  123727  000000G 000001                  CMPB    EOP.FLAG,#1                     ;                       3908
000026  001427                                  BEQ     3$
000030  104422                                  TRAP    22                              ;                       3909
000032  004737  000000V                         JSR     PC,PROC.RETPKT                  ;                       3911
000036  005237  000000G                         INC     RP.INDX                         ;                       3912
000042  023727  000000G 000010                  CMP     RP.INDX,#10                     ;                       3913
000050  002402                                  BLT     2$
000052  005037  000000G                         CLR     RP.INDX
000056  013746  000000G                 2$:     MOV     RP.INDX,-(SP)                   ;                       3914
000062  012746  000054                          MOV     #54,-(SP)
000066  004737  000000G                         JSR     PC,BL$MUL
000072  062700  000000G                         ADD     #RETPKT,R0
000076  010037  000000G                         MOV     R0,RP.ADDR
000102  022626                                  CMP     (SP)+,(SP)+                     ;                       3909
000104  000737                                  BR      1$                              ;                       3907
000106  013737  000002G 000000G         3$:     MOV     RANDOM+2,S.PATTERN              ;                       3918
000114  013700  001162'                         MOV     DUOFF,R0                        ;                       3920
000120  006300                                  ASL     R0
000122  063700  000000G                         ADD     CST.ADDR,R0
000126  005001                                  CLR     R1
000130  156001  000020                          BISB    20(R0),R1
000134  063701  000000G                         ADD     DUPROUND,R1
000140  020127  000220                          CMP     R1,#220
000144  002402                                  BLT     4$
000146  105060  000020                          CLRB    20(R0)                          ;                       3921
000152  004737  000000V                 4$:     JSR     PC,DUPIDLE                      ;                       3923
000156  013700  001162'                         MOV     DUOFF,R0                        ;                       3924
000162  006300                                  ASL     R0
000164  063700  000000G                         ADD     CST.ADDR,R0
000170  005760  000020                          TST     20(R0)
000174  100456                                  BMI     8$
000176  116037  000020  001210'                 MOVB    20(R0),TEMP                     ;                       3926
000204  105037  001211'                         CLRB    TEMP+1
000210  013703  001210'                         MOV     TEMP,R3                         ;                       3927
000214  063703  000000G                         ADD     DUPROUND,R3
000220  013700  001162'                         MOV     DUOFF,R0                        ;                       3929
000224  006300                                  ASL     R0
000226  063700  000000G                         ADD     CST.ADDR,R0
000232  010001                                  MOV     R0,R1
000234  062701  000020                          ADD     #20,R1
000240  013702  001210'                         MOV     TEMP,R2                         ; *,DBNCNT              3927
000244  000427                                  BR      7$
000246  032711  010000                  5$:     BIT     #10000,(R1)                     ;                       3929
000252  001404                                  BEQ     6$
000254  004737  000000V                         JSR     PC,DUPIDLE                      ;                       3932
```

```
000260  004737  000000V                    JSR     PC,DUPWRTDBN                    ;                       3933
000264  004737  000000V            6$:     JSR     PC,DUPIDLE                      ;                       3936
000270  004737  000000V                    JSR     PC,DUPREDDBN                    ;                       3937
000274  013700  001162'                    MOV     DUOFF,R0                        ;                       3939
000300  006300                            ASL     R0
000302  063700  000000G                    ADD     CST.ADDR,R0
000306  010001                            MOV     R0,R1
000310  062701  000020                     ADD     #20,R1
000314  105211                            INCB    (R1)
000316  032711  040000                     BIT     #40000,(R1)                     ;                       3941
000322  001003                            BNE     8$                              ;                       3942
000324  005202            7$:             INC     R2                              ; DBNCNT                3927
000326  020203                            CMP     R2,R3                           ; DBNCNT,*
000330  003746                            BLE     5$
000332  000207            8$:             RTS     PC                              ;                       3861

; Routine Size:  110 words,    Routine Base:  $CODE$ + 13652
; Maximum stack depth per invocation:  7 words


;   3945  1
```

```
;    3946  1      GLOBAL ROUTINE DUPWRTDBN : NOVALUE =
;    3947  1
;    3948  1      !+
;    3949  1      ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;    3950  1      ! "WRTDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;    3951  1      ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM WRITES TO A DIAGONOSTIC BLOCK (DBN)
;    3952  1      ! THE WORD IN "S_PATTERN" IS WRITTEN TO THE 256 WORDS IN THE DBN.  IF AN ERROR OCCURS
;    3953  1      ! WHILE RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;    3954  1      ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;    3955  1      !
;    3956  1      !       IMPLICIT INPUTS:
;    3957  1      !               CST_ADDRS  - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;    3958  1      !               DUOFF      - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;    3959  1      !               S_PATTERN  - CONTAINS PATTERN WORD!-
;    3960  2      BEGIN
;    3961  2      !PRINTX (DER11);
;    3962  2      T_ADDR [T_DBN_WT] = .T_ADDR [T_DBN_WT] + 1;     ! INCREMENT # OF WRITES GIVEN
;    3963  2
;    3964  2      MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;               ! PACKET SIZE                                   EXECUTE LOCAL PROGRAM WRT DB
N
;    3965  2      MSCP_PKT [.MX1, OPCODE] = OP_ELP;               ! OPCODE = EXECUTE LOCAL PROGRAM
;    3966  2      MSCP_PKT [.MX1, L1] = %ascii'W';                ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;    3967  2      MSCP_PKT [.MX1, L2] = %ascii'R';
;    3968  2      MSCP_PKT [.MX1, L3] = %ascii'T';
;    3969  2      MSCP_PKT [.MX1, L4] = %ascii'D';
;    3970  2      MSCP_PKT [.MX1, L5] = %ascii'B';
;    3971  2      MSCP_PKT [.MX1, L6] = %ascii'N';
;    3972  2      MSCP_PKT [.MX1, MODIFY] = 1;                    ! STANDALONE MODIFIER
;    3973  2      MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;              ! CALL IT IMMEDIATE
;    3974  2      DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    3975  2
;    3976  2      IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  !status error
;    3977  2      THEN RETURN;                                    ! AND RETURN
;    3978  2
;    3979  3      DO (MX1 = GET_PKT (.CCTLR))
;    3980  2      UNTIL (.MX1 GEQ 0);                     ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    3981  2
;    3982  2              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                   RECIEVE DATA
;    3983  2              MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
;    3984  2              MSCP_PKT [.MX1, BC_LO] = 80;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 2 ***see pg 26 of DUP s
pec
;    3985  2              MSCP_PKT [.MX1, BUF_O] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;    3986  2              MSCP_PKT [.MX1, MODIFY] = 0;                    !
;    3987  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;    3988  2              DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    3989  2
;    3990  2              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;    3991  2                 (.DUPPKT [DUPTYPE] NEQU 1) OR                !dup type error
;    3992  3                 (.DUPPKT [DUPMESG] NEQU 6)
;    3993  2              THEN
;    3994  3                      (HARD_ERROR ();
;    3995  3                       CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    3996  2                       RETURN;);                              ! NO POINT IN CONTINUING
;    3997  2
;    3998  3              DO (MX1 = GET_PKT (.CCTLR))
```

```
;    3999  2              UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4000  2
;    4001  2              MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;            ! PACKET SIZE                              SEND DATA
;    4002  2              MSCP_PKT [.MX1, OPCODE] = OP_SDD;            ! OPCODE = SEND DATA
;    4003  2              MSCP_PKT [.MX1, BC_LO] = 6;                  ! BYTE COUNT TO BE TRANSFERED EQUALS 6
;    4004  2              MSCP_PKT [.MX1, BUF_0] = DUPPKT;             ! LOAD DESCRIBTOR BUFFER
;    4005  2              DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; !LOAD UNIT NUMBER (RDRX)
;    4006  2              DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;    4007  2              DUPPKT [DUPBF2] = .S_PATTERN;                ! LOAD PATTERN
;    4008  2              MSCP_PKT [.MX1, MODIFY] = 0;                 !
;    4009  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;           ! CALL IT sequential
;    4010  2              DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
;    4011  2
;    4012  2              IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  ! status error
;    4013  2              THEN RETURN;
;    4014  2
;    4015  3              DO (MX1 = GET_PKT (.CCTLR))
;    4016  2              UNTIL (.MX1 GEQ 0);                  ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4017  2
;    4018  2              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;            ! PACKET SIZE                              RECEIVE DATA
;    4019  2              MSCP_PKT [.MX1, OPCODE] = OP_RCD;            ! OPCODE = RECEIVE DATA
;    4020  2              MSCP_PKT [.MX1, BC_LO] = 4;                  ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;    4021  2              MSCP_PKT [.MX1, BUF_0] = DUPPKT;             ! LOAD DESCRIBTOR BUFFER
;    4022  2              MSCP_PKT [.MX1, MODIFY] = 0;                 !
;    4023  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;           ! CALL IT sequential
;    4024  2              DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
;    4025  2
;    4026  2              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) or     !status error
;    4027  2                 (.DUPPKT [DUPTYPE] NEQU 3) or                !dup type error
;    4028  2                 (.DUPPKT [DUPMESG] NEQU 3) or
;    4029  3                 (.DUPPKT [DUPBF1] NEQU 0)                    !non successful write code
;    4030  2              THEN
;    4031  3                      (HARD_ERROR ();
;    4032  3                       CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    4033  2                       RETURN;);                              ! NO POINT IN CONTINUING
;    4034  2
;    4035  3         DO (MX1 = GET_PKT (.CCTLR))
;    4036  2         UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE.
;    4037  2
;    4038  2         T_ADDR [T_BLK_WT] = .T_ADDR [T_BLK_WT] + 1;      !INCREMENT COUNTER IF A SUCCESS
;    4039  2
;    4040  1         END;


                                        .SBTTL  DUPWRTDBN MULTI-DRIVE TEST ROUTINES
000000  010146                     DUPWRTDBN::
                                        MOV     R1,-(SP)                        ;                                  3946
000002  013700  000000G                 MOV     T.ADDR,R0                       ;                                  3962
000006  005260  000060                  INC     60(R0)
000012  013746  000112'                 MOV     MX1,-(SP)                       ;                                  3964
000016  012746  000106                  MOV     #106,-(SP)
000022  004737  000000G                 JSR     PC,BL$MUL
000026  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
```

```
000034  112760  000003  000022G                  MOVB    #3,MSCP.PKT+22(R0)              ;                       3965
000042  112760  000127  000026G                  MOVB    #127,MSCP.PKT+26(R0)            ;                       3966
000050  112760  000122  000027G                  MOVB    #122,MSCP.PKT+27(R0)            ;                       3967
000056  112760  000124  000030G                  MOVB    #124,MSCP.PKT+30(R0)            ;                       3968
000064  112760  000104  000031G                  MOVB    #104,MSCP.PKT+31(R0)            ;                       3969
000072  112760  000102  000032G                  MOVB    #102,MSCP.PKT+32(R0)            ;                       3970
000100  112760  000116  000033G                  MOVB    #116,MSCP.PKT+33(R0)            ;                       3971
000106  012760  000001  000024G                  MOV     #1,MSCP.PKT+24(R0)              ;                       3972
000114  142760  000360  000010G                  BICB    #360,MSCP.PKT+10(R0)            ;                       3973
000122  004737  000000V                          JSR     PC,DUPCOMMAND                   ;                       3974
000126  013700  001162'                          MOV     DUOFF,R0                        ;                       3976
000132  006300                                   ASL     R0
000134  063700  000000G                          ADD     CST.ADDR,R0
000140  032760  040000  000020                   BIT     #40000,20(R0)
000146  001402                                   BEQ     1$
000150  022626                                   CMP     (SP)+,(SP)+                     ;                       3946
000152  000505                                   BR      3$                              ;                       3977
000154  013716  000000G                  1$:     MOV     CCTLR,(SP)                      ;                       3979
000160  004737  000000G                          JSR     PC,GET.PKT
000164  010037  000112'                          MOV     R0,MX1
000170  002771                                   BLT     1$                              ;                       3980
000172  010016                                   MOV     R0,(SP)                         ;                       3982
000174  012746  000106                           MOV     #106,-(SP)
000200  004737  000000G                          JSR     PC,BL$MUL
000204  012760  000034  000006G                  MOV     #34,MSCP.PKT+6(R0)
000212  112760  000005  000022G                  MOVB    #5,MSCP.PKT+22(R0)              ;                       3983
000220  012760  000120  000026G                  MOV     #120,MSCP.PKT+26(R0)            ;                       3984
000226  012760  000000G 000032G                  MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       3985
000234  005060  000024G                          CLR     MSCP.PKT+24(R0)                 ;                       3986
000240  142760  000360  000010G                  BICB    #360,MSCP.PKT+10(R0)            ;                       3987
000246  152760  000020  000010G                  BISB    #20,MSCP.PKT+10(R0)
000254  004737  000000V                          JSR     PC,DUPCOMMAND                   ;                       3988
000260  013700  001162'                          MOV     DUOFF,R0                        ;                       3990
000264  006300                                   ASL     R0
000266  063700  000000G                          ADD     CST.ADDR,R0
000272  032760  040000  000020                   BIT     #40000,20(R0)
000300  001016                                   BNE     2$
000302  013700  000000G                          MOV     DUPPKT,R0                       ;                       3991
000306  042700  007777                           BIC     #7777,R0
000312  020027  010000                           CMP     R0,#10000
000316  001007                                   BNE     2$
000320  013700  000000G                          MOV     DUPPKT,R0                       ;                       3992
000324  042700  170000                           BIC     #170000,R0
000330  020027  000006                           CMP     R0,#6
000334  001415                                   BEQ     4$
000336  004737  000000V                  2$:     JSR     PC,HARD.ERROR                   ;                       3994
000342  013700  001162'                          MOV     DUOFF,R0                        ;                       3995
000346  006300                                   ASL     R0
000350  063700  000000G                          ADD     CST.ADDR,R0
000354  052760  040000  000020                   BIS     #40000,20(R0)
000362  062706  000006                           ADD     #6,SP                           ;                       3996
000366  000504                          3$:     BR      5$                              ;                       3994
000370  013716  000000G                  4$:     MOV     CCTLR,(SP)                      ;                       3998
```

```
000374  004737  000000G                        JSR     PC,GET.PKT
000400  010037  000112'                        MOV     R0,MX1
000404  002771                                 BLT     4$                                      ;3999
000406  010016                                 MOV     R0,(SP)                                 ;4001
000410  012746  000106                         MOV     #106,-(SP)
000414  004737  000000G                        JSR     PC,BL$MUL
000420  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000426  112760  000004  000022G                MOVB    #4,MSCP.PKT+22(R0)                      ;4002
000434  012760  000006  000026G                MOV     #6,MSCP.PKT+26(R0)                      ;4003
000442  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)                 ;4004
000450  013701  001162'                        MOV     DUOFF,R1                                ;4005
000454  006301                                 ASL     R1
000456  063701  000000G                        ADD     CST.ADDR,R1
000462  111137  000000G                        MOVB    (R1),DUPPKT
000466  042737  177760  000000G               BIC     #177760,DUPPKT
000474  013701  001162'                        MOV     DUOFF,R1                                ;4006
000500  006301                                 ASL     R1
000502  063701  000000G                        ADD     CST.ADDR,R1
000506  116137  000020  000002G                MOVB    20(R1),DUPPKT+2
000514  105037  000003G                        CLRB    DUPPKT+3
000520  013737  000000G 000004G                MOV     S.PATTERN,DUPPKT+4                      ;4007
000526  005060  000024G                        CLR     MSCP.PKT+24(R0)                         ;4008
000532  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)                    ;4009
000540  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000546  004737  000000V                        JSR     PC,DUPCOMMAND                           ;4010
000552  013700  001162'                        MOV     DUOFF,R0                                ;4012
000556  006300                                 ASL     R0
000560  063700  000000G                        ADD     CST.ADDR,R0
000564  032760  040000  000020                BIT     #40000,20(R0)
000572  001403                                 BEQ     6$
000574  062706  000010                         ADD     #10,SP                                  ;3946
000600  000524                          5$:    BR      10$                                     ;4013
000602  013716  000000G                 6$:    MOV     CCTLR,(SP)                              ;4015
000606  004737  000000G                        JSR     PC,GET.PKT
000612  010037  000112'                        MOV     R0,MX1
000616  002771                                 BLT     6$                                      ;4016
000620  010016                                 MOV     R0,(SP)                                 ;4018
000622  012746  000106                         MOV     #106,-(SP)
000626  004737  000000G                        JSR     PC,BL$MUL
000632  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000640  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)                      ;4019
000646  012760  000004  000026G                MOV     #4,MSCP.PKT+26(R0)                      ;4020
000654  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)                 ;4021
000662  005060  000024G                        CLR     MSCP.PKT+24(R0)                         ;4022
000666  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)                    ;4023
000674  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000702  004737  000000V                        JSR     PC,DUPCOMMAND                           ;4024
000706  013700  001162'                        MOV     DUOFF,R0                                ;4026
000712  006300                                 ASL     R0
000714  063700  000000G                        ADD     CST.ADDR,R0
000720  032760  040000  000020                BIT     #40000,20(R0)
000726  001021                                 BNE     7$
000730  013700  000000G                        MOV     DUPPKT,R0                               ;4027
```

```
000734  042700  007777                          BIC     #7777,R0
000740  020027  030000                          CMP     R0,#30000
000744  001012                                  BNE     7$
000746  013700  000000G                         MOV     DUPPKT,R0       ;                       4028
000752  042700  170000                          BIC     #170000,R0
000756  020027  000003                          CMP     R0,#3
000762  001003                                  BNE     7$
000764  005737  000002G                         TST     DUPPKT+2        ;                       4029
000770  001413                                  BEQ     8$
000772  004737  000000V                 7$:     JSR     PC,HARD.ERROR   ;                       4031
000776  013700  001162'                         MOV     DUOFF,R0        ;                       4032
001002  006300                                  ASL     R0
001004  063700  000000G                         ADD     CST.ADDR,R0
001010  052760  040000  000020                  BIS     #40000,20(R0)
001016  000413                                  BR      9$              ;                       4033
001020  013716  000000G                 8$:     MOV     CCTLR,(SP)      ;                       4035
001024  004737  000000G                         JSR     PC,GET.PKT
001030  010037  000112'                         MOV     R0,MX1
001034  002771                                  BLT     8$              ;                       4036
001036  013700  000000G                         MOV     T.ADDR,R0       ;                       4038
001042  005260  000056                          INC     56(R0)
001046  062706  000012                 9$:     ADD     #12,SP          ;                       3960
001052  012601                         10$:    MOV     (SP)+,R1        ;                       3946
001054  000207                                  RTS     PC

; Routine Size:  279 words,     Routine Base:  $CODE$ + 14206
; Maximum stack depth per invocation:  7 words
```

```
;    4041  1     GLOBAL ROUTINE DUPREDDBN : NOVALUE =
;    4042  1
;    4043  1     !+
;    4044  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;    4045  1     ! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;    4046  1     ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
;    4047  1     ! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
;    4048  1     ! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;    4049  1     ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;    4050  1     !
;    4051  1     !
;    4052  1     !       IMPLICIT INPUTS:
;    4053  1     !               CST_ADDR    - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;    4054  1     !               DUOFF       - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;    4055  1     !
;    4056  2     BEGIN
;    4057  2     !PRINTX (DER12);
;    4058  2     T_ADDR [T_DBN_RD] = .T_ADDR [T_DBN_RD] + 1;     ! INCREMENT # OF READS GIVEN
;    4059  2
;    4060  2     MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;               ! PACKET SIZE                    EXECUTE REDDBN PROGRAM
;    4061  2     MSCP_PKT [.MX1, OPCODE] = OP_ELP;               ! OPCODE = EXECUTE LOCAL PROGRAM
;    4062  2     MSCP_PKT [.MX1, L1] = %ascii'R';                ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;    4063  2     MSCP_PKT [.MX1, L2] = %ascii'E';
;    4064  2     MSCP_PKT [.MX1, L3] = %ascii'D';
;    4065  2     MSCP_PKT [.MX1, L4] = %ascii'D';
;    4066  2     MSCP_PKT [.MX1, L5] = %ascii'B';
;    4067  2     MSCP_PKT [.MX1, L6] = %ascii'N';
;    4068  2     MSCP_PKT [.MX1, MODIFY] = 1;                    ! STANDALONE MODIFIER
;    4069  2     MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;              ! CALL IT IMMEDIATE
;    4070  2     DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    4071  2
;    4072  2     IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  !status error
;    4073  2     THEN RETURN;
;    4074  2
;    4075  3     DO (MX1 = GET_PKT (.CCTLR))
;    4076  2     UNTIL (.MX1 GEQ 0);                     ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4077  2
;    4078  2             MSCP_PKT [.MX1, MSGLEN] = SZ_REC;           ! PACKET SIZE                    RECIEVE DATA
;    4079  2             MSCP_PKT [.MX1, OPCODE] = OP_RCD;           ! OPCODE = RECEIVE DATA
;    4080  2             MSCP_PKT [.MX1, BC_LO] = 80;                ! BYTE COUNT TO BE TRANSFERED EQUALS 2 *****see pg 26 DUP sp
ec
;    4081  2             MSCP_PKT [.MX1, BUF_O] = DUPPKT;            ! LOAD DESCRIBTOR BUFFER
;    4082  2             MSCP_PKT [.MX1, MODIFY] = 0;                !
;    4083  2             MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;          ! CALL IT sequential
;    4084  2             DUPCOMMAND ();                              ! SENDS AND RECEIVES THE COMMAND
;    4085  2
;    4086  2             IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;    4087  2                (.DUPPKT [DUPTYPE] NEQU 1) OR                !dup type error
;    4088  3                (.DUPPKT [DUPMESG] NEQU 5)
;    4089  2             THEN
;    4090  3                     (HARD_ERROR ();
;    4091  3                      CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    4092  2                      RETURN;);                                      ! NO POINT IN CONTINUING
;    4093  2
```

```
;    4094  3          DO (MX1 = GET_PKT (.CCTLR))
;    4095  2          UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4096  2
;    4097  2          MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;            ! PACKET SIZE                              SEND DATA
;    4098  2          MSCP_PKT [.MX1, OPCODE] = OP_SDD;            ! OPCODE = SEND DATA
;    4099  2          MSCP_PKT [.MX1, BC_LO] = 4;                  ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;    4100  2          MSCP_PKT [.MX1, BUF_0] = DUPPKT;             ! LOAD DESCRIBTOR BUFFER
;    4101  2          DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; ! LOAD UNIT NUMBER (RDRX)
;    4102  2          DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;    4103  2          MSCP_PKT [.MX1, MODIFY] = 0;                 !
;    4104  2          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;           ! CALL IT sequential
;    4105  2          DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
;    4106  2
;    4107  2          IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  !status error
;    4108  2          THEN RETURN;
;    4109  2
;    4110  3          DO (MX1 = GET_PKT (.CCTLR))
;    4111  2          UNTIL (.MX1 GEQ 0);                      ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4112  2
;    4113  2          MSCP_PKT [.MX1, MSGLEN] = SZ_REC;            ! PACKET SIZE                              RECEIVE DATA
;    4114  2          MSCP_PKT [.MX1, OPCODE] = OP_RCD;            ! OPCODE = GET DUST STATUS
;    4115  2          MSCP_PKT [.MX1, BC_LO] = 514;                ! BYTE COUNT TO BE TRANSFERED EQUALS 512
;    4116  2          MSCP_PKT [.MX1, BUF_0] = DUPPKT;             ! LOAD DESCRIBTOR BUFFER
;    4117  2          MSCP_PKT [.MX1, MODIFY] = 0;                 !
;    4118  2          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;           ! CALL IT sequential
;    4119  2          DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
;    4120  2
;    4121  2          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;    4122  2             (.DUPPKT [DUPTYPE] NEQU 6) OR             !dup type error
;    4123  3             (.DUPPKT [DUPMESG] NEQU 2)
;    4124  2          THEN
;    4125  3                  (HARD_ERROR ();
;    4126  3                   CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    4127  2                   RETURN;);                                      ! NO POINT IN CONTINUING
;    4128  2
;    4129  3     DO (MX1 = GET_PKT (.CCTLR))
;    4130  2     UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4131  2
;    4132  2     T_ADDR [T_BLK_RD] = .T_ADDR [T_BLK_RD] + 1;              !IF DUP NO ERROR THEN INCREMENT COUNTER
;    4133  2
;    4134  1     END;


                                        .SBTTL  DUPREDDBN MULTI-DRIVE TEST ROUTINES
000000  010146                    DUPREDDBN::
                                        MOV     R1,-(SP)                        ;                                  4041
000002  013700  000000G                 MOV     T.ADDR,R0                       ;                                  4058
000006  005260  000064                  INC     64(R0)
000012  013746  000112'                 MOV     MX1,-(SP)                       ;                                  4060
000016  012746  000106                  MOV     #106,-(SP)
000022  004737  000000G                 JSR     PC,BL$MUL
000026  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
000034  112760  000003  000022G         MOVB    #3,MSCP.PKT+22(R0)              ;                                  4061
```

```
000042  112760  000122  000026G                MOVB    #122,MSCP.PKT+26(R0)    ;                4062
000050  112760  000105  000027G                MOVB    #105,MSCP.PKT+27(R0)    ;                4063
000056  112760  000104  000030G                MOVB    #104,MSCP.PKT+30(R0)    ;                4064
000064  112760  000104  000031G                MOVB    #104,MSCP.PKT+31(R0)    ;                4065
000072  112760  000102  000032G                MOVB    #102,MSCP.PKT+32(R0)    ;                4066
000100  112760  000116  000033G                MOVB    #116,MSCP.PKT+33(R0)    ;                4067
000106  012760  000001  000024G                MOV     #1,MSCP.PKT+24(R0)      ;                4068
000114  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;                4069
000122  004737  000000V                        JSR     PC,DUPCOMMAND           ;                4070
000126  013700  001162'                        MOV     DUOFF,R0                ;                4072
000132  006300                                ASL     R0
000134  063700  000000G                        ADD     CST.ADDR,R0
000140  032760  040000  000020                BIT     #40000,20(R0)
000146  001402                                BEQ     1$
000150  022626                                CMP     (SP)+,(SP)+             ;                4041
000152  000505                                BR      3$                      ;                4073
000154  013716  000000G                1$:     MOV     CCTLR,(SP)              ;                4075
000160  004737  000000G                        JSR     PC,GET.PKT
000164  010037  000112'                        MOV     R0,MX1
000170  002771                                BLT     1$                      ;                4076
000172  010016                                MOV     R0,(SP)                 ;                4078
000174  012746  000106                        MOV     #106,-(SP)
000200  004737  000000G                        JSR     PC,BL$MUL
000204  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000212  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)      ;                4079
000220  012760  000120  000026G                MOV     #120,MSCP.PKT+26(R0)    ;                4080
000226  012760  000000G  000032G                MOV     #DUPPKT,MSCP.PKT+32(R0) ;                4081
000234  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;                4082
000240  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;                4083
000246  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000254  004737  000000V                        JSR     PC,DUPCOMMAND           ;                4084
000260  013700  001162'                        MOV     DUOFF,R0                ;                4086
000264  006300                                ASL     R0
000266  063700  000000G                        ADD     CST.ADDR,R0
000272  032760  040000  000020                BIT     #40000,20(R0)
000300  001016                                BNE     2$
000302  013700  000000G                        MOV     DUPPKT,R0               ;                4087
000306  042700  007777                        BIC     #7777,R0
000312  020027  010000                        CMP     R0,#10000
000316  001007                                BNE     2$
000320  013700  000000G                        MOV     DUPPKT,R0               ;                4088
000324  042700  170000                        BIC     #170000,R0
000330  020027  000005                        CMP     R0,#5
000334  001415                                BEQ     4$
000336  004737  000000V                2$:     JSR     PC,HARD.ERROR           ;                4090
000342  013700  001162'                        MOV     DUOFF,R0                ;                4091
000346  006300                                ASL     R0
000350  063700  000000G                        ADD     CST.ADDR,R0
000354  052760  040000  000020                BIS     #40000,20(R0)
000362  062706  000006                        ADD     #6,SP                   ;                4092
000366  000501                        3$:     BR      5$                      ;                4090
000370  013716  000000G                4$:     MOV     CCTLR,(SP)              ;                4094
000374  004737  000000G                        JSR     PC,GET.PKT
```

```
000400  010037  000112'                        MOV     R0,MX1
000404  002771                                 BLT     4$                              ;  4095
000406  010016                                 MOV     R0,(SP)                         ;  4097
000410  012746  000106                         MOV     #106,-(SP)
000414  004737  000000G                        JSR     PC,BL$MUL
000420  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000426  112760  000004  000022G                MOVB    #4,MSCP.PKT+22(R0)              ;  4098
000434  012760  000004  000026G                MOV     #4,MSCP.PKT+26(R0)              ;  4099
000442  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)         ;  4100
000450  013701  001162'                        MOV     DUOFF,R1                        ;  4101
000454  006301                                 ASL     R1
000456  063701  000000G                        ADD     CST.ADDR,R1
000462  111137  000000G                        MOVB    (R1),DUPPKT
000466  042737  177760  000000G               BIC     #177760,DUPPKT
000474  013701  001162'                        MOV     DUOFF,R1                        ;  4102
000500  006301                                 ASL     R1
000502  063701  000000G                        ADD     CST.ADDR,R1
000506  116137  000020  000002G                MOVB    20(R1),DUPPKT+2
000514  105037  000003G                        CLRB    DUPPKT+3
000520  005060  000024G                        CLR     MSCP.PKT+24(R0)                 ;  4103
000524  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)            ;  4104
000532  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000540  004737  000000V                        JSR     PC,DUPCOMMAND                   ;  4105
000544  013700  001162'                        MOV     DUOFF,R0                        ;  4107
000550  006300                                 ASL     R0
000552  063700  000000G                        ADD     CST.ADDR,R0
000556  032760  040000  000020                BIT     #40000,20(R0)
000564  001403                                 BEQ     6$
000566  062706  000010                         ADD     #10,SP                          ;  4041
000572  000521                         5$:     BR      10$                             ;  4108
000574  013716  000000G                 6$:     MOV     CCTLR,(SP)                      ;  4110
000600  004737  000000G                        JSR     PC,GET.PKT
000604  010037  000112'                        MOV     R0,MX1
000610  002771                                 BLT     6$                              ;  4111
000612  010016                                 MOV     R0,(SP)                         ;  4113
000614  012746  000106                         MOV     #106,-(SP)
000620  004737  000000G                        JSR     PC,BL$MUL
000624  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000632  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)              ;  4114
000640  012760  001002  000026G                MOV     #1002,MSCP.PKT+26(R0)           ;  4115
000646  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)         ;  4116
000654  005060  000024G                        CLR     MSCP.PKT+24(R0)                 ;  4117
000660  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)            ;  4118
000666  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000674  004737  000000V                        JSR     PC,DUPCOMMAND                   ;  4119
000700  013700  001162'                        MOV     DUOFF,R0                        ;  4121
000704  006300                                 ASL     R0
000706  063700  000000G                        ADD     CST.ADDR,R0
000712  032760  040000  000020                BIT     #40000,20(R0)
000720  001016                                 BNE     7$
000722  013700  000000G                        MOV     DUPPKT,R0                       ;  4122
000726  042700  007777                         BIC     #7777,R0
000732  020027  060000                         CMP     R0,#60000
```

```
000736  001007                              BNE     7$
000740  013700  000000G                     MOV     DUPPKT,R0                       ;                       4123
000744  042700  170000                      BIC     #170000,R0
000750  020027  000002                      CMP     R0,#2
000754  001413                              BEQ     8$
000756  004737  000000V             7$:     JSR     PC,HARD.ERROR                   ;                       4125
000762  013700  001162'                     MOV     DUOFF,R0                        ;                       4126
000766  006300                              ASL     R0
000770  063700  000000G                     ADD     CST.ADDR,R0
000774  052760  040000  000020             BIS     #40000,20(R0)
001002  000413                              BR      9$                              ;                       4127
001004  013716  000000G             8$:     MOV     CCTLR,(SP)                      ;                       4129
001010  004737  000000G                     JSR     PC,GET.PKT
001014  010037  000112'                     MOV     R0,MX1
001020  002771                              BLT     8$                              ;                       4130
001022  013700  000000G                     MOV     T.ADDR,R0                       ;                       4132
001026  005260  000062                      INC     62(R0)
001032  062706  000012             9$:     ADD     #12,SP                          ;                       4056
001036  012601                     10$:    MOV     (SP)+,R1                        ;                       4041
001040  000207                              RTS     PC

; Routine Size:  273 words,     Routine Base:  $CODE$ + 15264
; Maximum stack depth per invocation:  7 words


;   4135  1
```

```
;  4136  1
;  4137  1     GLOBAL ROUTINE DUPCOMMAND : NOVALUE =
;  4138  1
;  4139  1     !+
;  4140  1     ! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
;  4141  1     ! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
;  4142  1     ! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
;  4143  1     ! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
;  4144  1     !-
;  4145  2     BEGIN
;  4146  2     !PRINTX (DER13);
;  4147  2
;  4148  2     MSCP_PKT [.MX1, CREDITS] = 0;                   ! DUP DOES NOT USE THE CREDIT SYSTEM
;  4149  2     MSCP_PKT [.MX1, CONNID] = CID_DUP;              ! MAKE PACKAGE EQUAL A DUP COMMAND
;  4150  2     MSCP_PKT [.MX1, DK_NUM] = 0;                    ! DISK NUMBER (NOT APPLICABLE)
;  4151  2
;  4152  2     IF SEND (.MX1) EQLU FAILURE                     ! ATTEMPT SEND; IF CTLR IS OFFLINE
;  4153  2     THEN
;  4154  3             BEGIN
;  4155  3             PUT_PKT (.MX1);
;  4156  3             MX1 = -1;                               ! RETURN ENVELOPE TO POOL
;  4157  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;
;  4158  3     !       PRINTF (DBM112);                        ! "DUP: PKT NOT AVAILABLE"  ZZZ
;  4159  3             END
;  4160  3
;  4161  2     ELSE
;  4162  2             do
;  4163  3                     begin
;  4164  3                     BREAK;                                  ! BREAK FOR ACT
;  4165  3                     PROC_RETPKT ();                                 ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;  4166  3                     end
;  4167  2             until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;  4168  2             (.EOP_FLAG eql true);                           ! or end of pass caused by error
;  4169  1     END;


                                        .SBTTL  DUPCOMMAND MULTI-DRIVE TEST ROUTINES
000000  013746  000112'              DUPCOMMAND::
                                        MOV     MX1,-(SP)                               ;                               4148
000004  012746  000106                  MOV     #106,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)
000022  112760  000002  000011G         MOVB    #2,MSCP.PKT+11(R0)              ;               4149
000030  005060  000016G                 CLR     MSCP.PKT+16(R0)                 ;               4150
000034  013716  000112'                 MOV     MX1,(SP)                        ;               4152
000040  004737  000000G                 JSR     PC,SEND
000044  005700                          TST     R0
000046  001020                          BNE     1$
000050  013716  000112'                 MOV     MX1,(SP)                        ;               4155
000054  004737  000000G                 JSR     PC,PUT.PKT
000060  012737  177777  000112'         MOV     #-1,MX1                         ;               4156
000066  013700  001162'                 MOV     DUOFF,R0                        ;               4157
000072  006300                          ASL     R0
```

```
000074  063700  000000G                        ADD     CST.ADDR,R0
000100  052760  040000  000020                BIS     #40000,20(R0)
000106  000415                                 BR      2$                      ;                       4152
000110  104422                          1$:    TRAP    22                      ;                       4163
000112  004737  000000V                        JSR     PC,PROC.RETPKT          ;                       4165
000116  013700  000000G                        MOV     RP.ADDR,R0              ;                       4167
000122  023760  000000G 000004                CMP     CRN.LOW,4(R0)
000130  001404                                 BEQ     2$
000132  123727  000000G 000001                CMPB    EOP.FLAG,#1             ;                       4168
000140  001363                                 BNE     1$
000142  022626                          2$:    CMP     (SP)+,(SP)+             ;                       4145
000144  000207                                 RTS     PC                      ;                       4137

; Routine Size:  51 words,      Routine Base:  $CODE$ + 16326
; Maximum stack depth per invocation:  4 words
```

```
;    4170  1
;    4171  1     GLOBAL ROUTINE DUPIDLE : NOVALUE =
;    4172  1     !+
;    4173  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO INSURE THAT THE CONTROLLER
;    4174  1     ! IS NOT IN A ACTIVE STATE. IF CALLED AND THE CONTROLLER IS IN AN ACTIVE
;    4175  1     ! STATE THE CONTROLLER WILL GIVE AN ABORT COMMAND WHICH SHOULD KILL THE
;    4176  1     ! CURRENT JOB OR LOCAL PROGRAM.
;    4177  1     !-
;    4178  2     BEGIN
;    4179  2     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 0;              !CLEAR DUP ERROR FLAG;
;    4180  2
;    4181  2     MSCP_PKT [.MX1, MSGLEN] = SZ_GDS;              ! PACKET SIZE                    GET DUST STATUS
;    4182  2     MSCP_PKT [.MX1, OPCODE] = OP_GDS;              ! OPCODE = GET DUST STATUS
;    4183  2     MSCP_PKT [.MX1, MODIFY] = 0;                   !
;    4184  2     MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;             ! CALL IT IMMEDIATE
;    4185  2     DUPCOMMAND ();                                 ! SENDS AND RECEIVES THE COMMAND
;    4186  2                                                    ! GDS ONLY RETURNS SUCCESS or it don't return
;    4187  2
;    4188  3     DO (MX1 = GET_PKT (.CCTLR))
;    4189  2     UNTIL (.MX1 GEQ 0);                            ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;    4190  2
;    4191  2     if .CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] neq IDLE      ! if not in idle state then abort the program
;    4192  2     then
;    4193  3         begin
;    4194  3         MSCP_PKT [.MX1, MSGLEN] = SZ_ABT;                  ! PACKET SIZE                    ABORT CMD
;    4195  3         MSCP_PKT [.MX1, OPCODE] = OP_ABT;                  ! OPCODE = ABORT PROGRAM
;    4196  3         MSCP_PKT [.MX1, MODIFY] = 0;                       !
;    4197  3         MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                 ! CALL IT IMMEDIATE
;    4198  3         DUPCOMMAND ();                                     ! SENDS AND RECEIVES THE COMMAND
;    4199  3                                                            !ONLY ERROR IS already in idle state
;    4200  4         DO (MX1 = GET_PKT (.CCTLR))
;    4201  3         UNTIL (.MX1 GEQ 0);                        ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;    4202  2         end;
;    4203  1     end;


                                        .SBTTL  DUPIDLE MULTI-DRIVE TEST ROUTINES
000000  010146                  DUPIDLE::
                                        MOV     R1,-(SP)                        ;                       4171
000002  013700  001162'                 MOV     DUOFF,R0                        ;                       4179
000006  006300                          ASL     R0
000010  063700  000000G                 ADD     CST.ADDR,R0
000014  042760  040000  000020          BIC     #40000,20(R0)
000022  013746  000112'                 MOV     MX1,-(SP)                       ;                       4181
000026  012746  000106                  MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  012760  000014  000006G         MOV     #14,MSCP.PKT+6(R0)
000044  112760  000001  000022G         MOVB    #1,MSCP.PKT+22(R0)              ;                       4182
000052  005060  000024G                 CLR     MSCP.PKT+24(R0)                 ;                       4183
000056  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                       4184
000064  004737  016326'                 JSR     PC,DUPCOMMAND                   ;                       4185
000070  013716  000000G         1$:     MOV     CCTLR,(SP)                      ;                       4188
000074  004737  000000G                 JSR     PC,GET.PKT
```

```
000100  010037  000112'                        MOV     R0,MX1
000104  010001                                 MOV     R0,R1                   ; MX1,*                 4189
000106  002770                                 BLT     1$
000110  013700  001162'                        MOV     DUOFF,R0                ;                       4191
000114  006300                                 ASL     R0
000116  063700  000000G                        ADD     CST.ADDR,R0
000122  032760  020000  000020                 BIT     #20000,20(R0)
000130  001432                                 BEQ     3$
000132  010116                                 MOV     R1,(SP)                 ;                       4194
000134  012746  000106                        MOV     #106,-(SP)
000140  004737  000000G                        JSR     PC,BL$MUL
000144  012760  000014  000006G                MOV     #14,MSCP.PKT+6(R0)
000152  112760  000006  000022G                MOVB    #6,MSCP.PKT+22(R0)      ;                       4195
000160  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;                       4196
000164  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;                       4197
000172  004737  016326'                        JSR     PC,DUPCOMMAND           ;                       4198
000176  013716  000000G                2$:     MOV     CCTLR,(SP)              ;                       4200
000202  004737  000000G                        JSR     PC,GET.PKT
000206  010037  000112'                        MOV     R0,MX1
000212  002771                                 BLT     2$                      ;                       4201
000214  005726                                 TST     (SP)+                   ;                       4193
000216  022626                        3$:     CMP     (SP)+,(SP)+             ;                       4178
000220  012601                                 MOV     (SP)+,R1                ;                       4171
000222  000207                                 RTS     PC

; Routine Size:  74 words,      Routine Base:  $CODE$ + 16474
; Maximum stack depth per invocation:  5 words
```

```
;    4204  1     GLOBAL ROUTINE QIO_LBN : NOVALUE =
;    4205  1
;    4206  1     !+
;    4207  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
;    4208  1     !       BE USED FOR THE CURRENT QIO OR QIO PAIR.
;    4209  1     !
;    4210  1     !       IF THE OPERATOR CHOSE THE RANDOM BLOCK MODE OPTION, THEN THE LBN IS
;    4211  1     !       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR THE LBN.
;    4212  1     !       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
;    4213  1     !       TABLE (BST).
;    4214  1     !
;    4215  1     !       IMPLICIT INPUTS:
;    4216  1     !               L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
;    4217  1     !
;    4218  1     !       IMPLICIT OUTPUTS:
;    4219  1     !               THE LBN IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4220  1     !-
;    4221  1
;    4222  2     begin
;    4223  2
;    4224  2     local
;    4225  2         RD_DISK : byte;      ! FLAG TO INDICATE WINCHESTER DISK SELECTED
;    4226  2
;    4227  2
;    4228  2     if .CST_ADDR [.CUOFF, D_TYPE] eql RX_50
;    4229  2     then
;    4230  2         RD_DISK = FALSE
;    4231  2     else
;    4232  2         RD_DISK = TRUE;
;    4233  2
;    4234  2     MAD1 [LBN_L] = .BST [.L$LUN, LO_WRD];                    ! LOAD LBN INTO 1ST PACKET
;    4235  2     MAD1 [LBN_H] = .BST [.L$LUN, HI_WRD];                    ! LOAD LBN INTO 1ST PACKET
;    4236  2
;    4237  2     if .MX2 geq 0                              ! IF 2 QIOS
;    4238  2     then
;    4239  3         begin
;    4240  3         MAD2 [LBN_L] = .BST [.L$LUN, LO_WRD];                        ! LOAD LBN INTO 2ND PACKET
;    4241  3         MAD2 [LBN_H] = .BST [.L$LUN, HI_WRD];                ! LOAD LBN INTO 2ND PACKET
;    4242  2         end;
;    4243  2
;    4244  3     if BIT_TST (SWP_FLAGS, SWF_BLK)            ! IF RANDOM BLOCK MODE
;    4245  2     then
;    4246  3         begin
;    4247  4         if  NOT ((.RD_DISK) and
;    4248  4           (((.RANDOM [0] and %o'077777') mod (99)) lequ 33))                    ! REDUCES SEEKS TO 66% OF TRANFERS o
n winchesters
;    4249  3         then
;    4250  4             begin                                                             ! NOTE IF NOT CALLED THE LBN REMAINS
 THE SAME
;    4251  4             if .CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .CST_ADDR [.CUOFF + 4, D_END1]
;    4252  4             then                                                              ! if upper word of beg trk and endin
g trk same
;    4253  4                     BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0] +           ! select low lbn from random
 number WINDOW
;    4254  4                     MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], .CST_ADDR [.CUOFF + 3, D_END0])
;    4255  4
;    4256  4             else                                                              ! if upper word of beg trk and end t
rk different
```

```
;    4257  5                  begin
;    4258  5                  BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1] +              ! select upper lbn from window
;    4259  5                  MODULAS (.CST_ADDR [.CUOFF + 2, D_BEG1], .CST_ADDR [.CUOFF + 4, D_END1]);
;    4260  5
;    4261  5                  if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 4, D_END1]          ! IF UPPER WORD EQUALS HI LIMIT BE S
URE LOWER
;    4262  5                  then .BST [.L$LUN, LO_WRD] = MODULAS (0, .CST_ADDR [.CUOFF + 3, D_END0]); ! WORD DOES NOT PASS HI LIMIT
;    4263  5
;    4264  5                  if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 2, D_BEG1]          ! if upper word equal lo limit make
sure lower
;    4265  5                  then BST [.L$LUN, LO_WRD] = %O'177777' - MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], %O'177777');
;    4266  5                                                                                       ! word is above lo limit
;    4267  5
;    4268  5                  if .BST [.L$LUN, HI_WRD] gtr .CST_ADDR [.CUOFF + 2, D_BEG1] and
;    4269  5                     .BST [.L$LUN, HI_WRD] lss .CST_ADDR [.CUOFF + 4, D_END1]          ! if neither of the above then any n
umber is good
;    4270  5                  then BST [.L$LUN, LO_WRD] = .RANDOM [5];
;    4271  4                  end;
;    4272  3              end;
;    4273  3          end
;    4274  2      else
;    4275  3          begin                                          ! ELSE - SEQUENTIAL LBN MODE
;    4276  4          if (.TRK_SGN [.L$LUN] geq 1)
;    4277  3          then                                           ! if positive track direction add one to multiword
;    4278  4              (if .BST [.L$LUN, LO_WRD] eqlu %o'177777'
;    4279  4              then
;    4280  5                      begin
;    4281  5                      BST [.L$LUN, LO_WRD] = 0;
;    4282  5                      BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1;
;    4283  5                      end
;    4284  4              else
;    4285  4                      BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] + 1)
;    4286  3          else                                           ! if negative track direction subtract one from multiword
;    4287  3              if .BST [.L$LUN, LO_WRD] eqlu %o'0'
;    4288  3              then
;    4289  4                      begin
;    4290  4                      BST [.L$LUN, LO_WRD] = %o'177777';
;    4291  4                      BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] - 1;
;    4292  4                      end
;    4293  3              else
;    4294  3                      BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] - 1;
;    4295  3
;    4296  3
;    4297  3          if  .BST [.L$LUN, LO_WRD] gequ (.CST_ADDR [.CUOFF + 3, D_END0]) and ! if hi limit then change direction
;    4298  4              .BST [.L$LUN, HI_WRD] gequ (.CST_ADDR [.CUOFF + 4, D_END1])
;    4299  3          then TRK_SGN [.L$LUN] =  -1;
;    4300  3
;    4301  3          if  .BST [.L$LUN, LO_WRD] lequ (.CST_ADDR [.CUOFF + 1, D_BEG0] + 1)  and    ! if low limit then change direction
;    4302  4              .BST [.L$LUN, HI_WRD] lequ (.CST_ADDR [.CUOFF + 2, D_BEG1])
;    4303  3          then TRK_SGN [.L$LUN] = 1;
;    4304  2          end;
;    4305  1      end;                                                   ! ROUTINE QIO_LBN


                                        .SBTTL  QIO.LBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                QIO.LBN::
```

```
                                        JSR     R1,$SAVE5                       ;                               4204
000004  013705  000000G                 MOV     CST.ADDR,R5                     ;                               4228
000010  013700  000000G                 MOV     CUOFF,R0
000014  006300                          ASL     R0
000016  060500                          ADD     R5,R0
000020  132710  000020                  BITB    #20,(R0)
000024  001002                          BNE     1$
000026  105004                          CLRB    R4                              ; RD.DISK                       4230
000030  000402                          BR      2$                              ;                               4228
000032  112704  000001          1$:     MOVB    #1,R4                           ; *,RD.DISK                     4232
000036  013700  000116'        2$:     MOV     MAD1,R0                         ;                               4234
000042  013701  000000G                 MOV     L$LUN,R1
000046  010103                          MOV     R1,R3
000050  006303                          ASL     R3
000052  006303                          ASL     R3
000054  012702  000000G                 MOV     #BST,R2
000060  060302                          ADD     R3,R2
000062  011260  000046                  MOV     (R2),46(R0)
000066  062703  000002G                 ADD     #BST+2,R3                       ;                               4235
000072  011360  000050                  MOV     (R3),50(R0)
000076  005737  000114'                 TST     MX2                             ;                               4237
000102  002406                          BLT     3$
000104  013700  000120'                 MOV     MAD2,R0                         ;                               4240
000110  011260  000046                  MOV     (R2),46(R0)
000114  011360  000050                  MOV     (R3),50(R0)                     ;                               4241
000120  032737  040000  000000G 3$:     BIT     #40000,SWP.FLAGS                ;                               4244
000126  001002                          BNE     4$
000130  000137  017474'                 JMP     10$
000134  006004                  4$:     ROR     R4                              ; RD.DISK                       4247
000136  103015                          BCC     5$
000140  013746  000000G                 MOV     RANDOM,-(SP)                    ;                               4248
000144  042716  100000                  BIC     #100000,(SP)
000150  012746  000143                  MOV     #143,-(SP)
000154  004737  000000G                 JSR     PC,BL$MOD
000160  022626                          CMP     (SP)+,(SP)+
000162  020027  000041                  CMP     R0,#41
000166  101001                          BHI     5$
000170  000207                          RTS     PC
000172  013700  000000G          5$:     MOV     CUOFF,R0                        ;                               4251
000176  006300                          ASL     R0
000200  060500                          ADD     R5,R0
000202  016004  000004                  MOV     4(R0),R4
000206  013700  000000G                 MOV     CUOFF,R0
000212  006300                          ASL     R0
000214  060500                          ADD     R5,R0
000216  020460  000010                  CMP     R4,10(R0)
000222  001022                          BNE     6$
000224  013701  000000G                 MOV     CUOFF,R1                        ;                               4253
000230  006301                          ASL     R1
000232  060501                          ADD     R5,R1
000234  016146  000002                  MOV     2(R1),-(SP)                     ;                               4254
000240  013700  000000G                 MOV     CUOFF,R0
000244  006300                          ASL     R0
```

```
000246  060500                          ADD     R5,R0
000250  016046  000006                  MOV     6(R0),-(SP)
000254  004737  000000G                 JSR     PC,MODULAS
000260  066100  000002                  ADD     2(R1),R0        ;                       4253
000264  010012                          MOV     R0,(R2)
000266  000530                          BR      9$              ;                       4251
000270  010446                  6$:     MOV     R4,-(SP)        ;                       4259
000272  016046  000010                  MOV     10(R0),-(SP)
000276  004737  000000G                 JSR     PC,MODULAS
000302  060400                          ADD     R4,R0           ;                       4258
000304  010013                          MOV     R0,(R3)
000306  013701  000000G                 MOV     L$LUN,R1        ;                       4261
000312  006301                          ASL     R1
000314  006301                          ASL     R1
000316  013700  000000G                 MOV     CUOFF,R0
000322  006300                          ASL     R0
000324  063700  000000G                 ADD     CST.ADDR,R0
000330  026160  000002G 000010          CMP     BST+2(R1),10(R0)
000336  001015                          BNE     7$
000340  005016                          CLR     (SP)            ;                       4262
000342  013700  000000G                 MOV     CUOFF,R0
000346  006300                          ASL     R0
000350  063700  000000G                 ADD     CST.ADDR,R0
000354  016046  000006                  MOV     6(R0),-(SP)
000360  004737  000000G                 JSR     PC,MODULAS
000364  010071  000000G                 MOV     R0,@BST(R1)
000370  005726                          TST     (SP)+
000372  013701  000000G          7$:    MOV     L$LUN,R1        ;                       4264
000376  006301                          ASL     R1
000400  006301                          ASL     R1
000402  013700  000000G                 MOV     CUOFF,R0
000406  006300                          ASL     R0
000410  063700  000000G                 ADD     CST.ADDR,R0
000414  026160  000002G 000004          CMP     BST+2(R1),4(R0)
000422  001021                          BNE     8$
000424  013700  000000G                 MOV     CUOFF,R0        ;                       4265
000430  006300                          ASL     R0
000432  063700  000000G                 ADD     CST.ADDR,R0
000436  016016  000002                  MOV     2(R0),(SP)
000442  012746  177777                  MOV     #-1,-(SP)
000446  004737  000000G                 JSR     PC,MODULAS
000452  012761  177777  000000G        MOV     #-1,BST(R1)
000460  160061  000000G                 SUB     R0,BST(R1)
000464  005726                          TST     (SP)+
000466  013700  000000G          8$:    MOV     L$LUN,R0        ;                       4268
000472  006300                          ASL     R0
000474  006300                          ASL     R0
000476  013701  000000G                 MOV     CUOFF,R1
000502  006301                          ASL     R1
000504  063701  000000G                 ADD     CST.ADDR,R1
000510  026061  000002G 000004          CMP     BST+2(R0),4(R1)
000516  003414                          BLE     9$
000520  013701  000000G                 MOV     CUOFF,R1        ;                       4269
```

```
000524  006301                          ASL     R1
000526  063701  000000G                 ADD     CST.ADDR,R1
000532  026061  000002G 000010          CMP     BST+2(R0),10(R1)
000540  002003                          BGE     9$
000542  013760  000012G 000000G         MOV     RANDOM+12,BST(R0)       ;                4270
000550  022626                  9$:     CMP     (SP)+,(SP)+             ;                4250
000552  000207                          RTS     PC                      ;                4244
000554  062701  000000G         10$:    ADD     #TRK.SGN,R1             ;                4276
000560  105711                          TSTB    (R1)
000562  003410                          BLE     12$
000564  021227  177777                  CMP     (R2),#-1                ;                4278
000570  001003                          BNE     11$
000572  005012                          CLR     (R2)                    ;                4281
000574  005213                          INC     (R3)                    ;                4282
000576  000411                          BR      14$                     ;                4278
000600  005212                  11$:    INC     (R2)                    ;                4285
000602  000407                          BR      14$                     ;                4276
000604  005712                  12$:    TST     (R2)                    ;                4287
000606  001004                          BNE     13$
000610  012712  177777                  MOV     #-1,(R2)                ;                4290
000614  005313                          DEC     (R3)                    ;                4291
000616  000401                          BR      14$                     ;                4287
000620  005312                  13$:    DEC     (R2)                    ;                4294
000622  013700  000000G         14$:    MOV     CUOFF,R0                ;                4297
000626  006300                          ASL     R0
000630  060500                          ADD     R5,R0
000632  021260  000006                  CMP     (R2),6(R0)
000636  103411                          BLO     15$
000640  013700  000000G                 MOV     CUOFF,R0                ;                4298
000644  006300                          ASL     R0
000646  060500                          ADD     R5,R0
000650  021360  000010                  CMP     (R3),10(R0)
000654  103402                          BLO     15$
000656  112711  000377                  MOVB    #377,(R1)               ;                4299
000662  013700  000000G         15$:    MOV     CUOFF,R0                ;                4301
000666  006300                          ASL     R0
000670  060500                          ADD     R5,R0
000672  016000  000002                  MOV     2(R0),R0
000676  005200                          INC     R0
000700  021200                          CMP     (R2),R0
000702  101011                          BHI     16$
000704  013700  000000G                 MOV     CUOFF,R0                ;                4302
000710  006300                          ASL     R0
000712  060500                          ADD     R5,R0
000714  021360  000004                  CMP     (R3),4(R0)
000720  101002                          BHI     16$
000722  112711  000001                  MOVB    #1,(R1)                 ;                4303
000726  000207                  16$:    RTS     PC                      ;                4204

; Routine Size:  236 words,    Routine Base:  $CODE$ + 16720
; Maximum stack depth per invocation:  10 words
```

```
;    4306  1     !!ZZZ routine QIO_SIZE : novalue =
;    4307  1     GLOBAL ROUTINE QIO_SIZE : NOVALUE =
;    4308  1
;    4309  1     !+
;    4310  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;    4311  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
;    4312  1     !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS FALL BETWEEN 1 AND THE
;    4313  1     !       I/O BUFFER SIZE (BYTS_PER_QIO). It is assumed that BYTS_PER_QIO will
;    4314  1     !       never be larger than one binary word or  65000 bytes.
;    4315  1     !
;    4316  1     !       IMPLICIT OUTPUTS:
;    4317  1     !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4318  1     !-
;    4319  1
;    4320  2         begin
;    4321  2
;    4322  2         local
;    4323  2             SIZE : word,                                        ! BYTE COUNT
;    4324  2             BLOCKS_LEFT : word;                                 ! REMAINING BLOCKS LEFT
;    4325  2
;    4326  2         SIZE = ((.RANDOM [4] and %o'077777') mod (.BYTS_PER_QIO + 1)) and %o'177760';!GET BYTE COUNT FROM RANDOM NUMBER
;    4327  2
;    4328  2         if .SIZE eql 0
;    4329  2         then
;    4330  2             SIZE = 16;
;    4331  2
;    4332  2         if .CST_ADDR [.CUOFF + 4, D_END1] gtru .MAD1 [LBN_H]
;    4333  2         then BLOCKS_LEFT = %o'177777'                           ! find
;    4334  2         else BLOCKS_LEFT = .CST_ADDR [.CUOFF + 3, D_END0] - .MAD1 [LBN_L] + 1;   ! REMAINING BLOCK COUNT
;    4335  2
;    4336  2         if ((.SIZE + BYTES_PER_SECT - 1) / BYTES_PER_SECT) gtru .BLOCKS_LEFT     ! IF BLOCK COUNT NOT ENOUGH
;    4337  2         then
;    4338  2             SIZE = .BLOCKS_LEFT * BYTES_PER_SECT;               ! ADJUST BYTE COUNT DOWN
;    4339  2
;    4340  2         MAD1 [BC_LO] = .SIZE;                                   ! LOAD SIZE INTO 1ST MSCP PACKET
;    4341  2
;    4342  2         if .MX2 geq 0                                           ! IF 2 QIOS
;    4343  2         then
;    4344  2             MAD2 [BC_LO] = .SIZE;                               ! LOAD SIZE INTO 2ND MSCP PACKET
;    4345  2
;    4346  1         end;                                                    ! ROUTINE QIO_SIZE


                                        .SBTTL  QIO.SIZE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          QIO.SIZE::
                                        JSR     R1,$SAVE3                   ;                                   4307
000004  013746  000010G                 MOV     RANDOM+10,-(SP)             ;                                   4326
000010  042716  100000                  BIC     #100000,(SP)
000014  013746  000000G                 MOV     BYTS.PER.QIO,-(SP)
000020  005216                          INC     (SP)
000022  004737  000000G                 JSR     PC,BL$MOD
000026  010003                          MOV     R0,R3                       ; *,SIZE
000030  042703  000017                  BIC     #17,R3                      ; *,SIZE
```

```
000034  001002                          BNE     1$                      ;                               4328
000036  012703  000020                  MOV     #20,R3                  ; *,SIZE                        4330
000042  013700  000000G         1$:     MOV     CUOFF,R0                ;                               4332
000046  006300                          ASL     R0
000050  063700  000000G                 ADD     CST.ADDR,R0
000054  013701  000116'                 MOV     MAD1,R1
000060  026061  000010  000050          CMP     10(R0),50(R1)
000066  101403                          BLOS    2$
000070  012702  177777                  MOV     #-1,R2                  ; *,BLOCKS.LEFT                 4333
000074  000413                          BR      3$                      ;                               4332
000076  013700  000000G         2$:     MOV     CUOFF,R0                ;                               4334
000102  006300                          ASL     R0
000104  063700  000000G                 ADD     CST.ADDR,R0
000110  016000  000006                  MOV     6(R0),R0
000114  166100  000046                  SUB     46(R1),R0
000120  010002                          MOV     R0,R2                   ; *,BLOCKS.LEFT
000122  005202                          INC     R2                      ; BLOCKS.LEFT
000124  010316                  3$:     MOV     R3,(SP)                 ; SIZE,*                        4336
000126  062716  000777                  ADD     #777,(SP)
000132  012746  001000                  MOV     #1000,-(SP)
000136  004737  000000G                 JSR     PC,BL$DIV
000142  005726                          TST     (SP)+
000144  020002                          CMP     R0,R2                   ; *,BLOCKS.LEFT
000146  101405                          BLOS    4$
000150  010200                          MOV     R2,R0                   ; BLOCKS.LEFT,*                 4338
000152  000300                          SWAB    R0
000154  105000                          CLRB    R0
000156  006300                          ASL     R0
000160  010003                          MOV     R0,R3                   ; *,SIZE
000162  010361  000026          4$:     MOV     R3,26(R1)               ; SIZE,*                        4340
000166  005737  000114'                 TST     MX2                     ;                               4342
000172  002404                          BLT     5$
000174  013700  000120'                 MOV     MAD2,R0                 ;                               4344
000200  010360  000026                  MOV     R3,26(R0)               ; SIZE,*
000204  022626                  5$:     CMP     (SP)+,(SP)+             ;                               4320
000206  000207                          RTS     PC                      ;                               4307

; Routine Size:  68 words,      Routine Base:  $CODE$ + 17650
; Maximum stack depth per invocation:  8 words
```

```
;  4347  1     GLOBAL routine FILL_BUFF : novalue =
;  4348  1
;  4349  1     !+
;  4350  1     !       THIS ROUTINE IS CALLED BY QIO GEN TO LOAD THE I/O BUFFER DESCRIBED IN
;  4351  1     !       THE FIRST MSCP PACKET WITH THE APPROPRIATE DATA PATTERN.
;  4352  1     !
;  4353  1     !       THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
;  4354  1     !
;  4355  1     !               IF THE OPERATOR DEFINED A DATA PATTERN
;  4356  1     !               THEN
;  4357  1     !                       SELECT IT
;  4358  1     !               ELSE
;  4359  1     !                       GET DATA PATTERN NUMBER FROM SW P-TABLE
;  4360  1     !                       IF DATA PATTERN NUMBER = 0
;  4361  1     !                       THEN
;  4362  1     !                               GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
;  4363  1     !                               IN THE DATA PATTERN SEQUENCE TABLE (DPST)
;  4364  1     !
;  4365  1     !       NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
;  4366  1     !       21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
;  4367  1     !
;  4368  1     !       IMPLICIT INPUTS:
;  4369  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;  4370  1     !-
;  4371  1
;  4372  2         begin
;  4373  2
;  4374  2         local
;  4375  2             DP_NUM : word,                          ! DATA PATTERN NUMBER SELECTED
;  4376  2             DP_ADDR,                                ! ADDR OF DATA PATTERN (LENGTH)
;  4377  2             IOB_ADDR,                               ! I/O BUFFER ADDRESS (DESTINATION)
;  4378  2             SRC_ADDR,                               ! WORKING SOURCE ADDRESS
;  4379  2             COUNT : word;                           ! NO. OF WORDS IN DATA PATTERN
;  4380  2
;  4381  3         if BIT_TST (SWP_FLAGS, SWF_UDP)             ! IF USER DEFINED A DATA PATTERN
;  4382  2         then
;  4383  2             DP_ADDR = SWP_UCNT                      ! SELECT IT
;  4384  2         else
;  4385  3             begin
;  4386  3
;  4387  3             if .SWP_DPAT neq 0                      ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;  4388  3             then
;  4389  3                 DP_NUM = .SWP_DPAT                  ! SELECT IT
;  4390  3             else
;  4391  4                 begin
;  4392  4                 DP_NUM = .DPST [.L$LUN];            ! GET PATTERN NUMBER FROM SEQUENCE TABLE
;  4393  4                 DPST [.L$LUN] = .DPST [.L$LUN] + 1; ! ADVANCE TO NEXT PATTERN NUMBER
;  4394  4
;  4395  4                 if .DPST [.L$LUN] gtru DP_CNT       ! CHECK FOR HIGH LIMIT
;  4396  4                 then
;  4397  4                     DPST [.L$LUN] = 1;
;  4398  4
;  4399  3                 end;
```

```
;  4400  3
;  4401  3              DP_ADDR = .DPA_TBL [.DP_NUM - 1];                ! ADDRESS OF DATA PATTERN (COUNT)
;  4402  3
;  4403  3              if .DP_NUM gequ 17
;  4404  3              then
;  4405  3
;  4406  3                  if .DP_NUM                                   ! CHECK MACRO (IF PATTERN 17, 19, OR 21)
;  4407  3                  then
;  4408  3                      (.DP_ADDR + 2) = .MAD1 [LBN_L]           ! LOAD LBN INTO FIRST WORD OF PATTERN
;  4409  3                  else
;  4410  3                      (.DP_ADDR + 4) = .MAD1 [LBN_L];          ! LOAD LBN INTO SECOND WORD OF PATTERN
;  4411  3
;  4412  2              end;
;  4413  2
;  4414  2          IOB_ADDR = .MAD1 [BUF_0];                            ! I/O BUFFER ADDRESS
;  4415  2          COUNT = ..DP_ADDR;                                   ! NO. OF WORDS IN DATA PATTERN
;  4416  2          SRC_ADDR = .DP_ADDR + 2;                             ! START OF THE ACTUAL DATA PATTERN
;  4417  2
;  4418  2          incr N from 1 to ((.MAD1 [BC_LO] + 1) / 2) do        ! FOR EACH WORD IN THIS WRITE REQUEST
;  4419  3              begin
;  4420  3              .IOB_ADDR = ..SRC_ADDR;                          ! MOVE 1 WORD
;  4421  3              IOB_ADDR = .IOB_ADDR + 2;                        ! ADVANCE DESTINATION ADDRESS
;  4422  3              SRC_ADDR = .SRC_ADDR + 2;                        ! ADVANCE SOURCE ADDRESS
;  4423  3              COUNT = .COUNT - 1;                              ! DECREMENT COUNT
;  4424  3
;  4425  3              if .COUNT eql 0                                  ! IF END OF DATA PATTERN
;  4426  3              then
;  4427  4                  begin
;  4428  4                  COUNT = ..DP_ADDR;                           ! REPEAT DATA PATTERN
;  4429  4                  SRC_ADDR = .DP_ADDR + 2;
;  4430  3                  end;
;  4431  3
;  4432  2              end;                                             ! WORD TRANSFER LOOP
;  4433  2
;  4434  1          end;                                                 ! ROUTINE FILL_BUFF


                                        .SBTTL  FILL.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          FILL.BUFF::
                                        JSR     R1,$SAVE5                       ;                               4347
000004  005746                          TST     -(SP)
000006  032737  000100  000000G         BIT     #100,SWP.FLAGS                  ;                               4381
000014  001403                          BEQ     1$
000016  012701  000000G                 MOV     #SWP.UCNT,R1                    ; *,DP.ADDR                     4383
000022  000443                          BR      5$                              ;                               4381
000024  013700  000000G          1$:    MOV     SWP.DPAT,R0                     ;                               4387
000030  001402                          BEQ     2$
000032  010002                          MOV     R0,R2                           ; *,DP.NUM                      4389
000034  000414                          BR      3$                              ;                               4387
000036  013700  000000G          2$:    MOV     L$LUN,R0                        ;                               4392
000042  062700  000050'                 ADD     #DPST,R0
000046  005002                          CLR     R2                              ; DP.NUM
000050  151002                          BISB    (R0),R2                         ; *,DP.NUM
```

```
000052  105210                          INCB    (R0)                            ;                                       4393
000054  121027  000025                  CMPB    (R0),#25                        ;                                       4395
000060  101402                          BLOS    3$
000062  112710  000001                  MOVB    #1,(R0)                         ;                                       4397
000066  010200                  3$:     MOV     R2,R0                           ; DP.NUM,*                              4401
000070  006300                          ASL     R0
000072  016001  001076'                  MOV     DPA.TBL-2(R0),R1                ; *,DP.ADDR
000076  020227  000021                  CMP     R2,#21                          ; DP.NUM,*                              4403
000102  103413                          BLO     5$
000104  013700  000116'                  MOV     MAD1,R0                         ;                                       4408
000110  006002                          ROR     R2                              ; DP.NUM                                4406
000112  103004                          BCC     4$
000114  016061  000046  000002          MOV     46(R0),2(R1)                    ; *,*(DP.ADDR)                          4408
000122  000403                          BR      5$                              ;                                       4406
000124  016061  000046  000004  4$:     MOV     46(R0),4(R1)                    ; *,*(DP.ADDR)                          4410
000132  013700  000116'          5$:     MOV     MAD1,R0                         ;                                       4414
000136  016004  000032                  MOV     32(R0),R4                       ; *,IOB.ADDR
000142  011103                          MOV     (R1),R3                         ; DP.ADDR,COUNT                         4415
000144  012705  000002                  MOV     #2,R5                           ;                                       4416
000150  060105                          ADD     R1,R5                           ; DP.ADDR,*
000152  010502                          MOV     R5,R2                           ; *,SRC.ADDR
000154  016046  000026                  MOV     26(R0),-(SP)                    ;                                       4418
000160  005216                          INC     (SP)
000162  012746  000002                  MOV     #2,-(SP)
000166  004737  000000G                 JSR     PC,BL$DIV
000172  010066  000004                  MOV     R0,4(SP)
000176  005000                          CLR     R0                              ; N
000200  000405                          BR      7$
000202  012224                  6$:     MOV     (R2)+,(R4)+                     ; SRC.ADDR,IOB.ADDR                     4420
000204  005303                          DEC     R3                              ; COUNT                                 4423
000206  001002                          BNE     7$                              ;                                       4425
000210  011103                          MOV     (R1),R3                         ; DP.ADDR,COUNT                         4428
000212  010502                          MOV     R5,R2                           ; *,SRC.ADDR                            4429
000214  005200                  7$:     INC     R0                              ; N                                     4418
000216  020066  000004                  CMP     R0,4(SP)                        ; N,*
000222  003767                          BLE     6$
000224  062706  000006                  ADD     #6,SP                           ;                                       4347
000230  000207                          RTS     PC

; Routine Size:  77 words,      Routine Base:  $CODE$ + 20060
; Maximum stack depth per invocation:  10 words


;   4435  1
;   4436  1
```

```
;  4437  1     GLOBAL ROUTINE PROC_RETPKT : NOVALUE =
;  4438  1
;  4439  1     !+
;  4440  1     !       THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" AND DUP_COMMAND TO CHECK FOR
;  4441  1     !       AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;  4442  1     !       PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;  4443  1     !       BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;  4444  1     !       REQUIRE PROCESSING.
;  4445  1     !
;  4446  1     !       UNDER THE MULTI-DRIVE TEST, RETURN PACKETS ORIGINATE FROM TWO SOURCES:
;  4447  1     !               1. MSCP         - THE MORE COMMON, DESCRIBING A COMPLETED I/O
;  4448  1     !                                 OPERATION.
;  4449  1     !               2. DUP          - THE LESS COMMON, DESCRIBING A PORTION OF I/O
;  4450  1     !                                 COMMUNICATIONS WITH THE CONTROLLER PROGRAM.
;  4451  1     !               3. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;  4452  1     !                                 COMMAND TIMEOUT.
;  4453  1     !-
;  4454  1
;  4455  1         while .IODQ_IN neq .IODQ_OUT do                         ! DO UNTIL I/O DONE QUEUE IS EMPTY
;  4456  2             begin
;  4457  2             RP_INDX = OUT_IODQ ();                              ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;  4458  2             RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);         ! CALCULATE RETPKT ADDRESS
;  4459  3             if NOT (.RP_ADDR [CONID] eql CID_DUP)               ! if not DUP than
;  4460  2             then (SET_CPAR (.RP_ADDR [CTLR]));;                 ! SET UP CURRENT CONTROLLER PARAMETERS
;  4461  2
;  4462  2             selectoneu .RP_ADDR [CONID] of                      ! CONNECTION ID INDICATES PACKET SOURCE
;  4463  2                 set
;  4464  2
;  4465  2                 [CID_MSCP] :        IO_RETPKT ();               ! DISK MSCP (I/O TRANSFER DONE)
;  4466  2                 [CID_DUP] :         DIO_RETPKT ();              ! DUP (I/O TRANSFER DONE)
;  4467  2                 [CID_DRIVER] :      DR_RETPKT ();               ! MESSAGE FROM "DRIVER"
;  4468  2
;  4469  2                 [otherwise] :       PRINTF (DBM12, .RP_ADDR [CONID]);!"CONN ID = XXXXX RECEIVED"
;  4470  2                 tes;
;  4471  2
;  4472  1             end;                                                ! UNITL I/O DONE QUEUE IS EMPTY


                                        .SBTTL  PROC.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                    PROC.RETPKT::
                                        MOV     R1,-(SP)                        ;                               4437
000002  023737  000000G 000000G 1$:     CMP     IODQ.IN,IODQ.OUT                ;                               4455
000010  001467                          BEQ     7$
000012  004737  000000G                 JSR     PC,OUT.IODQ                     ;                               4457
000016  010037  000000G                 MOV     R0,RP.INDX
000022  010046                          MOV     R0,-(SP)                        ; RP.INDX,*                     4458
000024  012746  000054                  MOV     #54,-(SP)
000030  004737  000000G                 JSR     PC,BL$MUL
000034  062700  000000G                 ADD     #RETPKT,R0
000040  010037  000000G                 MOV     R0,RP.ADDR
000044  126027  000003  000002          CMPB    3(R0),#2                        ;                               4459
000052  001406                          BEQ     2$
000054  116016  000002                  MOVB    2(R0),(SP)                      ;                               4460
```

```
000060  042716  177760                          BIC     #177760,(SP)
000064  004737  000000G                         JSR     PC,SET.CPAR
000070  013700  000000G             2$:         MOV     RP.ADDR,R0              ;                       4462
000074  005001                                  CLR     R1
000076  156001  000003                          BISB    3(R0),R1
000102  005701                                  TST     R1                      ;                       4465
000104  001003                                  BNE     3$
000106  004737  000000V                         JSR     PC,IO.RETPKT
000112  000424                                  BR      6$                      ;                       4462
000114  020127  000002              3$:         CMP     R1,#2                   ;                       4466
000120  001003                                  BNE     4$
000122  004737  000000V                         JSR     PC,DIO.RETPKT
000126  000416                                  BR      6$                      ;                       4462
000130  020127  000003              4$:         CMP     R1,#3                   ;                       4467
000134  001003                                  BNE     5$
000136  004737  000000V                         JSR     PC,DR.RETPKT
000142  000410                                  BR      6$                      ;                       4462
000144  010116                      5$:         MOV     R1,(SP)                 ;                       4469
000146  012746  000000G                         MOV     #DBM12,-(SP)
000152  012746  000002                          MOV     #2,-(SP)
000156  010600                                  MOV     SP,R0                   ; SP,*
000160  104417                                  TRAP    17
000162  022626                                  CMP     (SP)+,(SP)+
000164  022626                      6$:         CMP     (SP)+,(SP)+             ;                       4456
000166  000705                                  BR      1$                      ;                       4455
000170  012601                      7$:         MOV     (SP)+,R1                ;                       4437
000172  000207                                  RTS     PC

; Routine Size:  62 words,      Routine Base:  $CODE$ + 20312
; Maximum stack depth per invocation:  7 words
```

```
;  4473  1     !+
;  4474  1     GLOBAL ROUTINE DIO_RETPKT : NOVALUE =
;  4475  1
;  4476  1     !*
;  4477  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
;  4478  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;  4479  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS.
;  4480  1     !
;  4481  1     !       IMPLICIT INPUTS:
;  4482  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  4483  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  4484  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;  4485  1     !               DUOFF - CST OFFSET FOR THE CURRENT UNIT
;  4486  1     !               L$LUN - CURRENT UNIT NUMBER
;  4487  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  4488  1     !
;  4489  1     !       IMPLICIT OUTPUTS
;  4490  1     !               CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] - IF THIS BIT SET NO DUP EXERCISER
;  4491  1     !
;  4492  1     !-
;  4493  2     BEGIN
;  4494  2
;  4495  2     LOCAL   FLAG : BYTE INITIAL(BYTE(TRUE)),
;  4496  2             SUM2 : WORD,
;  4497  2             SUM : WORD;                                    ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;  4498  2     !PRINTX (DER18);
;  4499  2
;  4500  2     IF .RP_ADDR [STATUS] NEQU ST_SUC                              ! IF STATUS CODE INDICATES ERROR
;  4501  2     THEN
;  4502  3             BEGIN
;  4503  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;             ! SET DUP ERROR FLAG
;  4504  3             HARD_ERROR ();
;  4505  3             IF .RP_ADDR [ENDCOD] EQLU (OP_ELP + OP_END) OR  ! IF ENDCODE IS EXECUTE LOCAL PROGRAM
;  4506  4                .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)     !            OR GET DUST STATUS
;  4507  4             THEN    BEGIN
;  4508  4                     CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;           ! TURN OFF DUP EXERCISR
;  4509  3                     END;
;  4510  3             END
;  4511  2     ELSE                                            ! ELSE - I/O WAS SUCCESSFUL
;  4512  3         BEGIN
;  4513  3
;  4514  4         IF .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) ! IF ENDCODE IS GET DUST STATUS
;  4515  3         THEN
;  4516  4             BEGIN
;  4517  4             IF .RP_ADDR [9,11,1,0] EQL 1
;  4518  4                     THEN CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = ACTIVE      ! CONTROLLER IN AN ACTIVE STAE
;  4519  4                     ELSE CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = IDLE;       ! CONTROLLER IN AN IDLE STATE
;  4520  4             IF .RP_ADDR [9,9,1,0] NEQ 1 THEN        ! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
;  4521  5                 BEGIN
;  4522  5                     HARD_ERROR ();
;  4523  5                     CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;           ! TURN OFF DUP EXERCISR
;  4524  4                     END;
;  4525  3             END;
```

```
;   4526  3
;   4527  3
;   4528  3     IF (.RP_ADDR [ENDCOD] EQL (OP_RCD + OP_END)) AND
;   4529  3        (.DUPPKT [DUPTYPE] EQL 6) AND
;   4530  3        (.DUPPKT [DUPMESG] EQL 2) AND                !IF IT IS A RECEIVE DBN COMMAND WITH TYPE 6 AND MESSAGE 2 THEN
;   4531  4        (.CST_ADDR [.DUOFF + OF_DBN, DUPWRITE] EQLU 1)      ! IF WRITE FLAG SET IN CST TABLE THEN COMPARE BLOCKS
;   4532  3             THEN DUP_COMPARE ();
;   4533  3
;   4534  2     END;                                           ! COMPARE THE FOLLOWING 512 BYTES
;   4535  2
;   4536  2     PUT_RETPKT (.RP_INDX);
;   4537  1     END;                                           ! ROUTINE DIO_RETPKT


                                    .SBTTL  DIO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                  DIO.RETPKT::
                                    MOV     R1,-(SP)                        ;                                       4474
000002  112700  000001              MOVB    #1,R0                           ; *,FLAG                                4493
000006  013701  000000G             MOV     RP.ADDR,R1                      ;                                       4500
000012  005761  000016              TST     16(R1)
000016  001435                      BEQ     2$
000020  013700  001162'              MOV     DUOFF,R0                        ;                                       4503
000024  006300                      ASL     R0
000026  063700  000000G             ADD     CST.ADDR,R0
000032  052760  040000  000020      BIS     #40000,20(R0)
000040  004737  000000V             JSR     PC,HARD.ERROR                   ;                                       4504
000044  013700  000000G             MOV     RP.ADDR,R0                      ;                                       4505
000050  126027  000014  000203      CMPB    14(R0),#203
000056  001404                      BEQ     1$
000060  126027  000014  000201      CMPB    14(R0),#201                     ;                                       4506
000066  001112                      BNE     6$
000070  013700  001162'      1$:     MOV     DUOFF,R0                        ;                                       4508
000074  006300                      ASL     R0
000076  063700  000000G             ADD     CST.ADDR,R0
000102  052760  100000  000020      BIS     #100000,20(R0)
000110  000501                      BR      6$                              ;                                       4500
000112  126127  000014  000201  2$: CMPB    14(R1),#201                     ;                                       4514
000120  001036                      BNE     5$
000122  013700  001162'              MOV     DUOFF,R0                        ;                                       4518
000126  006300                      ASL     R0
000130  063700  000000G             ADD     CST.ADDR,R0
000134  032761  004000  000022      BIT     #4000,22(R1)                    ;                                       4517
000142  001404                      BEQ     3$
000144  052760  020000  000020      BIS     #20000,20(R0)                   ;                                       4518
000152  000403                      BR      4$                              ;                                       4517
000154  042760  020000  000020  3$: BIC     #20000,20(R0)                   ;                                       4519
000162  032761  001000  000022  4$: BIT     #1000,22(R1)                    ;                                       4520
000170  001012                      BNE     5$
000172  004737  000000V             JSR     PC,HARD.ERROR                   ;                                       4522
000176  013700  001162'              MOV     DUOFF,R0                        ;                                       4523
000202  006300                      ASL     R0
000204  063700  000000G             ADD     CST.ADDR,R0
000210  052760  100000  000020      BIS     #100000,20(R0)
```

```
000216  013700  000000G                 5$:     MOV     RP.ADDR,R0                      ;                       4528
000222  126027  000014  000205                  CMPB    14(R0),#205
000230  001031                                  BNE     6$
000232  013700  000000G                         MOV     DUPPKT,R0                       ;                       4529
000236  042700  007777                          BIC     #7777,R0
000242  020027  060000                          CMP     R0,#60000
000246  001022                                  BNE     6$
000250  013700  000000G                         MOV     DUPPKT,R0                       ;                       4530
000254  042700  170000                          BIC     #170000,R0
000260  020027  000002                          CMP     R0,#2
000264  001013                                  BNE     6$
000266  013700  001162'                         MOV     DUOFF,R0                        ;                       4531
000272  006300                                  ASL     R0
000274  063700  000000G                         ADD     CST.ADDR,R0
000300  032760  010000  000020                  BIT     #10000,20(R0)
000306  001402                                  BEQ     6$
000310  004737  000000V                         JSR     PC,DUP.COMPARE                  ;                       4532
000314  013746  000000G                 6$:     MOV     RP.INDX,-(SP)                   ;                       4536
000320  004737  000000G                         JSR     PC,PUT.RETPKT
000324  005726                                  TST     (SP)+                           ;                       4493
000326  012601                                  MOV     (SP)+,R1                        ;                       4474
000330  000207                                  RTS     PC

; Routine Size:  109 words,    Routine Base:  $CODE$ + 20506
; Maximum stack depth per invocation:  3 words


;   4538  1
```

```
;    4539  1     GLOBAL ROUTINE DUP_COMPARE : NOVALUE =
;    4540  1
;    4541  1
;    4542  1     !*
;    4543  1     !       THIS ROUTINE IS CALLED BY DIO_RETPKT WHEN THE RECEIVE DATA COMMAND
;    4544  1     !       IS BEING PROCESSED. THIS COMMAND COMPARES THE WRITTEN BUFFER WITH
;    4545  1     !       THE PATERN WORD GIVEN IN SEND DATA COMMAND.  FOR EVERY WORD COMPARED
;    4546  1     !       THE ROUTINE INCREMENTS THE TALLY TABLE.  IF THE COMPARE SHOWS AN
;    4547  1     !       ERROR. THE DBN HARD ERROR COUNTER WILL BE INCREMENTED AND THE
;    4548  1     !       THE DBN NUMBER AND BYTE COUNT WILL BE PRINTED.
;    4549  1     !
;    4550  1     !       IMPLICIT INPUTS:
;    4551  1     !               S_PATTERN       ! THE SAVED PATTERN WRITTEN TO THE DBN'S
;    4552  1     !               S_DUPPKT        ! THE POINTER FOR DUP BUFFER
;    4553  1     !               T_ADDR          ! THE ADDRESS OF THE TALLY TABLE FOR THIS UNIT
;    4554  1     !               CST_ADDR        ! THE ADDRESS OF PRESENT CONTROLLER STATUS TABLE
;    4555  1     !-
;    4556  2     BEGIN
;    4557  2
;    4558  2     OWN
;    4559  2             COUNT : WORD;
;    4560  2
;    4561  2     !PRINTX (DER19);
;    4562  2     S_DUPPKT = 0;
;    4563  2     INCR COUNT FROM 1 TO 256 DO     !INDEX PIONTER FOR DATA STORED IN MSCP ENV PACKET
;    4564  3             BEGIN
;    4565  3             S_DUPPKT = .S_DUPPKT + 2;                          ! INITIALLY THIS SKIPS THE FIRST WORD OF DUPPKT
;    4566  3             IF .(DUPPKT + .S_DUPPKT) NEQ .S_PATTERN THEN       !IF THE CONTENTS OF DBN DOESN'T EQUAL PATTERN
;    4567  4                     BEGIN
;    4568  4                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;          ! SET DUP ERROR FLAG
;    4569  4                     ERRHRD (46, EH_10, EMS_22);                !LIST ERROR
;    4570  4                     EXITLOOP;
;    4571  3                     END;
;    4572  2             END;                                       !GO THROUGH ALL DBN WORDS
;    4573  1     END;                                               !END ROUTINE DUP-COMPARE


001212                                  .PSECT  $GGG$,  RO
001212                          COUNT:  .BLKW   1


                                        .SBTTL  DUP.COMPARE MULTI-DRIVE TEST ROUTINES
021040                                  .PSECT  $CODE$,  RO

000000  010146                  DUP.COMPARE::
                                        MOV     R1,-(SP)                        ;                               4539
000002  005037  000000G                 CLR     S.DUPPKT                        ;                               4562
000006  012701  000400                  MOV     #400,R1                         ; *,COUNT                       4563
000012  062737  000002  000000G 1$:     ADD     #2,S.DUPPKT                     ;                               4565
000020  013700  000000G                 MOV     S.DUPPKT,R0                     ;                               4566
000024  026037  000000G 000000G         CMP     DUPPKT(R0),S.PATTERN
000032  001415                          BEQ     2$
```

```
000034  013700  001162'                        MOV     DUOFF,R0                        ;                       4568
000040  006300                                  ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  052760  040000  000020                 BIS     #40000,20(R0)
000054  104456                                  TRAP    56                              ;                       4569
000056  000056                                  .WORD   56
000060  000000G                                .WORD   EH.10
000062  000000G                                .WORD   EMS.22
000064  000402                                  BR      3$                              ;                       4567
000066  005301                          2$:     DEC     R1                              ; COUNT                 4563
000070  001350                                  BNE     1$
000072  012601                          3$:     MOV     (SP)+,R1                        ;                       4539
000074  000207                                  RTS     PC

; Routine Size:  31 words,      Routine Base:  $CODE$ + 21040
; Maximum stack depth per invocation:  3 words


;   4574  1
;   4575  1
;   4576  1
```

```
;    4577  1     GLOBAL routine IO_RETPKT : novalue =
;    4578  1
;    4579  1     !+
;    4580  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;    4581  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;    4582  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;    4583  1     !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;    4584  1     !
;    4585  1     !       IMPLICIT INPUTS:
;    4586  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    4587  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    4588  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4589  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    4590  1     !               L$LUN - CURRENT UNIT NUMBER
;    4591  1     !-
;    4592  1
;    4593  2         begin
;    4594  2
;    4595  2         local
;    4596  2             FLAG : byte initial (byte (TRUE));
;    4597  2
;    4598  2         FSET_UPAR ();                           ! FIND UNIT'S ENTRY IS CST AND SET UP UNIT-RELATED DATA
;    4599  2         ST_CODE = .RP_ADDR [STSCOD];            ! GET STATUS CODE FROM RETPKT
;    4600  2         SB_CODE = .RP_ADDR [SUBCOD];            ! GET SUB-CODE, IF ANY
;    4601  2
;    4602  3         if (.ST_CODE neq ST_SUC)                ! IF STATUS CODE INDICATES ERROR
;    4603  2         then
;    4604  3             begin
;    4605  3             HARD_ERROR ();                      ! UPDATE ERROR COUNT
;    4606  3             COMPARE_DATA = FALSE;               ! NO POINT IN DOING HOST COMPARES ON ERRORS
;    4607  3
;    4608  3             if (.ST_CODE neq ST_OFL) and        ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
;    4609  3                (.ST_CODE neq ST_AVL) and
;    4610  4                (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
;    4611  3             then
;    4612  4                 begin
;    4613  4                 DUR [.L$LUN] = DU_HERR;         ! LOAD REASON FOR DROPPING UNIT
;    4614  4                 DODU (.L$LUN);                  ! DROP UNIT
;    4615  3                 end;
;    4616  3
;    4617  3             end
;    4618  2         else                                    ! IF I/O WAS SUCCESSFUL
;    4619  3             begin
;    4620  3             UPD_IO_TALLY ();                    ! UPDATE I/O TALLY (STATISTICS)
;    4621  3
;    4622  4             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
;    4623  3             then
;    4624  3                 COMPARE_DATA = TRUE;            ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS
;    4625  3
;    4626  3             if (BIT_TST (SWP_FLAGS, SWF_HWC)) and   ! IF HOST IS DOING WRITE-COMPARES
;    4627  4                (.COMPARE_DATA)
;    4628  3             then
;    4629  3                 FLAG = HOST_WRT_CHK ();         ! SAVE I/O PACKET OR DO WRITE-CHECK
```

```
;   4630  3
;   4631  2              end;
;   4632  2
;   4633  2          if .FLAG                                ! IF FLAG IS STILL TRUE
;   4634  2          then
;   4635  2              SWEEP ();                           ! DEALLOCATE BUFFER(S) AND RETPKT(S)
;   4636  2
;   4637  2          QIO [.CCTLR] = .QIO [.CCTLR] - 1;       ! DECREMENT NO. OF OUTSTANDING QIOs
;   4638  1          end;                                    ! ROUTINE IO_RETPKT


                                        .SBTTL  IO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          IO.RETPKT::
                                        JSR     R1,$SAVE2                       ;                       4577
000004  112701  000001                  MOVB    #1,R1                           ; *,FLAG                4593
000010  004737  000000V                  JSR     PC,FSET.UPAR                    ;                       4598
000014  013700  000000G                  MOV     RP.ADDR,R0                      ;                       4599
000020  116037  000016  000000G         MOVB    16(R0),ST.CODE
000026  042737  177740  000000G         BIC     #177740,ST.CODE
000034  016002  000016                  MOV     16(R0),R2                       ;                       4600
000040  006202                          ASR     R2
000042  006202                          ASR     R2
000044  006202                          ASR     R2
000046  006202                          ASR     R2
000050  006202                          ASR     R2
000052  042702  174000                  BIC     #174000,R2
000056  010237  000000G                 MOV     R2,SB.CODE
000062  005737  000000G                 TST     ST.CODE                         ;                       4602
000066  001431                          BEQ     1$
000070  004737  000000V                  JSR     PC,HARD.ERROR                   ;                       4605
000074  105037  001174'                 CLRB    COMPARE.DATA                    ;                       4606
000100  023727  000000G  000003         CMP     ST.CODE,#3                      ;                       4608
000106  001447                          BEQ     3$
000110  023727  000000G  000004         CMP     ST.CODE,#4                      ;                       4609
000116  001443                          BEQ     3$
000120  013700  000000G                 MOV     T.ADDR,R0                       ;                       4610
000124  026037  000014  000000G         CMP     14(R0),SWP.ERROR
000132  103435                          BLO     3$
000134  013700  000000G                 MOV     L$LUN,R0                        ;                       4613
000140  112760  000004  000000G         MOVB    #4,DUR(R0)
000146  104451                          TRAP    51                              ;                       4614
000150  000426                          BR      3$                              ;                       4602
000152  004737  000000V          1$:     JSR     PC,UPD.IO.TALLY                 ;                       4620
000156  013700  000000G                 MOV     RP.ADDR,R0                      ;                       4622
000162  126027  000014  000242         CMPB    14(R0),#242
000170  001003                          BNE     2$
000172  112737  000001  001174'         MOVB    #1,COMPARE.DATA                 ;                       4624
000200  032737  000040  000000G  2$:     BIT     #40,SWP.FLAGS                   ;                       4626
000206  001407                          BEQ     3$
000210  032737  000001  001174'         BIT     #1,COMPARE.DATA                 ;                       4627
000216  001403                          BEQ     3$
000220  004737  000000V                  JSR     PC,HOST.WRT.CHK                 ;                       4629
000224  110001                          MOVB    R0,R1                           ; *,FLAG
```

```
000226  006001                          3$:     ROR     R1                      ; FLAG                          4633
000230  103002                                  BCC     4$
000232  004737  000000V                        JSR     PC,SWEEP                ;                               4635
000236  013700  000000G                4$:     MOV     CCTLR,R0                ;                               4637
000242  105360  000000G                        DECB    QIO(R0)
000246  000207                                RTS     PC                      ;                               4577

; Routine Size:  84 words,      Routine Base:  $CODE$ + 21136
; Maximum stack depth per invocation:  5 words
```

```
;    4639  1     GLOBAL routine FSET_UPAR : novalue =
;    4640  1
;    4641  1     !+
;    4642  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;    4643  1     !       CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;    4644  1     !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;    4645  1     !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;    4646  1     !       DATA PARAMETERS.
;    4647  1     !
;    4648  1     !       IMPLICIT INPUTS:
;    4649  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    4650  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    4651  1     !-
;    4652  1
;    4653  2         begin
;    4654  2
;    4655  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    4656  2
;    4657  2             if .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]        ! IF RETPKT UNIT MATCHES CST ENTRY
;    4658  2             then
;    4659  3                 begin
;    4660  3                 SET_UPAR (.OFFSET);                                                ! SET UP UNIT-RELATED DATA
;    4661  3                 return;                                                            ! DONE
;    4662  2                 end;
;    4663  2
;    4664  2         PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                   ! "CAN'T FIND DISK XXX IN CST X"
;    4665  1         end;                                                                       ! ROUTINE FSET_UPAR


                                .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G   FSET.UPAR::
                            JSR     R1,$SAVE4                       ;                       4639
000004  012702  000003              MOV     #3,R2                   ; *,OFFSET              4655
000010  010201      1$:     MOV     R2,R1                           ; OFFSET,*              4657
000012  006301              ASL     R1
000014  063701  000000G             ADD     CST.ADDR,R1
000020  013700  000000G             MOV     RP.ADDR,R0
000024  016004  000010              MOV     10(R0),R4
000030  111103                      MOVB    (R1),R3
000032  042703  177760              BIC     #177760,R3
000036  020304                      CMP     R3,R4
000040  001005                      BNE     2$
000042  010246                      MOV     R2,-(SP)                ; OFFSET,*              4660
000044  004737  000000G             JSR     PC,SET.UPAR
000050  005726                      TST     (SP)+                   ;                       4661
000052  000207                      RTS     PC                      ;                       4659
000054  062702  000012      2$:     ADD     #12,R2                  ; *,OFFSET              4655
000060  020227  000041              CMP     R2,#41                  ; OFFSET,*
000064  003751                      BLE     1$
000066  013746  000000G             MOV     CCTLR,-(SP)             ;                       4664
000072  013700  000000G             MOV     RP.ADDR,R0
000076  016046  000010              MOV     10(R0),-(SP)
000102  012746  000000G             MOV     #DBM19,-(SP)
```

```
000106  012746  000003                        MOV     #3,-(SP)
000112  010600                                MOV     SP,R0                   ; SP,R0
000114  104417                                TRAP    17
000116  062706  000010                        ADD     #10,SP                  ;                       4653
000122  000207                                RTS     PC                      ;                       4639

; Routine Size:  42 words,      Routine Base:  $CODE$ + 21406
; Maximum stack depth per invocation:  11 words
```

```
;    4666  1     GLOBAL routine HARD_ERROR : novalue =
;    4667  1
;    4668  1     !+
;    4669  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO INCREMENT THE HARD
;    4670  1     !       ERROR STATISTIC FIELD FOR THE CURRENT UNIT. IF THE HARD ERROR COUNT
;    4671  1     !       HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS DROPPED
;    4672  1     !       FROM TESTING.
;    4673  1     !
;    4674  1     !       IMPLICIT INPUTS:
;    4675  1     !               L$LUN - CURRENT UNIT NUMBER
;    4676  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    4677  1     !               CUOFF - CST OFFSET FOR CURRENT UNIT
;    4678  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4679  1     !-
;    4680  1
;    4681  2         begin
;    4682  2         T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;                     ! INCREMENT UNIT'S HARD ERROR COUNT
;    4683  2     if .RP_ADDR [CONID] EQL CID_MSCP                                    !FOR MSCP ERRORS   ZZZ
;    4684  2     THEN                                                                !                  ZZZ
;    4685  2
;    4686  2         selectoneu .ST_CODE of
;    4687  2             set
;    4688  2
;    4689  2             [ST_SUC]:       if .SB_CODE neq 0                           ! SUCCESS WITH NON-ZERO SUB-CODE
;    4690  2                             then
;    4691  3                                 begin
;    4692  3
;    4693  3                                 if .SB_CODE eql 4
;    4694  3                                 then
;    4695  3                                     T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1
;    4696  3                                 else
;    4697  3                                     T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    4698  3
;    4699  3                                 if .APT_MODE
;    4700  3                                 then
;    4701  3                                     ERR_HRD_RTNE_APT (44)
;    4702  3                                 else
;    4703  3                                     ERR_HRD_RTNE (44);
;    4704  3
;    4705  2                                 end;
;    4706  2
;    4707  3                     [ST_CMD]:   begin                                   ! INVALID COMMAND
;    4708  3                     T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    4709  3
;    4710  3                     if .APT_MODE
;    4711  3                     then
;    4712  3                         ERR_HRD_RTNE_APT (31)
;    4713  3                     else
;    4714  3                         ERR_HRD_RTNE (31);
;    4715  3
;    4716  2                     end;
;    4717  2
;    4718  3                     [ST_ABO]:   begin                                   ! COMMMAND ABORTED
```

```
;  4719  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4720  3
;  4721  3                        if .APT_MODE
;  4722  3                        then
;  4723  3                            ERR_HRD_RTNE_APT (32)
;  4724  3                        else
;  4725  3                            ERR_HRD_RTNE (32);
;  4726  3
;  4727  2                        end;
;  4728  2
;  4729  3                [ST_OFL] :      begin                                   ! OFFLINE
;  4730  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4731  3
;  4732  3                        if .APT_MODE
;  4733  3                        then
;  4734  4                            begin
;  4735  4                            .MAIL_BOX_TESTNUM = 1;
;  4736  4                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  4737  3                            end;
;  4738  3
;  4739  3                        ERRDF (18, EGD_18, EMS_18);                     !
;  4740  3                        DUR [.L$LUN] = DU_DFATAL;                       ! DEVICE FATAL ERROR
;  4741  3                        DODU (.L$LUN);                                  ! DROP UNIT
;  4742  2                        end;
;  4743  2
;  4744  3                [ST_AVL]:       begin                                   ! WENT TO AVAILABLE STATE
;  4745  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4746  3
;  4747  3                        if .APT_MODE
;  4748  3                        then
;  4749  4                            begin
;  4750  4                            .MAIL_BOX_TESTNUM = 1;
;  4751  4                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  4752  3                            end;
;  4753  3
;  4754  3                        ERRDF (24, EGD_18, EMS_24);                     !
;  4755  3                        DUR [.L$LUN] = DU_DFATAL;                       ! DEVICE FATAL ERROR
;  4756  3                        DODU (.L$LUN);                                  ! DROP UNIT
;  4757  2                        end;
;  4758  2
;  4759  3                [ST_MFE]:       begin                                   ! MEDIA FORMAT ERROR
;  4760  3                        T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1;
;  4761  3
;  4762  3                        if .APT_MODE
;  4763  3                        then
;  4764  3                            ERR_HRD_RTNE_APT (35)
;  4765  3                        else
;  4766  3                            ERR_HRD_RTNE (35);
;  4767  3
;  4768  2                        end;
;  4769  2
;  4770  3                [ST_WPT]:       begin                                   ! DEVICE WRITE PROTECTED
;  4771  3
```

```
;  4772  3                        if .SB_CODE eql 128
;  4773  3                        then
;  4774  3                            T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1
;  4775  3                        else
;  4776  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4777  3
;  4778  3                        if .APT_MODE
;  4779  3                        then
;  4780  3                            ERR_HRD_RTNE_APT (36)
;  4781  3                        else
;  4782  3                            ERR_HRD_RTNE (36);
;  4783  3
;  4784  2                        end;
;  4785  2
;  4786  3            [ST_CMP]:   begin                                          ! COMPARE ERROR
;  4787  3                        T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  4788  3
;  4789  3                        if .APT_MODE
;  4790  3                        then
;  4791  3                            ERR_HRD_RTNE_APT (37)
;  4792  3                        else
;  4793  3                            ERR_HRD_RTNE (37);
;  4794  3
;  4795  2                        end;
;  4796  2
;  4797  3            [ST_DAT]:   begin                                          ! DATA ERROR
;  4798  3
;  4799  3                        if .SB_CODE eql 2
;  4800  3                        then
;  4801  3                            T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  4802  3                        else
;  4803  3                            T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  4804  3
;  4805  3                        if (.SB_CODE eql 0) and
;  4806  3                           (not .FORCED_ERROR) and
;  4807  4                           (BIT_TST (SWP_FLAGS, SWF_FER))
;  4808  3                        then
;  4809  4                            begin
;  4810  4                            FORCED_ERROR = TRUE;                       ! BLOCK WITH "FORCED ERROR" FOUND
;  4811  4                            FER_LBN = .RP_ADDR [LBN_LO];               !
;  4812  4                            FER_BC = .RP_ADDR [CBCNT_LO];              !
;  4813  3                            end;
;  4814  3
;  4815  3                        if .APT_MODE
;  4816  3                        then
;  4817  3                            ERR_HRD_RTNE_APT (38)
;  4818  3                        else
;  4819  3                            ERR_HRD_RTNE (38);
;  4820  3
;  4821  2                        end;
;  4822  2
;  4823  3            [ST_HST]:   begin                                          ! HOST ACCESS ERROR
;  4824  3                        T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
```

```
;  4825  3
;  4826  3                        if .APT_MODE
;  4827  3                        then
;  4828  3                            ERR_HRD_RTNE_APT (39)
;  4829  3                        else
;  4830  3                            ERR_HRD_RTNE (39);
;  4831  3
;  4832  2                        end;
;  4833  2
;  4834  3            [ST_CNT]:   begin                                   ! CONTROLLER ERROR
;  4835  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4836  3
;  4837  3                        if .APT_MODE
;  4838  3                        then
;  4839  3                            ERR_HRD_RTNE_APT (40)
;  4840  3                        else
;  4841  3                            ERR_HRD_RTNE (40);
;  4842  3
;  4843  2                        end;
;  4844  2
;  4845  3            [ST_DRV]:   begin                                   ! DRIVE ERROR
;  4846  3
;  4847  3                        if .SB_CODE eql 3
;  4848  3                        then
;  4849  3                            T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  4850  3                        else
;  4851  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4852  3
;  4853  3                        if .APT_MODE
;  4854  3                        then
;  4855  3                            ERR_HRD_RTNE_APT (41)
;  4856  3                        else
;  4857  3                            ERR_HRD_RTNE (41);
;  4858  3
;  4859  2                        end;
;  4860  2
;  4861  3            [ST_DIA]:   begin                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  4862  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4863  3
;  4864  3                        if .APT_MODE
;  4865  3                        then
;  4866  3                            ERR_HRD_RTNE_APT (43)
;  4867  3                        else
;  4868  3                            ERR_HRD_RTNE (43);
;  4869  3
;  4870  2                        end;
;  4871  2
;  4872  3            [otherwise]: begin                                  ! PRINT STATUS CODE IF NO MATCH
;  4873  3                        C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  4874  3
;  4875  3                        if .APT_MODE
;  4876  3                        then
;  4877  3                            ERR_HRD_RTNE_APT (45)
```

```
;    4878  3                          else
;    4879  3                              ERR_HRD_RTNE (45);
;    4880  3
;    4881  2                          end;
;    4882  2
;    4883  2              tes;
;    4884  2
;    4885  2
;    4886  2          if .RP_ADDR [CONID] EQL CID_DUP                                    !FOR DUP ERRORS        ZZZ
;    4887  2          THEN                                                               !                      ZZZ
;    4888  2
;    4889  2              selectoneu .RP_ADDR [STSCOD] of
;    4890  2                      SET
;    4891  3                      [%o'0']             : begin                    ! if status code succesful
;    4892  3                                            if .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) and ! IF ENDCODE IS GET DUST STATUS
;    4893  3                                               .RP_ADDR [9,9,1,0] NEQ 1                     ! TEST TO SEE IF CONTROLLER LOCAL PR
OGRAMS
;    4894  3                                                                                            ! (PG 18 OF DUP DOC)
;    4895  4                                            then
;    4896  4                                                    BEGIN
;    4897  4                                                    ERR_HRD_RTNE (60);          !UNABLE TO LOAD LOCAL CONTROLLER DUP
 MEDIA !ZZZ
;    4898  4                                                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    4899  3                                                    END
;    4900  4                                            else
;    4901  5                                                    begin
;    4902  4                                                    if (.DUPPKT [DUPTYPE] eql 5)    ! if fatal error
;    4903  5                                                    then
;    4904  5                                                            begin
;    4905  5                                                            DUR [.L$LUN] = DU_DFATAL;
;    4906  4                                                            DODU (.L$LUN);                  ! FATAL DEVICE ERROR DROP UNIT);
;    4907  4                                                            end;                            ! SET REASON FOR DROPPING UNIT
;    4908  4                                                    selectoneu .DUPPKT [DUPMESG] of
;    4909  5                                                            SET
;    4910  5                                                            [%o'1'] : begin
;    4911  5                                                                      ERR_HRD_RTNE (62);        ! illegal unit number    !ZZZ
;    4912  4                                                                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    4913  5                                                                      end;
;    4914  5                                                            [%o'2'] : begin
;    4915  5                                                                      ERR_HRD_RTNE (63);        ! illegal relative or physical b
lock # !ZZZ
;    4916  4                                                                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    4917  5                                                                      end;
;    4918  5                                                            [%o'3'] : begin
;    4919  5                                                                      ERR_HRD_RTNE (64);        ! device error    !ZZZ
;    4920  4                                                                      T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    4921  5                                                                      end;
;    4922  5                                                            [%o'4'] : begin
;    4923  5                                                                      ERR_HRD_RTNE (65);        ! zero lenght message   !ZZZ
;    4924  4                                                                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    4925  5                                                                      end;
;    4926  5                                                            [OTHERWISE] : begin
;    4927  5                                                                      ERR_HRD_RTNE (66);      ! DUP UNKNOWN STATUS CODE   !ZZZ
;                                                                                 C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HR
D] + 1;
;    4928  4                                                                      end;
;    4929  4                                                            tes;
;    4930  3                          end;
```

```
;   4931  3
;   4932  2                                  end;
;   4933  3                  [%o'1']        : begin
;   4934  3                                ERR_HRD_RTNE (67);       ! INVALID COMMAND            !ZZZ
;   4935  3                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;   4936  2                                end;
;   4937  3                  [%o'2']        : begin
;   4938  3                                ERR_HRD_RTNE (68);       ! NO REGION AVAILABLE        !ZZZ
;   4939  3                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;   4940  2                                end;
;   4941  3                  [%o'3']        : begin
;   4942  3                                ERR_HRD_RTNE (69);                ! NO REGION SUITABLE         !ZZZ
;   4943  3                                T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;   4944  2                                end;
;   4945  3                  [%o'4']        : begin
;   4946  3                                ERR_HRD_RTNE (70);       ! PROGRAM NOT KNOWN          !ZZZ
;   4947  3                                T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;   4948  2                                end;
;   4949  3                  [%o'5']        : begin
;   4950  3                                ERR_HRD_RTNE (71);                ! LOAD FAILURE               !ZZZ
;   4951  3                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;   4952  2                                end;
;   4953  3                  [%o'6']        : begin
;   4954  3                                ERR_HRD_RTNE (72);                ! STANDALONE                 !ZZZ
;   4955  3                                T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;   4956  2                                end;
;   4957  3                  [OTHERWISE] : begin
;   4958  3                                ERR_HRD_RTNE (73);                ! DUP UNKNOWN STATUS CODE   !ZZZ
;   4959  3                                C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;   4960  2                                end;
;   4961  2                  TES;
;   4962  2
;   4963  1          end;                                                                    ! ROUTINE HARD_ERROR


                                                .SBTTL  HARD.ERROR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                  HARD.ERROR::
                                                JSR     R1,$SAVE4                       ;                                4666
000004  013701  000000G                         MOV     T.ADDR,R1                       ;                                4682
000010  005261  000014                          INC     14(R1)
000014  013703  000000G                         MOV     RP.ADDR,R3                      ;                                4683
000020  105763  000003                          TSTB    3(R3)
000024  001171                                  BNE     12$
000026  013702  000000G                         MOV     ST.CODE,R2                      ;                                4686
000032  001027                                  BNE     4$                              ;                                4689
000034  013704  000000G                         MOV     SB.CODE,R4
000040  001563                                  BEQ     12$
000042  012700  000050                          MOV     #50,R0                          ;                                4695
000046  060100                                  ADD     R1,R0
000050  020427  000004                          CMP     R4,#4                           ;                                4693
000054  001002                                  BNE     1$
000056  105210                                  INCB    (R0)                            ;                                4695
000060  000402                                  BR      2$                              ;                                4693
```

```
000062  105260  000001                  1$:     INCB    1(R0)                           ;                       4697
000066  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4699
000074  001403                                  BEQ     3$
000076  012746  000054                          MOV     #54,-(SP)                       ;                       4701
000102  000557                                  BR      14$
000104  012746  000054                  3$:     MOV     #54,-(SP)                       ;                       4703
000110  000557                                  BR      16$
000112  020227  000001                  4$:     CMP     R2,#1                           ;                       4707
000116  001014                                  BNE     6$
000120  105261  000051                          INCB    51(R1)                          ;                       4708
000124  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4710
000132  001403                                  BEQ     5$
000134  012746  000037                          MOV     #37,-(SP)                       ;                       4712
000140  000570                                  BR      20$
000142  012746  000037                  5$:     MOV     #37,-(SP)                       ;                       4714
000146  000570                                  BR      22$
000150  020227  000002                  6$:     CMP     R2,#2                           ;                       4718
000154  001014                                  BNE     8$
000156  105261  000050                          INCB    50(R1)                          ;                       4719
000162  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4721
000170  001403                                  BEQ     7$
000172  012746  000040                          MOV     #40,-(SP)                       ;                       4723
000176  000570                                  BR      24$
000200  012746  000040                  7$:     MOV     #40,-(SP)                       ;                       4725
000204  000570                                  BR      26$
000206  020227  000003                  8$:     CMP     R2,#3                           ;                       4729
000212  001036                                  BNE     10$
000214  105261  000050                          INCB    50(R1)                          ;                       4730
000220  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4732
000226  001415                                  BEQ     9$
000230  012777  000001  001170'                MOV     #1,@MAIL.BOX.TESTNUM            ;                       4735
000236  013700  000000G                         MOV     CUOFF,R0                        ;                       4736
000242  006300                                  ASL     R0
000244  063700  000000G                         ADD     CST.ADDR,R0
000250  111077  001172'                        MOVB    (R0),@MAIL.BOX.SUBTST
000254  042777  177760  001172'                BIC     #177760,@MAIL.BOX.SUBTST
000262  104455                          9$:     TRAP    55                              ;                       4739
000264  000022                                  .WORD   22
000266  000000G                                 .WORD   EGD.18
000270  000000G                                 .WORD   EMS.18
000272  013700  000000G                         MOV     L$LUN,R0                        ;                       4740
000276  112760  000005  000000G                MOVB    #5,DUR(R0)
000304  104451                                  TRAP    51                              ;                       4741
000306  000440                                  BR      12$                             ;                       4686
000310  020227  000004                  10$:    CMP     R2,#4                           ;                       4744
000314  001037                                  BNE     13$
000316  105261  000050                          INCB    50(R1)                          ;                       4745
000322  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4747
000330  001415                                  BEQ     11$
000332  012777  000001  001170'                MOV     #1,@MAIL.BOX.TESTNUM            ;                       4750
000340  013700  000000G                         MOV     CUOFF,R0                        ;                       4751
000344  006300                                  ASL     R0
000346  063700  000000G                         ADD     CST.ADDR,R0
```

```
000352  111077  001172'                        MOVB    (R0),@MAIL.BOX.SUBTST
000356  042777  177760  001172'                BIC     #177760,@MAIL.BOX.SUBTST
000364  104455                          11$:    TRAP    55                              ;                       4754
000366  000030                                  .WORD   30
000370  000000G                                 .WORD   EGD.18
000372  000000G                                 .WORD   EMS.24
000374  013700  000000G                         MOV     L$LUN,R0                        ;                       4755
000400  112760  000005  000000G                MOVB    #5,DUR(R0)
000406  104451                                  TRAP    51                              ;                       4756
000410  000137  022726'                  12$:    JMP     51$                             ;                       4686
000414  020227  000005                    13$:    CMP     R2,#5                           ;                       4759
000420  001014                                  BNE     17$
000422  105261  000046                          INCB    46(R1)                          ;                       4760
000426  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4762
000434  001403                                  BEQ     15$
000436  012746  000043                          MOV     #43,-(SP)                       ;                       4764
000442  000560                          14$:    BR      35$
000444  012746  000043                    15$:  MOV     #43,-(SP)                       ;                       4766
000450  000560                          16$:    BR      37$
000452  020227  000006                    17$:  CMP     R2,#6                           ;                       4770
000456  001025                                  BNE     23$
000460  012700  000050                          MOV     #50,R0                          ;                       4774
000464  060100                                  ADD     R1,R0
000466  023727  000000G 000200                  CMP     SB.CODE,#200                    ;                       4772
000474  001003                                  BNE     18$
000476  105260  000001                          INCB    1(R0)                           ;                       4774
000502  000401                                  BR      19$                             ;                       4772
000504  105210                          18$:    INCB    (R0)                            ;                       4776
000506  032737  000001  001166'          19$:    BIT     #1,APT.MODE                     ;                       4778
000514  001403                                  BEQ     21$
000516  012746  000044                          MOV     #44,-(SP)                       ;                       4780
000522  000575                          20$:    BR      43$
000524  012746  000044                    21$:  MOV     #44,-(SP)                       ;                       4782
000530  000575                          22$:    BR      45$
000532  020227  000007                    23$:  CMP     R2,#7                           ;                       4786
000536  001014                                  BNE     27$
000540  105261  000047                          INCB    47(R1)                          ;                       4787
000544  032737  000001  001166'                BIT     #1,APT.MODE                     ;                       4789
000552  001403                                  BEQ     25$
000554  012746  000045                          MOV     #45,-(SP)                       ;                       4791
000560  000575                          24$:    BR      47$
000562  012746  000045                    25$:  MOV     #45,-(SP)                       ;                       4793
000566  000577                          26$:    BR      49$
000570  020227  000010                    27$:  CMP     R2,#10                          ;                       4797
000574  001051                                  BNE     32$
000576  012700  000046                          MOV     #46,R0                          ;                       4801
000602  060100                                  ADD     R1,R0
000604  023727  000000G 000002                  CMP     SB.CODE,#2                      ;                       4799
000612  001002                                  BNE     28$
000614  105210                                  INCB    (R0)                            ;                       4801
000616  000402                                  BR      29$                             ;                       4799
000620  105260  000001                  28$:    INCB    1(R0)                           ;                       4803
000624  005737  000000G                 29$:    TST     SB.CODE                         ;                       4805
```

```
000630  001021                                  BNE     30$
000632  132737  000001  000000G                 BITB    #1,FORCED.ERROR         ;                       4806
000640  001015                                  BNE     30$
000642  032737  004000  000000G                 BIT     #4000,SWP.FLAGS         ;                       4807
000650  001411                                  BEQ     30$
000652  112737  000001  000000G                 MOVB    #1,FORCED.ERROR         ;                       4810
000660  016337  000050  000000G                 MOV     50(R3),FER.LBN          ;                       4811
000666  016337  000044  000000G                 MOV     44(R3),FER.BC           ;                       4812
000674  032737  000001  001166'         30$:    BIT     #1,APT.MODE             ;                       4815
000702  001403                                  BEQ     31$
000704  012746  000046                          MOV     #46,-(SP)               ;                       4817
000710  000521                                  BR      47$
000712  012746  000046                  31$:    MOV     #46,-(SP)               ;                       4819
000716  000523                                  BR      49$
000720  020227  000011                  32$:    CMP     R2,#11                  ;                       4823
000724  001014                                  BNE     34$
000726  105261  000051                          INCB    51(R1)                  ;                       4824
000732  032737  000001  001166'                 BIT     #1,APT.MODE             ;                       4826
000740  001403                                  BEQ     33$
000742  012746  000047                          MOV     #47,-(SP)               ;                       4828
000746  000502                                  BR      47$
000750  012746  000047                  33$:    MOV     #47,-(SP)               ;                       4830
000754  000504                                  BR      49$
000756  020227  000012                  34$:    CMP     R2,#12                  ;                       4834
000762  001014                                  BNE     38$
000764  105261  000050                          INCB    50(R1)                  ;                       4835
000770  032737  000001  001166'                 BIT     #1,APT.MODE             ;                       4837
000776  001403                                  BEQ     36$
001000  012746  000050                          MOV     #50,-(SP)               ;                       4839
001004  000463                  35$:    BR      47$
001006  012746  000050                  36$:    MOV     #50,-(SP)               ;                       4841
001012  000465                  37$:    BR      49$
001014  020227  000013                  38$:    CMP     R2,#13                  ;                       4845
001020  001023                                  BNE     42$
001022  023727  000000G 000003                  CMP     SB.CODE,#3              ;                       4847
001030  001003                                  BNE     39$
001032  105261  000046                          INCB    46(R1)                  ;                       4849
001036  000402                                  BR      40$                     ;                       4847
001040  105261  000050                  39$:    INCB    50(R1)                  ;                       4851
001044  032737  000001  001166'         40$:    BIT     #1,APT.MODE             ;                       4853
001052  001403                                  BEQ     41$
001054  012746  000051                          MOV     #51,-(SP)               ;                       4855
001060  000435                                  BR      47$
001062  012746  000051                  41$:    MOV     #51,-(SP)               ;                       4857
001066  000437                                  BR      49$
001070  020227  000037                  42$:    CMP     R2,#37                  ;                       4861
001074  001014                                  BNE     46$
001076  105261  000050                          INCB    50(R1)                  ;                       4862
001102  032737  000001  001166'                 BIT     #1,APT.MODE             ;                       4864
001110  001403                                  BEQ     44$
001112  012746  000053                          MOV     #53,-(SP)               ;                       4866
001116  000416                  43$:    BR      47$
001120  012746  000053                  44$:    MOV     #53,-(SP)               ;                       4868
```

```
001124  000420                          45$:    BR      49$
001126  013700  000000G                 46$:    MOV     CCTLR,R0                ;                       4873
001132  006300                                  ASL     R0
001134  105260  000000G                         INCB    C.ERR.TBL(R0)
001140  032737  000001  001166'                BIT     #1,APT.MODE             ;                       4875
001146  001405                                  BEQ     48$
001150  012746  000055                         MOV     #55,-(SP)               ;                       4877
001154  004737  000000V                 47$:    JSR     PC,ERR.HRD.RTNE.APT
001160  000404                                  BR      50$                     ;                       4875
001162  012746  000055                 48$:    MOV     #55,-(SP)               ;                       4879
001166  004737  000000V                 49$:    JSR     PC,ERR.HRD.RTNE
001172  005726                          50$:    TST     (SP)+                   ;                       4882
001174  013700  000000G                 51$:    MOV     RP.ADDR,R0              ;                       4886
001200  126027  000003  000002                  CMPB    3(R0),#2
001206  001166                                  BNE     69$
001210  116001  000016                         MOVB    16(R0),R1               ;                       4889
001214  042701  177740                         BIC     #177740,R1
001220  001075                                  BNE     59$                     ;                       4891
001222  126027  000014  000201                  CMPB    14(R0),#201             ;                       4892
001230  001015                                  BNE     53$
001232  032760  001000  000022                  BIT     #1000,22(R0)            ;                       4893
001240  001011                                  BNE     53$
001242  012746  000074                         MOV     #74,-(SP)               ;                       4896
001246  004737  000000V                 52$:    JSR     PC,ERR.HRD.RTNE
001252  013700  000000G                         MOV     T.ADDR,R0               ;                       4897
001256  105260  000050                         INCB    50(R0)
001262  000537                                  BR      68$                     ;                       4895
001264  013700  000000G                 53$:    MOV     DUPPKT,R0               ;                       4901
001270  042700  007777                         BIC     #7777,R0
001274  020027  050000                         CMP     R0,#50000
001300  001006                                  BNE     54$
001302  013700  000000G                         MOV     L$LUN,R0                ;                       4904
001306  112760  000005  000000G                MOVB    #5,DUR(R0)
001314  104451                                  TRAP    51                      ;                       4905
001316  013701  000000G                 54$:    MOV     DUPPKT,R1               ;                       4907
001322  042701  170000                         BIC     #170000,R1
001326  020127  000001                         CMP     R1,#1                   ;                       4909
001332  001003                                  BNE     55$
001334  012746  000076                         MOV     #76,-(SP)               ;                       4910
001340  000470                                  BR      65$
001342  020127  000002                 55$:    CMP     R1,#2                   ;                       4913
001346  001003                                  BNE     56$
001350  012746  000077                         MOV     #77,-(SP)               ;                       4914
001354  000462                                  BR      65$
001356  020127  000003                 56$:    CMP     R1,#3                   ;                       4917
001362  001003                                  BNE     57$
001364  012746  000100                         MOV     #100,-(SP)              ;                       4918
001370  000726                                  BR      52$
001372  020127  000004                 57$:    CMP     R1,#4                   ;                       4921
001376  001003                                  BNE     58$
001400  012746  000101                         MOV     #101,-(SP)              ;                       4922
001404  000446                                  BR      65$
001406  012746  000102                 58$:    MOV     #102,-(SP)              ;                       4926
```

```
001412  000454                          BR      67$
001414  020127  000001          59$:    CMP     R1,#1                   ;                       4933
001420  001003                          BNE     60$
001422  012746  000103                  MOV     #103,-(SP)              ;                       4934
001426  000707                          BR      52$
001430  020127  000002          60$:    CMP     R1,#2                   ;                       4937
001434  001003                          BNE     61$
001436  012746  000104                  MOV     #104,-(SP)              ;                       4938
001442  000701                          BR      52$
001444  020127  000003          61$:    CMP     R1,#3                   ;                       4941
001450  001003                          BNE     62$
001452  012746  000105                  MOV     #105,-(SP)              ;                       4942
001456  000421                          BR      65$
001460  020127  000004          62$:    CMP     R1,#4                   ;                       4945
001464  001003                          BNE     63$
001466  012746  000106                  MOV     #106,-(SP)              ;                       4946
001472  000413                          BR      65$
001474  020127  000005          63$:    CMP     R1,#5                   ;                       4949
001500  001003                          BNE     64$
001502  012746  000107                  MOV     #107,-(SP)              ;                       4950
001506  000657                          BR      52$
001510  020127  000006          64$:    CMP     R1,#6                   ;                       4953
001514  001011                          BNE     66$
001516  012746  000110                  MOV     #110,-(SP)              ;                       4954
001522  004737  000000V         65$:    JSR     PC,ERR.HRD.RTNE
001526  013700  000000G                 MOV     T.ADDR,R0               ;                       4955
001532  105260  000051                  INCB    51(R0)
001536  000411                          BR      68$                     ;                       4953
001540  012746  000111          66$:    MOV     #111,-(SP)              ;                       4958
001544  004737  000000V         67$:    JSR     PC,ERR.HRD.RTNE
001550  013700  000000G                 MOV     CCTLR,R0                ;                       4959
001554  006300                          ASL     R0
001556  105260  000000G                 INCB    C.ERR.TBL(R0)
001562  005726                  68$:    TST     (SP)+                   ;                       4957
001564  000207                  69$:    RTS     PC                      ;                       4666

; Routine Size:  443 words,     Routine Base:  $CODE$ + 21532
; Maximum stack depth per invocation:  7 words


;   4964  1
```

```
;    4965  1     GLOBAL routine UPD_IO_TALLY : novalue =
;    4966  1
;    4967  1     !+
;    4968  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    4969  1     !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;    4970  1     !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;    4971  1     !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;    4972  1     !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS DROPPED.
;    4973  1     !
;    4974  1     !       IMPLICIT INPUTS:
;    4975  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    4976  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4977  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    4978  1     !               CUOFF - CST OFFSET FOR THE CURRENT UNIT
;    4979  1     !               L$LUN - CURRENT UNIT NUMBER
;    4980  1     !-
;    4981  1
;    4982  2         begin
;    4983  2
;    4984  2         local
;    4985  2             THOUSANDS : word,
;    4986  2             MILLIONS : word;                                    ! TOTAL NO. OF BYTES XFERRED TO/FROM A UNIT
;    4987  2
;    4988  3         if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)              ! IF ENDCODE IS READ
;    4989  2         then
;    4990  3             begin
;    4991  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1;         ! INCREMENT NO. OF READS AND ADD BYTE COUNT
;    4992  3             T_ADDR [BYTES_READ_LO] = .T_ADDR [BYTES_READ_LO] + .RP_ADDR [BCNT_LO];
;    4993  3             T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO];
;    4994  3             OVF_CHK (T_ADDR [TOT_READS_LO]);                    ! CHECK FOR FIELD OVERFLOW
;    4995  3             OVF_CHK (T_ADDR [BYTES_READ_LO]);                   !
;    4996  3             OVF_CHK (T_ADDR [TOT_BYT_RED_LO]);                  !
;    4997  3             end
;    4998  2         else
;    4999  2
;    5000  3             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)         ! IF ENDCODE IS WRITE
;    5001  2             then
;    5002  3                 begin
;    5003  3                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1;   ! INCREMENT NO. OF WRITES, ADD BYTE COUNT
;    5004  3                 T_ADDR [BYTES_WRIT_LO] = .T_ADDR [BYTES_WRIT_LO] + .RP_ADDR [BCNT_LO];
;    5005  3                 T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO];
;    5006  3                 OVF_CHK (T_ADDR [TOT_WRITES_LO]);               ! CHECK FOR FIELD OVERFLOW
;    5007  3                 OVF_CHK (T_ADDR [BYTES_WRIT_LO]);               !
;    5008  3                 OVF_CHK (T_ADDR [TOT_BYT_WRT_LO]);              !
;    5009  2                 end;
;    5010  2
;    5011  2         if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) or
;    5012  3            (.RP_ADDR [ENDCOD] eql (OP_WRT or OP_END))
;    5013  2         then
;    5014  3             begin
;    5015  3             MILLIONS = .T_ADDR [MBYTES_READ] + .T_ADDR [MBYTES_WRIT];   ! TOTAL BYTES TRANSFERRED
;    5016  3             THOUSANDS = .T_ADDR [BYTES_READ_HI] + .T_ADDR [BYTES_WRIT_HI];
;    5017  3
```

```
;  5016  3          if .THOUSANDS gequ 1000
;  5019  3          then
;  5020  4              begin
;  5021  4              MILLIONS = .MILLIONS + 1;                          ! COUNT THE LOWER OVERFLOW TOO!
;  5022  4              THOUSANDS = .THOUSANDS - 1000;
;  5023  3              end;
;  5024  3
;  5025  3      !
;  5026  3      ! THIS ADDED BECAUSE IT WILL TAKE FOREVER TO TRANSFER ON THE ORDER OF A MEGABYTE TO A FLOPPY
;  5027  3      ! BUT IT IS A MUCH MORE REASONABLE MEASURE FOR THE RD51/52 WINCHESTER.  THE QUESTION NOW REFERS TO
;  5028  3      ! THE TOTAL DATA TRANSFER TO THE CONTROLLER AND THIS IS PRETTY CLOSE SINCE THE FLOPPIES GET
;  5029  3      ! ABOUT 1/1000 THE DATA THE HARD DISK(S) GET.
;  5030  3      !
;  5031  3
;  5032  3          if .SWP_XFER eql 0                                     ! IF THERE IS A TRANSFER LIMIT
;  5033  3          then
;  5034  4              begin
;  5035  4
;  5036  4              if .THOUSANDS gtru 50                              !ZZZ
;  5037  4              then
;  5038  4                  EOP_FLAG = TRUE;                               ! SET END-OF-PASS FLAG
;  5039  4
;  5040  4              end
;  5041  3          else
;  5042  3
;  5043  3              if .MILLIONS gequ .SWP_XFER                        ! IF TRANSFER LIMIT IS REACHED
;  5044  3              then
;  5045  3                  EOP_FLAG = TRUE;                               ! SET END-OF-PASS FLAG
;  5046  3
;  5047  2          end;                                                   ! IF UNIT IS STILL ALIVE
;  5048  2
;  5049  2          !.....................................................................
;  5050  2          !                                                                    .
;  5051  2          ! THE FOLLOWING IS ADDED TO MAKE THE RUN TIME ABOUT 1.5 MINUTES FOR A  .
;  5052  2          ! QUICK PASS IF ALL UNITS UNDER TEST ARE FLOPPIES.                     .
;  5053  2          !.....................................................................
;  5054  2
;  5055  2      !!ZZZ   IF .RD_COUNT EQL 0                                 !IF THERE ARE NO WINCHESTERS   ZZZ
;  5056  2      !!ZZZ      THEN                                            !                              ZZZ
;  5057  2      !!ZZZ        BEGIN                                         !                              ZZZ
;  5058  2      !!ZZZ        IF .THOUSANDS GTRU 44                         !IF ABOUT 1.5 MINUTES GONE BY  ZZZ
;  5059  2      !!ZZZ        THEN                                          !                              ZZZ
;  5060  2      !!ZZZ           EOP_FLAG = TRUE;                           !SET THE END OF PASS FLAG      ZZZ
;  5061  2      !!ZZZ        END;                                          !                              ZZZ
;  5062  2
;  5063  2
;  5064  2          ROUND_OUTPUT ();                                       ! ROUND TOTALS TO FIT PRINT POSITIONS
;  5065  1          end;                                                   ! ROUTINE UPD_IO_TALLY


                                    .SBTTL  UPD.IO.TALLY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G        UPD.IO.TALLY::
                                    JSR     R1,$SAVE2                   ;                                  4965
```

```
000004  013701  000000G                       MOV     RP.ADDR,R1              ;                               4988
000010  126127  000014  000241                CMPB    14(R1),#241
000016  001027                                BNE     1$
000020  013700  000000G                       MOV     T.ADDR,R0               ;                               4991
000024  005260  000016                        INC     16(R0)
000030  066110  000020                        ADD     20(R1),(R0)             ;                               4992
000034  066160  000020  000032                ADD     20(R1),32(R0)           ;                               4993
000042  012746  000016                        MOV     #16,-(SP)               ;                               4994
000046  060016                                ADD     R0,(SP)
000050  004737  000000V                       JSR     PC,OVF.CHK
000054  013716  000000G                       MOV     T.ADDR,(SP)             ;                               4995
000060  004737  000000V                       JSR     PC,OVF.CHK
000064  013716  000000G                       MOV     T.ADDR,(SP)             ;                               4996
000070  062716  000032                        ADD     #32,(SP)
000074  000435                                BR      2$
000076  126127  000014  000242        1$:     CMPB    14(R1),#242             ;                               5000
000104  001034                                BNE     3$
000106  013700  000000G                       MOV     T.ADDR,R0               ;                               5003
000112  005260  000024                        INC     24(R0)
000116  066160  000020  000006                ADD     20(R1),6(R0)            ;                               5004
000124  066160  000020  000040                ADD     20(R1),40(R0)           ;                               5005
000132  012746  000024                        MOV     #24,-(SP)               ;                               5006
000136  060016                                ADD     R0,(SP)
000140  004737  000000V                       JSR     PC,OVF.CHK
000144  013716  000000G                       MOV     T.ADDR,(SP)             ;                               5007
000150  062716  000006                        ADD     #6,(SP)
000154  004737  000000V                       JSR     PC,OVF.CHK
000160  013716  000000G                       MOV     T.ADDR,(SP)             ;                               5008
000164  062716  000040                        ADD     #40,(SP)
000170  004737  000000V                2$:    JSR     PC,OVF.CHK
000174  005726                                TST     (SP)+                   ;                               5002
000176  013700  000000G                3$:    MOV     RP.ADDR,R0              ;                               5011
000202  126027  000014  000241                CMPB    14(R0),#241
000210  001404                                BEQ     4$
000212  126027  000014  000242                CMPB    14(R0),#242             ;                               5012
000220  001034                                BNE     8$
000222  013700  000000G                4$:    MOV     T.ADDR,R0               ;                               5015
000226  016002  000004                        MOV     4(R0),R2                ; *,MILLIONS
000232  066002  000012                        ADD     12(R0),R2               ; *,MILLIONS
000236  016001  000002                        MOV     2(R0),R1                ; *,THOUSANDS                   5016
000242  066001  000010                        ADD     10(R0),R1               ; *,THOUSANDS
000246  020127  001750                        CMP     R1,#1750                ; THOUSANDS,*                   5018
000252  103403                                BLO     5$
000254  005202                                INC     R2                      ; MILLIONS                      5021
000256  162701  001750                        SUB     #1750,R1                ; *,THOUSANDS                   5022
000262  013700  000000G                5$:    MOV     SWP.XFER,R0             ;                               5032
000266  001004                                BNE     6$
000270  020127  000062                        CMP     R1,#62                  ; THOUSANDS,*                   5036
000274  101406                                BLOS    8$
000276  000402                                BR      7$                      ;                               5038
000300  020200                         6$:    CMP     R2,R0                   ; MILLIONS,*                    5043
000302  103403                                BLO     8$
000304  112737  000001  000000G        7$:    MOVB    #1,EOP.FLAG             ;                               5045
```

```
000312  004737  000000V                8$:     JSR     PC,ROUND.OUTPUT                 ;                       5064
000316  000207                          8$:     RTS     PC                              ;                       4965

; Routine Size:  104 words,     Routine Base:  $CODE$ + 23320
; Maximum stack depth per invocation:  5 words
```

```
;    5066  1     GLOBAL routine OVF_CHK (ADDR) : novalue =
;    5067  1     !+
;    5068  1     !       THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;    5069  1     !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;    5070  1     !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;    5071  1     !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;    5072  1     !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;    5073  1     !
;    5074  1     !       INPUTS:
;    5075  1     !               ADDR - ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;    5076  1     !                      THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;    5077  1     !-
;    5078  2         begin
;    5079  2
;    5080  2         while ..ADDR gequ 1000 do                   ! IF LO-ORDER OVERFLOW
;    5081  3             begin
;    5082  3             .ADDR = ..ADDR - 1000;                  ! SUBTRACT 1000
;    5083  3             (.ADDR + 2) = .(.ADDR + 2) + 1;         ! INCR HI-ORDER
;    5084  2             end;
;    5085  2
;    5086  2         if .(.ADDR + 2) gequ 1000                   ! IF HI-ORDER OVERFLOW
;    5087  2         then
;    5088  3             begin
;    5089  3             (.ADDR + 2) = .(.ADDR + 2) - 1000;      ! SUBTRACT 1000
;    5090  3             (.ADDR + 4) = .(.ADDR + 4) + 1;         ! INCREMENT MBYTES
;    5091  2             end;
;    5092  2
;    5093  1         end;                                        ! ROUTINE OVF_CHK


                                        .SBTTL  OVF.CHK MULTI-DRIVE TEST ROUTINES
000000  010146                 OVF.CHK::
                                        MOV     R1,-(SP)                        ;                       5066
000002  016600  000004                  MOV     4(SP),R0                        ; ADDR,*                5080
000006  012701  000002                  MOV     #2,R1                           ;                       5083
000012  060001                          ADD     R0,R1
000014  021027  001750          1$:     CMP     (R0),#1750                      ;                       5080
000020  103404                          BLO     2$
000022  162710  001750                  SUB     #1750,(R0)                      ;                       5082
000026  005211                          INC     (R1)                            ;                       5083
000030  000771                          BR      1$                              ;                       5080
000032  021127  001750          2$:     CMP     (R1),#1750                      ;                       5086
000036  103404                          BLO     3$
000040  162711  001750                  SUB     #1750,(R1)                      ;                       5089
000044  005260  000004                  INC     4(R0)                           ;                       5090
000050  012601                  3$:     MOV     (SP)+,R1                        ;                       5066
000052  000207                          RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 23640
; Maximum stack depth per invocatior:  2 words
```

```
;    5094  1     GLOBAL routine ROUND_OUTPUT : novalue =
;    5095  1
;    5096  1     !+
;    5097  1     !       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
;    5098  1     !-
;    5099  1
;    5100  2         begin
;    5101  2
;    5102  2         if .T_ADDR [TOT_READS_HI] gtru 9999
;    5103  2         then
;    5104  3             begin
;    5105  3
;    5106  3             if .T_ADDR [TOT_READS_LO] lssu 999
;    5107  3             then
;    5108  4                 begin
;    5109  4                 T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 1;
;    5110  4                 T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
;    5111  3                 end;
;    5112  3
;    5113  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] - 999;
;    5114  3             T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 9999;
;    5115  2             end;
;    5116  2
;    5117  2         if .T_ADDR [TOT_WRITES_HI] gtru 9999
;    5118  2         then
;    5119  3             begin
;    5120  3
;    5121  3             if .T_ADDR [TOT_WRITES_LO] lssu 999
;    5122  3             then
;    5123  4                 begin
;    5124  4                 T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 1;
;    5125  4                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
;    5126  3                 end;
;    5127  3
;    5128  3             T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] - 999;
;    5129  3             T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 9999;
;    5130  2             end;
;    5131  2
;    5132  2         if .T_ADDR [MTOT_BYT_RED] gtru 999
;    5133  2         then
;    5134  3             begin
;    5135  3
;    5136  3             if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    5137  3             then
;    5138  4                 begin
;    5139  4                 T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    5140  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    5141  3                 end;
;    5142  3
;    5143  3             if .T_ADDR [TOT_BYT_RED_LO] lssu 999
;    5144  3             then
;    5145  4                 begin
;    5146  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 1;
```

```
;    5147  4                T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + 1000;
;    5148  4
;    5149  4                if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    5150  4                then
;    5151  5                    begin
;    5152  5                    T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    5153  5                    T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    5154  4                    end;
;    5155  3                end;
;    5156  3
;    5157  3            T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] - 999;
;    5158  3            T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 999;
;    5159  3            T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 999;
;    5160  2            end;
;    5161  2
;    5162  2        if .T_ADDR [MTOT_BYT_WRT] gtru 999
;    5163  2        then
;    5164  3            begin
;    5165  3
;    5166  3            if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    5167  3            then
;    5168  4                begin
;    5169  4                T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5170  4                T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    5171  3                end;
;    5172  3
;    5173  3            if .T_ADDR [TOT_BYT_WRT_LO] lssu 999
;    5174  3            then
;    5175  4                begin
;    5176  4                T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 1;
;    5177  4                T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + 1000;
;    5178  4
;    5179  4                if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    5180  4                then
;    5181  5                    begin
;    5182  5                    T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5183  5                    T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    5184  4                    end;
;    5185  3                end;
;    5186  3
;    5187  3            T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] - 999;
;    5188  3            T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 999;
;    5189  3            T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 999;
;    5190  2            end;
;    5191  2
;    5192  1        end;


                                    .SBTTL  ROUND.OUTPUT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       ROUND.OUTPUT::
                                    JSR     R1,$SAVE3                       ;                                   5094
000004  013700  000000G             MOV     T.ADDR,R0                       ;                                   5102
000010  012702  000020              MOV     #20,R2
```

```
000014  060002                          ADD     R0,R2
000016  021227  023417                  CMP     (R2),#23417
000022  101415                          BLOS    2$
000024  012701  000016                  MOV     #16,R1          ;                       5106
000030  060001                          ADD     R0,R1
000032  021127  001747                  CMP     (R1),#1747
000036  103003                          BHIS    1$
000040  005312                          DEC     (R2)            ;                       5109
000042  062711  001750                  ADD     #1750,(R1)      ;                       5110
000046  162711  001747          1$:     SUB     #1747,(R1)      ;                       5113
000052  162712  023417                  SUB     #23417,(R2)     ;                       5114
000056  012702  000026          2$:     MOV     #26,R2          ;                       5117
000062  060002                          ADD     R0,R2
000064  021227  023417                  CMP     (R2),#23417
000070  101415                          BLOS    4$
000072  012701  000024                  MOV     #24,R1          ;                       5121
000076  060001                          ADD     R0,R1
000100  021127  001747                  CMP     (R1),#1747
000104  103003                          BHIS    3$
000106  005312                          DEC     (R2)            ;                       5124
000110  062711  001750                  ADD     #1750,(R1)      ;                       5125
000114  162711  001747          3$:     SUB     #1747,(R1)      ;                       5128
000120  162712  023417                  SUB     #23417,(R2)     ;                       5129
000124  012703  000036          4$:     MOV     #36,R3          ;                       5132
000130  060003                          ADD     R0,R3
000132  021327  001747                  CMP     (R3),#1747
000136  101436                          BLOS    7$
000140  012701  000034                  MOV     #34,R1          ;                       5136
000144  060001                          ADD     R0,R1
000146  021127  001747                  CMP     (R1),#1747
000152  103003                          BHIS    5$
000154  005313                          DEC     (R3)            ;                       5139
000156  062711  001750                  ADD     #1750,(R1)      ;                       5140
000162  012702  000032          5$:     MOV     #32,R2          ;                       5143
000166  060002                          ADD     R0,R2
000170  021227  001747                  CMP     (R2),#1747
000174  103011                          BHIS    6$
000176  005311                          DEC     (R1)            ;                       5146
000200  062712  001750                  ADD     #1750,(R2)      ;                       5147
000204  021127  001747                  CMP     (R1),#1747      ;                       5149
000210  103003                          BHIS    6$
000212  005313                          DEC     (R3)            ;                       5152
000214  062711  001750                  ADD     #1750,(R1)      ;                       5153
000220  162712  001747          6$:     SUB     #1747,(R2)      ;                       5157
000224  162711  001747                  SUB     #1747,(R1)      ;                       5158
000230  162713  001747                  SUB     #1747,(R3)      ;                       5159
000234  012702  000044          7$:     MOV     #44,R2          ;                       5162
000240  060002                          ADD     R0,R2
000242  021227  001747                  CMP     (R2),#1747
000246  101435                          BLOS    10$
000250  012701  000042                  MOV     #42,R1          ;                       5166
000254  060001                          ADD     R0,R1
000256  021127  001747                  CMP     (R1),#1747
```

```
000262  103003                          BHIS    8$
000264  005312                          DEC     (R2)                            ;                       5169
000266  062711  001750                  ADD     #1750,(R1)                      ;                       5170
000272  062700  000040          8$:     ADD     #40,R0                          ;                       5173
000276  021027  001747                  CMP     (R0),#1747
000302  103011                          BHIS    9$
000304  005311                          DEC     (R1)                            ;                       5176
000306  062710  001750                  ADD     #1750,(R0)                      ;                       5177
000312  021127  001747                  CMP     (R1),#1747                      ;                       5179
000316  103003                          BHIS    9$
000320  005312                          DEC     (R2)                            ;                       5182
000322  062711  001750                  ADD     #1750,(R1)                      ;                       5183
000326  162710  001747          9$:     SUB     #1747,(R0)                      ;                       5187
000332  162711  001747                  SUB     #1747,(R1)                      ;                       5188
000336  162712  001747                  SUB     #1747,(R2)                      ;                       5189
000342  000207                  10$:    RTS     PC                              ;                       5094

; Routine Size:  114 words,    Routine Base:  $CODE$ + 23714
; Maximum stack depth per invocation:  5 words
```

```
;    5193  1     GLOBAL routine HOST_WRT_CHK =
;    5194  1
;    5195  1     !+
;    5196  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    5197  1     !       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
;    5198  1     !       OPTION WAS SELECTED BY THE OPERATOR.
;    5199  1     !
;    5200  1     !       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
;    5201  1     !       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
;    5202  1     !       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
;    5203  1     !       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A BYTE-BY-BYTE
;    5204  1     !       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
;    5205  1     !       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
;    5206  1     !
;    5207  1     !       IMPLICIT INPUTS:
;    5208  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5209  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;    5210  1     !-
;    5211  1
;    5212  2         begin
;    5213  2
;    5214  2         local
;    5215  2             BUFF1 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;    5216  2             BUFF2 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;    5217  2             BUFFW,                                            ! I/O BUFFER ADDRESS
;    5218  2             COUNT : word,                                     ! BYTE COUNT
;    5219  2             FLAG : byte initial (byte (TRUE)),
;    5220  2             index : signed word;
;    5221  2
;    5222  3         if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)           ! IF WRITE OPERATION
;    5223  2         then
;    5224  2             FLAG = FALSE                                      ! DON'T CALL SWEEP FROM IO_RETPKT
;    5225  2         else
;    5226  2
;    5227  2             if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) and
;    5228  3                ((index = RPS_REM ()) geq 0)                   ! IF ASSOCIATED WRITE PACKET IS FOUND
;    5229  2             then
;    5230  3                 begin
;    5231  3                 BUFFW = RETPKT [.index, BUFF_0];              ! ADDR OF ADDR OF WRITE I/O BUFFER
;    5232  3                 BUFF1 = ..BUFFW;                              ! ADDR OF WRITE I/O BUFFER
;    5233  3                 BUFF2 = .RP_ADDR [BUFF_0];                    ! ADDR OF READ I/O BUFFER
;    5234  3                 COUNT = .RP_ADDR [BCNT_LO];                   ! BYTE COUNT
;    5235  3
;    5236  3                 incr I from 1 to .COUNT do                    ! FOR EACH BYTE IN BUFFERS
;    5237  3
;    5238  3                     if .(.BUFF1)<0, 8, 0> eql .(.BUFF2)<0, 8, 0>    ! IF BYTES COMPARE O.K.
;    5239  3                     then
;    5240  4                         begin
;    5241  4                         BUFF1 = .BUFF1 + 1;                   ! ADVANCE WRITE BUFFER ADDR
;    5242  4                         BUFF2 = .BUFF2 + 1;                   ! ADVANCE READ BUFFER ADDR
;    5243  4                         end
;    5244  3                     else
;    5245  4                         begin                                 ! ELSE - COMPARE ERROR
```

```
;    5246  4                    T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;    5247  4                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5248  4
;    5249  4                    if .APT_MODE
;    5250  4                    then
;    5251  4                        ERR_HRD_RTNE_APT (42)                   ! I/O REQUEST FAILED
;    5252  4                    else
;    5253  4                        ERR_HRD_RTNE (42);
;    5254  4
;    5255  4                    EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;    5256  4
;    5257  4                    if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;    5258  4                    then
;    5259  5                        begin
;    5260  5                        DUR [.L$LUN] = DU_HERR;                 ! IF ERROR COUNT EXCEEDED
;    5261  5                        DODU (.L$LUN);                          ! DROP UNIT
;    5262  4                        end;
;    5263  4
;    5264  4                    exitloop;                                   ! NO NEED TO CONTINUE
;    5265  3                    end;                                        ! IF COMPARE ERROR
;    5266  3
;    5267  2                end;                                            ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;    5268  2
;    5269  2            return (.FLAG);
;    5270  1            end;                                                ! ROUTINE HOST_WRT_CHK


                                    .SBTTL  HOST.WRT.CHK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          HOST.WRT.CHK::
                                    JSR     R1,$SAVE5                       ;                                   5193
000004  005746                      TST     -(SP)
000006  112705  000001              MOVB    #1,R5                           ; *,FLAG                            5212
000012  013700  000000G             MOV     RP.ADDR,R0                      ;                                   5222
000016  126027  000014  000242      CMPB    14(R0),#242
000024  001002                      BNE     1$
000026  105005                      CLRB    R5                              ; FLAG                              5224
000030  000511                      BR      8$                              ;                                   5222
000032  126027  000014  000241  1$: CMPB    14(R0),#241                     ;                                   5227
000040  001105                      BNE     8$
000042  004737  000000V             JSR     PC,RPS.REM                      ;                                   5228
000046  005700                      TST     R0                              ; INDEX
000050  002501                      BLT     8$
000052  010046                      MOV     R0,-(SP)                        ; INDEX,*                           5231
000054  012746  000054              MOV     #54,-(SP)
000060  004737  000000G             JSR     PC,BL$MUL
000064  010066  000004              MOV     R0,4(SP)
000070  062700  000024G             ADD     #RETPKT+24,R0                   ; *,BUFFW
000074  011001                      MOV     (R0),R1                         ; BUFFW.BUFF1                       5232
000076  013700  000000G             MOV     RP.ADDR,R0                      ;                                   5233
000102  016002  000024              MOV     24(R0),R2                       ; *,BUFF2
000106  016004  000020              MOV     20(R0),R4                       ; *,COUNT                           5234
000112  005003                      CLR     R3                              ; I                                 5236
000114  000453                      BR      6$
```

```
000116  121112                          2$:     CMPB    (R1),(R2)               ; BUFF1,BUFF2                   5238
000120  001003                                  BNE     3$
000122  005201                                  INC     R1                      ; BUFF1                         5241
000124  005202                                  INC     R2                      ; BUFF2                         5242
000126  000446                                  BR      6$                      ;                               5238
000130  013700  000000G                  3$:     MOV     T.ADDR,R0               ;                               5246
000134  005260  000014                          INC     14(R0)
000140  105260  000051                          INCB    51(R0)                  ;                               5247
000144  032737  000001  001166'                BIT     #1,APT.MODE             ;                               5249
000152  001405                                  BEQ     4$
000154  012716  000052                          MOV     #52,(SP)                ;                               5251
000160  004737  000000V                        JSR     PC,ERR.HRD.RTNE.APT
000164  000404                                  BR      5$                      ;                               5249
000166  012716  000052                  4$:     MOV     #52,(SP)                ;                               5253
000172  004737  000000V                        JSR     PC,ERR.HRD.RTNE
000176  016616  000004                  5$:     MOV     4(SP),(SP)              ;                               5255
000202  062716  000000G                        ADD     #RETPKT,(SP)
000206  004737  000000G                        JSR     PC,EMS.CMP
000212  013700  000000G                        MOV     T.ADDR,R0               ;                               5257
000216  026037  000014  000000G               CMP     14(R0),SWP.ERROR
000224  103412                                  BLO     7$
000226  013700  000000G                        MOV     L$LUN,R0                ;                               5260
000232  112760  000004  000000G               MOVB    #4,DUR(R0)
000240  104451                                  TRAP    51                      ;                               5261
000242  000403                                  BR      7$                      ;                               5245
000244  005203                          6$:     INC     R3                      ; I                             5236
000246  020304                                  CMP     R3,R4                   ; I,COUNT
000250  003722                                  BLE     2$
000252  022626                          7$:     CMP     (SP)+,(SP)+             ;                               5230
000254  005000                          8$:     CLR     R0                      ;                               5269
000256  150500                                  BISB    R5,R0                   ; FLAG,*
000260  005726                                  TST     (SP)+                   ;                               5193
000262  000207                                  RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 24260
; Maximum stack depth per invocation:  11 words
```

```
;    5271  1     GLOBAL routine SWEEP : novalue =
;    5272  1
;    5273  1     !*
;    5274  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;    5275  1     !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;    5276  1     !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;    5277  1     !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;    5278  1     !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;    5279  1     !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;    5280  1     !       DEALLOCATED.
;    5281  1     !
;    5282  1     !       IMPLICIT INPUTS:
;    5283  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5284  1     !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;    5285  1     !-
;    5286  1
;    5287  2         begin
;    5288  2
;    5289  2         local
;    5290  2             index : signed word;
;    5291  2
;    5292  2         if (.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD          ! IF READ OPCODE OR ENDCODE
;    5293  2         then
;    5294  2
;    5295  3             if BIT_TST (SWP_FLAGS, SWF_HWC)                  ! IF HOST IS DOING WRITE-COMPARES
;    5296  2             then
;    5297  2
;    5298  2                 if (index = RPS_REM ()) geq 0                ! IF ASSOCIATED WRITE RETPKT IS FOUND
;    5299  2                 then
;    5300  3                     begin
;    5301  3                     PUT_IO_BUFF (RETPKT [.index, BUFF_0]);   ! RETURN WRITE I/O BUFFER TO POOL
;    5302  3                     PUT_RETPKT (.index);                     ! RETURN WRITE PACKET TO POOL
;    5303  2                     end;
;    5304  2
;    5305  2         PUT_IO_BUFF (RP_ADDR [BUFF_0]);                      ! RETURN CURRENT I/O BUFFER TO POOL
;    5306  2         PUT_RETPKT (.RP_INDX);                               ! RETURN CURRENT RETPKT TO POOL
;    5307  1         end;                                                 ! ROUTINE SWEEP


                                        .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                  SWEEP:: MOV     R1,-(SP)                        ;                       5271
000002  013700  000000G                 MOV     RP.ADDR,R0                      ;                       5292
000006  116000  000014                  MOVB    14(R0),R0
000012  042700  177600                  BIC     #177600,R0
000016  020027  000041                  CMP     R0,#41
000022  001026                         BNE     1$
000024  032737  000040  000000G         BIT     #40,SWP.FLAGS                   ;                       5295
000032  001422                         BEQ     1$
000034  004737  000000V                 JSR     PC,RPS.REM                      ;                       5298
000040  010001                         MOV     R0,R1                           ; *,INDEX
000042  002416                         BLT     1$
000044  010146                         MOV     R1,-(SP)                        ; INDEX,*               5301
000046  012746  000054                 MOV     #54,-(SP)
```

```
000052  004737  000000G                        JSR     PC,BL$MUL
000056  062700  000024G                        ADD     @RETPKT+24,R0
000062  010016                                 MOV     R0,(SP)
000064  004737  000000G                        JSR     PC,PUT.IO.BUFF
000070  010116                                 MOV     R1,(SP)                 ; INDEX,*               5302
000072  004737  000000G                        JSR     PC,PUT.RETPKT
000076  022626                                 CMP     (SP)+,(SP)+             ;                       5300
000100  013746  000000G                1$:     MOV     RP.ADDR,-(SP)           ;                       5305
000104  062716  000024                         ADD     #24,(SP)
000110  004737  000000G                        JSR     PC,PUT.IO.BUFF
000114  013716  000000G                        MOV     RP.INDX,(SP)            ;                       5306
000120  004737  000000G                        JSR     PC,PUT.RETPKT
000124  005726                                 TST     (SP)+                   ;                       5287
000126  012601                                 MOV     (SP)+,R1                ;                       5271
000130  000207                                 RTS     PC

; Routine Size:  45 words,    Routine Base:  $CODE$ + 24544
; Maximum stack depth per invocation:  4 words
```

```
;    5308  1     GLOBAL routine RPS_REM =
;    5309  1
;    5310  1     !*
;    5311  1     !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;    5312  1     !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;    5313  1     !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;    5314  1     !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;    5315  1     !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;    5316  1     !       OPERATION IS RETURNED TO THE CALLER.
;    5317  1     !
;    5318  1     !       IMPLICIT INPUTS:
;    5319  1     !               RP_ADDR  ADDRESS OF THE CURRENT RETURN PACKET
;    5320  1     !
;    5321  1     !       OUTPUTS:
;    5322  1     !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;    5323  1     !                       A CRN WHICH IS ONE LESS THAN THE CURRENT
;    5324  1     !-
;    5325  1
;    5326  2         begin
;    5327  2
;    5328  2         local
;    5329  2             index : signed word initial (-1);                          ! ASSUME NOT FOUND
;    5330  2
;    5331  2         incr COUNT from 0 to RP_CNT - 1 do                             ! FOR EACH ENTRY IN RP_SAVE
;    5332  2
;    5333  2             if (.RP_USE [.COUNT] eql .CCTLR) and                       ! IF THIS IS A VALID RETPKT INDEX
;    5334  3                (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;    5335  2             then
;    5336  2
;    5337  3                 if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO] - 1)) and     ! IF CORRECT CRN
;    5338  2                     (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI])) or
;    5339  3                    ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI] - 1)) and
;    5340  3                     (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;    5341  3                     (.RP_ADDR [CRF_LO] eql 0))
;    5342  2                 then
;    5343  3                     begin
;    5344  3                     index = .COUNT;                                    ! INDEX TO BE RETURNED
;    5345  3                     exitloop;                                          ! DONE
;    5346  2                     end;
;    5347  2
;    5348  2         return .index;
;    5349  1         end;                                                           ! ROUTINE RPS_REM


                                        .SBTTL  RPS.REM MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          RPS.REM::
                                        JSR     R1,$SAVE4                       ;                       5308
000004  012704  177777                  MOV     #-1,R4                          ; #,INDEX               5326
000010  005003                          CLR     R3                              ; COUNT                 5331
000012  116300  000000G          1$:    MOVB    RP.USE(R3),R0                   ; #(COUNT),#            5333
000016  020037  000000G                  CMP     R0,CCTLR
000022  001053                          BNE     4$
000024  010346                          MOV     R3,-(SP)                        ; COUNT,#               5334
```

```
000026  012746  000054                        MOV     #54,-(SP)
000032  004737  000000G                       JSR     PC,BL$MUL
000036  022626                                CMP     (SP)+,(SP)+
000040  126027  000014G 000242                CMPB    RETPKT+14(R0),#242
000046  001041                                BNE     4$
000050  010346                                MOV     R3,-(SP)                ; COUNT,*                       5337
000052  012746  000054                        MOV     #54,-(SP)
000056  004737  000000G                       JSR     PC,BL$MUL
000062  022626                                CMP     (SP)+,(SP)+
000064  013701  000000G                       MOV     RP.ADDR,R1
000070  016102  000004                        MOV     4(R1),R2
000074  005302                                DEC     R2
000076  026002  000004G                       CMP     RETPKT+4(R0),R2
000102  001004                                BNE     2$
000104  026061  000006G 000006                CMP     RETPKT+6(R0),6(R1)      ;                               5338
000112  001415                                BEQ     3$
000114  016102  000006              2$:       MOV     6(R1),R2                ;                               5339
000120  005302                                DEC     R2
000122  026002  000006G                       CMP     RETPKT+6(R0),R2
000126  001011                                BNE     4$
000130  026027  000004G 177777                CMP     RETPKT+4(R0),#-1        ;                               5340
000136  001005                                BNE     4$
000140  005761  000004                        TST     4(R1)                   ;                               5341
000144  001002                                BNE     4$
000146  010304                      3$:       MOV     R3,R4                   ; COUNT,INDEX                   5344
000150  000404                                BR      5$                      ;                               5343
000152  005203                      4$:       INC     R3                      ; COUNT                         5331
000154  020327  000007                        CMP     R3,#7                   ; COUNT,*
000160  003714                                BLE     1$
000162  010400                      5$:       MOV     R4,R0                   ; INDEX,*                       5326
000164  000207                                RTS     PC                      ;                               5308

; Routine Size:  59 words,      Routine Base:  $CODE$ + 24676
; Maximum stack depth per invocation:  8 words
```

```
;    5350  1     GLOBAL routine DR_RETPKT : novalue =
;    5351  1
;    5352  1     !+
;    5353  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT FOR ALL PACKETS ORIGINATING AT
;    5354  1     !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;    5355  1     !       FATAL DEVICE ERRORS.
;    5356  1     !
;    5357  1     !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;    5358  1     !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;    5359  1     !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED.
;    5360  1     !
;    5361  1     !       IMPLICIT INPUTS:
;    5362  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;    5363  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5364  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    5365  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5366  1     !-
;    5367  1
;    5368  2         begin
;    5369  2
;    5370  2
;    5371  2         PUTA_BUFF ();                           ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;    5372  2
;    5373  2         incr index from 0 to RP_CNT - 1 do      ! FOR EACH ENTRY IN CONTROLLER'S RP_SAVE
;    5374  2
;    5375  2             if .RP_USE [.index] eql .CCTLR      ! IF VALID RETPKT INDEX
;    5376  2             then
;    5377  2                 PUT_RETPKT (.index);            ! RETURN RETPKT TO POOL
;    5378  2
;    5379  2         QIO [.CCTLR] = 0;                       ! CLEAR NO. OF OUTSTANDING QIOs
;    5380  2         CST_ADDR [STATE] = OFFLINE;             ! MARK CST OFFLINE
;    5381  2         DROP_CTLR (.CCTLR, DU_CFATAL);          ! DROP CONTROLLER'S UNITS
;    5382  2         PUT_RETPKT (.RP_INDX);                  ! PUT BACK RETPKT
;    5383  1         end;                                    ! ROUTINE DR_RETPKT


                                    .SBTTL  DR.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                DR.RETPKT::
                                    MOV     R1,-(SP)                        ;                       5350
000002  004737  000000G             JSR     PC,PUTA.BUFF                    ;                       5371
000006  005001                      CLR     R1                              ; INDEX                 5373
000010  116100  000000G     1$:     MOVB    RP.USE(R1),R0                   ; *(INDEX),*            5375
000014  020037  000000G             CMP     R0,CCTLR
000020  001004                      BNE     2$
000022  010146                      MOV     R1,-(SP)                        ; INDEX,*               5377
000024  004737  000000G             JSR     PC,PUT.RETPKT
000030  005726                      TST     (SP)+
000032  005201              2$:     INC     R1                              ; INDEX                 5373
000034  020127  000007              CMP     R1,#7                           ; INDEX,*
000040  003763                      BLE     1$
000042  013701  000000G             MOV     CCTLR,R1                        ;                       5379
000046  105061  000000G             CLRB    QIO(R1)
000052  013700  000000G             MOV     CST.ADDR,R0                     ;                       5380
```

```
000056  042760  100000  000002                  BIC     #100000,2(R0)
000064  010146                                  MOV     R1,-(SP)                ;                       5381
000066  012746  000006                          MOV     #6,-(SP)
000072  004737  000000G                         JSR     PC,DROP.CTLR
000076  013716  000000G                         MOV     RP.INDX,(SP)            ;                       5382
000102  004737  000000G                         JSR     PC,PUT.RETPKT
000106  022626                                  CMP     (SP)+,(SP)+             ;                       5368
000110  012601                                  MOV     (SP)+,R1                ;                       5350
000112  000207                                  RTS     PC

; Routine Size:  38 words,      Routine Base:  $CODE$ + 25064
; Maximum stack depth per invocation:  4 words
```

```
;    5384  1     %sbttl 'RDRX INTERRUPT SERVICE ROUTINES'
;    5385  1
;    5386  1     !+
;    5387  1     !       THERE EXISTS AN RDRX INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;    5388  1     !       CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;    5389  1     !       APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
;    5390  1     !       BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;    5391  1     !-
;    5392  1
;    5393  2     BGNSRV (AZINTO);
;    5394  2     ICTLR = 0;
;    5395  2     AZINT ();
;    5396  1     ENDSRV;


                                        .SBTTL  AZINTO RDRX INTERRUPT SERVICE ROUTINES
000000  010046                  AZINTO::MOV     R0,-(SP)                        ;       5393
000002  005037  000104'                 CLR     ICTLR                           ;       5394
000006  004737  000000V                 JSR     PC,AZINT                        ;       5395
000012  012600                          MOV     (SP)+,R0                        ;       5393
000014  000002                          RTI

; Routine Size:  7 words,      Routine Base:  $CODE$ + 25200
; Maximum stack depth per invocation:  2 words
```

```
;    5397  1     GLOBAL routine AZINT : novalue =
;    5398  1
;    5399  1     !+
;    5400  1     !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RDRX CONTROLLERS.
;    5401  1     !       AFTER CALCULATING THE DCT ADDRESS FOR THE INTERRUPTING DEVICE, THIS
;    5402  1     !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;    5403  1     !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;    5404  1     !       OTHERWISE, THE SA VALUE IS CHECKED FOR A FATAL ERROR, AND THE COMMAND
;    5405  1     !       AND RESPONSE RINGS ARE POLLED.
;    5406  1     !-
;    5407  1
;    5408  2         begin
;    5409  2         IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);           ! GET DCT ADDRESS
;    5410  2         ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);           ! GET CST ADDRESS
;    5411  2         IRDRX_ADDR = .ICST_ADDR [IP_ADDR];                  ! GET RDRX ADDRESS
;    5412  2         ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2);    ! GET COMM_AREA ADDR
;    5413  2         IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];   ! SAVE SA REGISTER
;    5414  2
;    5415  2         if .IDCT_ADDR [IG_INT]                              ! IGNORE INTERRUPT?
;    5416  2         then
;    5417  2             return;                                         ! RETURN IF INTERRUPTS IGNORED
;    5418  2
;    5419  3         if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)            ! IF FATAL ERROR
;    5420  2         then
;    5421  2             FATAL_ERROR ()
;    5422  2         else
;    5423  3             begin
;    5424  3             POLL_CRING ();                                  ! POLL COMMAND RING
;    5425  3             POLL_RRING ();                                  ! POLL RESPONSE RING
;    5426  2             end;
;    5427  2
;    5428  1         end;


                                        .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146                 AZINT:: MOV     R1,-(SP)                        ;                       5397
000002  005746                         TST     -(SP)
000004  013701  000104'                MOV     ICTLR,R1                        ;                       5409
000010  010146                         MOV     R1,-(SP)
000012  012746  000022                 MOV     #22,-(SP)
000016  004737  000000G                JSR     PC,BL$MUL
000022  062700  000000G                ADD     #DCT,R0
000026  010037  000100'                MOV     R0,IDCT.ADDR
000032  010116                         MOV     R1,(SP)                         ;                       5410
000034  012746  000126                 MOV     #126,-(SP)
000040  004737  000000G                JSR     PC,BL$MUL
000044  062700  000000G                ADD     #CST,R0
000050  010037  000076'                MOV     R0,ICST.ADDR
000054  011037  000000G                MOV     (R0),IRDRX.ADDR                 ; ICST.ADDR,*           5411
000060  010116                         MOV     R1,(SP)                         ;                       5412
000062  012746  000050                 MOV     #50,-(SP)
000066  004737  000000G                JSR     PC,BL$MUL
000072  062700  000000'                ADD     #COMM.AREA,R0
```

```
000076  010037  000074'                          MOV     R0,ICOM.ADDR
000102  013701  000100'                          MOV     IDCT.ADDR,R1            ;                       5413
000106  013700  000000G                          MOV     IRDRX.ADDR,R0
000112  016066  000002  000010                   MOV     2(R0),10(SP)            ; *,RC.REG
000120  016661  000010  000002                   MOV     10(SP),2(R1)            ; RC.REG,*
000126  032711  040000                           BIT     #40000,(R1)             ; *,IDCT.ADDR           5415
000132  001016                                   BNE     2$                      ;                       5397
000134  016601  000010                           MOV     10(SP),R1               ;                       5419
000140  042701  077777                           BIC     #77777,R1
000144  020127  100000                           CMP     R1,#-100000
000150  001003                                   BNE     1$
000152  004737  000000V                          JSR     PC,FATAL.ERROR          ;                       5421
000156  000404                                   BR      2$                      ;                       5419
000160  004737  000000V                  1$:     JSR     PC,POLL.CRING           ;                       5424
000164  004737  000000V                          JSR     PC,POLL.RRING           ;                       5425
000170  062706  000012                  2$:     ADD     #12,SP                  ;                       5397
000174  012601                                   MOV     (SP)+,R1
000176  000207                                   RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 25216
; Maximum stack depth per invocation:  7 words


;   5429  1
```

```
;    5430  1
;    5431  1     !+
;    5432  1     GLOBAL ROUTINE DUP_RSP : NOVALUE =                                  !ZZZ
;    5433  1
;    5434  1     !*
;    5435  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH DUP RESPONSE
;    5436  1     !       ITS GENERAL PURPOSE IS TO ACT ON A DATAGRAM OR SEQUENTIAL MESSAGE.
;    5437  1     !       IF THE MESSAGE TYPE IS SEQUENTIAL, THE ROUTINE COPIES THE
;    5438  1     !       CONTENTS OF THE MESSAGE ENVELOPE INTO A RETURN PACKET SO THAT THE
;    5439  1     !       ENVELOPE CAN BE RETURNED TO THE CONTROLLER.
;    5440  1     !
;    5441  1     !       IMPLICIT INPUTS:
;    5442  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5443  1     !               IPKT_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;    5444  1     !-
;    5445  2         begin
;    5446  2
;    5447  2         local
;    5448  2             R_INDEX : signed word,
;    5449  2             DEBUG,                                                  !ZZZ
;    5450  2             SRC_ADDR,
;    5451  2             DST_ADDR,
;    5452  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    5453  2         !PRINTX (DER34);
;    5454  2
;    5455  2         incr COUNT from 0 to PKT_CNT - 1 do
;    5456  2
;    5457  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;    5458  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5459  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;    5460  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;    5461  2                (.MSCP_PKT [.COUNT, CONNID] eql CID_DUP) and
;    5462  3                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;    5463  2             then
;    5464  3                 begin
;    5465  3                 P_INDEX = .COUNT;                                   ! SET PKT NUMBER
;    5466  3                 exitloop;
;    5467  2                 end;
;    5468  2
;    5469  2         if .P_INDEX lss 0                                           ! IF COMMAND NOT FOUND
;    5470  2         then
;    5471  3             begin
;    5472  3             PRINTF (DBM108, .IPKT_ADDR [CRN_LO]);                   ! UNKNOWN COMMAND REF. NUMBER
;    5473  3             return;
;    5474  2             end;
;    5475  2
;    5476  2         if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                    ! IF RETPKT IS NOT AVAILABLE
;    5477  2         then
;    5478  2             DEBUG = TRUE                                            !TO SEE IF THIS PATH TAKEN          ZZZ
;    5479  2     !       PRINTF (DBM112)                                         ! "DUP-RSP: RETPKT NOT AVAILABLE"   ZZZ
;    5480  2         else
;    5481  3             begin
;    5482  3             SRC_ADDR = .IPKT_ADDR + 6;                              ! SET UP COPY (SKIP OVER PKT DESC)
```

```
;  5483  3          R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);   ! START OF ALLOCATED RETPKT
;  5484  3
;  5485  3          incr COUNT from 1 to RP_LEN do
;  5486  4              begin
;  5487  4              .DST_ADDR = ..SRC_ADDR;                 ! COPY 1 WORD
;  5488  4              DST_ADDR = .DST_ADDR + 2;               ! ADVANCE DESTINATION ADDR
;  5489  4              SRC_ADDR = .SRC_ADDR + 2;               ! ADVANCE SOURCE ADDR
;  5490  3              end;                                    ! COPY LOOP
;  5491  3
;  5492  3          IN_IODQ (.R_INDEX);                         ! PUT RETPKT INDEX INTO IODQ
;  5493  2          end;                                        ! IF RETPKT WAS ALLOCATED
;  5494  2
;  5495  2
;  5496  2      if .P_INDEX geq 0                               ! IF ASSOC CMD PKT WAS FOUND
;  5497  2      then
;  5498  2          PUT_PKT (.P_INDEX);                         ! RETURN COMMAND PACKET TO POOL
;  5499  2
;  5500  1      end;                                            ! ROUTINE DUP-RSP


                                    .SBTTL  DUP.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DUP.RSP::
                                    JSR     R1,$SAVE3                       ;                       5432
000004  013701  000000G                MOV     IPKT.ADDR,R1                    ;                       5457
000010  005002                         CLR     R2                              ; COUNT                 5455
000012  010246                  1$:    MOV     R2,-(SP)                        ; COUNT,*               5457
000014  012746  000106                 MOV     #106,-(SP)
000020  004737  000000G                JSR     PC,BL$MUL
000024  022626                         CMP     (SP)+,(SP)+
000026  026061  000012G 000012         CMP     MSCP.PKT+12(R0),12(R1)
000034  001024                         BNE     2$
000036  026061  000014G 000014         CMP     MSCP.PKT+14(R0),14(R1)          ;                       5458
000044  001020                         BNE     2$
000046  026011  000000G                CMP     MSCP.PKT(R0),(R1)               ;                       5459
000052  001415                         BEQ     2$
000054  105760  000022G                TSTB    MSCP.PKT+22(R0)                 ;                       5460
000060  100412                         BMI     2$
000062  126027  000011G 000002         CMPB    MSCP.PKT+11(R0),#2              ;                       5461
000070  001006                         BNE     2$
000072  105761  000022                 TSTB    22(R1)                          ;                       5462
000076  100003                         BPL     2$
000100  010237  000000G                MOV     R2,P.INDEX                      ; COUNT,*               5465
000104  000406                         BR      3$                              ;                       5464
000106  005202                  2$:    INC     R2                              ; COUNT                 5455
000110  020227  000013                 CMP     R2,#13                          ; COUNT,*
000114  003736                         BLE     1$
000116  005737  000000G                TST     P.INDEX                         ;                       5469
000122  002013                  3$:    BGE     4$
000124  016146  000012                 MOV     12(R1),-(SP)                    ;                       5472
000130  012746  000000G                MOV     #DBM108,-(SP)
000134  012746  000002                 MOV     #2,-(SP)
000140  010600                         MOV     SP,R0                           ; SP,*
000142  104417                         TRAP    17
```

```
000144  062706  000006                      ADD     #6,SP                           ;                                       5473
000150  000207                              RTS     PC                              ;                                       5471
000152  013746  000104'              4$:     MOV     ICTLR,-(SP)                     ;                                       5476
000156  004737  000000G                     JSR     PC,GET.RETPKT
000162  010001                              MOV     R0,R1                           ; *,R.INDEX
000164  005726                              TST     (SP)+
000166  005701                              TST     R1                              ; R.INDEX
000170  002003                              BGE     5$
000172  012700  000001                      MOV     #1,R0                           ; *,DEBUG                               5478
000176  000425                              BR      7$                              ;                                       5476
000200  013702  000000G              5$:     MOV     IPKT.ADDR,R2                    ; *,SRC.ADDR                            5482
000204  062702  000006                      ADD     #6,R2                           ; *,SRC.ADDR
000210  010146                              MOV     R1,-(SP)                        ; R.INDEX,*                             5483
000212  012746  000054                      MOV     #54,-(SP)
000216  004737  000000G                     JSR     PC,BL$MUL
000222  062700  000000G                     ADD     #RETPKT,R0
000226  010003                              MOV     R0,R3                           ; *,DST.ADDR
000230  012700  000026                      MOV     #26,R0                          ; *,COUNT                               5485
000234  012223                      6$:     MOV     (R2)+,(R3)+                     ; SRC.ADDR,DST.ADDR                     5487
000236  005300                              DEC     R0                              ; COUNT                                 5485
000240  001375                              BNE     6$
000242  010116                              MOV     R1,(SP)                         ; R.INDEX,*                             5492
000244  004737  000000G                     JSR     PC,IN.IODQ
000250  022626                              CMP     (SP)+,(SP)+                     ;                                       5481
000252  013700  000000G              7$:     MOV     P.INDEX,R0                      ;                                       5496
000256  002404                              BLT     8$
000260  010046                              MOV     R0,-(SP)                        ;                                       5498
000262  004737  000000G                     JSR     PC,PUT.PKT
000266  005726                              TST     (SP)+
000270  000207                      8$:     RTS     PC                              ;                                       5432

; Routine Size:  93 words,      Routine Base:  $CODE$ + 25416
; Maximum stack depth per invocation:  9 words


;   5501  1
```

```
;    5502  1     GLOBAL routine FATAL_ERROR : novalue =
;    5503  1
;    5504  1     !+
;    5505  1     !       THIS ROUTINE IS CALLED BY THE INTERRUPT SERVICE ROUTINE (AZINT) UPON
;    5506  1     !       DETECTING AN UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER.
;    5507  1     !       ITS PURPOSE IS TO CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF
;    5508  1     !       THE EXERCISER, AND TO INFORM THE "PROGRAM" PORTION OF THE EVENT VIA
;    5509  1     !       RETURN PACKET.
;    5510  1     !
;    5511  1     !       IMPLICIT INPUTS:
;    5512  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5513  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5514  1     !               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    5515  1     !               IRDRX_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S IP REGISTER
;    5516  1     !-
;    5517  1
;    5518  2         begin
;    5519  2
;    5520  2         local
;    5521  2             index : signed word,
;    5522  2             U_SAVE : word;
;    5523  2
;    5524  2         SA_REG = .IDCT_ADDR [SA_SAVE];
;    5525  2         U_SAVE = .L$LUN;                                    ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
;    5526  2         C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    5527  2
;    5528  2         if .APT_MODE
;    5529  2         then
;    5530  3             begin
;    5531  3             .MAIL_BOX_TESTNUM = 1;
;    5532  3             .MAIL_BOX_SUBTST = 0;
;    5533  2             end;
;    5534  2
;    5535  2         L$LUN = .ICST_ADDR [OF_UN + OF_DATA, D_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
;    5536  2         ERRDF (14, EGD_14, EMS_14);                         ! FATAL CONTROLLER ERROR
;    5537  2         L$LUN = .U_SAVE;                                    ! RESTORE PRE-INTERRUPT CURRENT UNIT
;    5538  2         DRV_CTLERR (.ICTLR);                                ! CLEAN UP DRIVER DATA FOR CONTROLLER
;    5539  2
;    5540  2         if (index = GET_RETPKT (.ICTLR)) lss 0              ! TRY TO GET A RETPKT; IF FAILURE
;    5541  2         then
;    5542  3             PRINTF (DBM18)                                  ! "FATAL_ERROR: RETPKT NOT AVAILABLE"
;    5543  2         else
;    5544  3             begin                                           ! IF RETPKT WAS ALLOCATED
;    5545  3             RETPKT [.index, CONID] = CID_DRIVER;            ! SET CONNECTION ID TO "DRIVER"
;    5546  3             RETPKT [.index, MESTYP] = MT_FATAL;             ! FATAL ERROR
;    5547  3             RETPKT [.index, CTLR] = .ICTLR;                 ! CONTROLLER NUMBER
;    5548  3             IN_IODQ (.index);                               ! LOAD RETPKT INDEX INTO IODQ
;    5549  2             end;                                            ! IF RETPKT WAS ALLOCATED
;    5550  2
;    5551  1         end;                                                ! ROUTINE FATAL_ERR

                                .SBTTL  FATAL.ERROR RDRX INTERRUPT SERVICE ROUTINES
```

```
000000  004137  000000G                FATAL.ERROR::
                                        JSR     R1,$SAVE2                       ;                               5502
000004  013700  000100'                 MOV     IDCT.ADDR,R0                    ;                               5524
000010  016037  000002  000000G        MOV     2(R0),SA.REG
000016  013701  000000G                MOV     L$LUN,R1                        ; *,U.SAVE                      5525
000022  013700  000104'                 MOV     ICTLR,R0                        ;                               5526
000026  006300                          ASL     R0
000030  105260  000000G                INCB    C.ERR.TBL(R0)
000034  032737  000001  001166'        BIT     #1,APT.MODE                     ;                               5528
000042  001405                          BEQ     1$
000044  012777  000001  001170'        MOV     #1,@MAIL.BOX.TESTNUM            ;                               5531
000052  005077  001172'                 CLR     @MAIL.BOX.SUBTST                ;                               5532
000056  013700  000076'        1$:     MOV     ICST.ADDR,R0                    ;                               5535
000062  016002  000006                  MOV     6(R0),R2
000066  000302                          SWAB    R2
000070  042702  177760                  BIC     #177760,R2
000074  010237  000000G                MOV     R2,L$LUN
000100  104455                          TRAP    55                              ;                               5536
000102  000016                          .WORD   16
000104  000000G                         .WORD   EGD.14
000106  000000G                         .WORD   EMS.14
000110  010137  000000G                MOV     R1,L$LUN                        ; U.SAVE,*                      5537
000114  013746  000104'                 MOV     ICTLR,-(SP)                     ;                               5538
000120  004737  000000G                JSR     PC,DRV.CTLERR
000124  013716  000104'                 MOV     ICTLR,(SP)                      ;                               5540
000130  004737  000000G                JSR     PC,GET.RETPKT
000134  010001                          MOV     R0,R1                           ; *,INDEX
000136  002007                          BGE     2$
000140  012716  000000G                MOV     #DBM18,(SP)                     ;                               5542
000144  012746  000001                  MOV     #1,-(SP)
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104417                          TRAP    17
000154  000424                          BR      3$                              ;                               5540
000156  010116                  2$:     MOV     R1,(SP)                         ; INDEX,*                       5545
000160  012746  000054                  MOV     #54,-(SP)
000164  004737  000000G                JSR     PC,BL$MUL
000170  062700  000002G                ADD     #RETPKT+2,R0
000174  112760  000003  000001        MOVB    #3,1(R0)
000202  013702  000104'                 MOV     ICTLR,R2                        ;                               5547
000206  042702  177760                  BIC     #177760,R2
000212  112710  000060                  MOVB    #60,(R0)
000216  150210                          BISB    R2,(R0)
000220  010116                          MOV     R1,(SP)                         ; INDEX,*                       5548
000222  004737  000000G                JSR     PC,IN.IODQ
000226  022626                  3$:     CMP     (SP)+,(SP)+                     ;                               5518
000230  000207                          RTS     PC                              ;                               5502

; Routine Size:  77 words,      Routine Base:  $CODE$ + 25710
; Maximum stack depth per invocation:  7 words
```

```
;    5552  1     GLOBAL routine POLL_CRING : novalue =
;    5553  1
;    5554  1     !+
;    5555  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    5556  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    5557  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S COMMAND RING AND CHECK FOR ANY
;    5558  1     !       COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY THE CONTROLLER. SUCH SLOTS
;    5559  1     !       HAVE BEEN RETURNED TO THE HOST, INDICATED BY A ZERO OWNERSHIP BIT. FOR
;    5560  1     !       EACH SLOT THAT HAS BEEN RETURNED TO THE HOST, THE CRING COUNT IS
;    5561  1     !       DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO THE NEXT SLOT IN
;    5562  1     !       THE COMMAND RING.
;    5563  1     !
;    5564  1     !       IMPLICIT INPUTS:
;    5565  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5566  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5567  1     !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
;    5568  1     !-
;    5569  1
;    5570  2         begin
;    5571  2
;    5572  3         while ((.IDCT_ADDR [CRING_CNT] gtru 0) and                  ! WHILE # OF COMMANDS IN CRING > 0 AND
;    5573  2             not (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) do  ! CURRENT SLOT IS HOST-OWNED
;    5574  3             begin
;    5575  3             IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1;     ! DECREMENT # CMDs IN CRING
;    5576  3             IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;         ! ADVANCE TO NEXT SLOT TO POLL
;    5577  3
;    5578  3             if .IDCT_ADDR [CR_POLL] gtra .IDCT_ADDR [CR_END]        ! IF BEYOND END OF RING
;    5579  3             then
;    5580  3                 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];          ! SET POINTER TO TOP OF CRING
;    5581  3
;    5582  2             end;
;    5583  2
;    5584  2         ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER
;    5585  1         end;


                                        .SBTTL  POLL.CRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G        POLL.CRING::
                                        JSR     R1,$SAVE2                       ;                       5552
000004  013701  000100'                 MOV     IDCT.ADDR,R1                    ;                       5572
000010  012702  000016                  MOV     #16,R2                          ;                       5576
000014  060102                          ADD     R1,R2
000016  105711                  1$:     TSTB    (R1)                            ;                       5572
000020  001422                          BEQ     2$
000022  016100  000016                  MOV     16(R1),R0                       ;                       5573
000026  016000  000002                  MOV     2(R0),R0
000032  042700  077777                  BIC     #77777,R0
000036  020027  100000                  CMP     R0,#-100000
000042  001411                          BEQ     2$
000044  105311                          DECB    (R1)                            ;                       5575
000046  062712  000004                  ADD     #4,(R2)                         ;                       5576
000052  021261  000012                  CMP     (R2),12(R1)                     ;                       5578
000056  101757                          BLOS    1$
```

```
000060  016112  000010                          MOV     10(R1),(R2)                     ;                               5580
000064  000754                                  BR      1$                              ;                               5572
000066  013700  000074'                 2$:     MOV     ICOM.ADDR,R0                    ;                               5584
000072  005060  000004                          CLR     4(R0)
000076  000207                                  RTS     PC                              ;                               5552

; Routine Size:  32 words,      Routine Base:  $CODE$ + 26142
; Maximum stack depth per invocation:  4 words
```

```
;    5586  1     GLOBAL routine POLL_RRING : novalue =
;    5587  1
;    5588  1
;    5589  1     !+
;    5590  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    5591  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    5592  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S RESPONSE RING AND CHECK FOR ANY
;    5593  1     !       SLOTS WHICH HAVE BEEN RETURNED TO THE HOST (OWNERSHIP BIT = 0). FOR
;    5594  1     !       EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS PROCESSED BASED ON ITS
;    5595  1     !       CONNECTION ID (DISK OR DUP). AFTER PROCESSING, THE MESSAGE PACKET
;    5596  1     !       IS RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET
;    5597  1     !       TO 1).
;    5598  1     !
;    5599  1     !       IMPLICIT INPUTS:
;    5600  1     !               ICTLR - NUMBER OF INTERRUPTING CONTROLLER
;    5601  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5602  1     !-
;    5603  1
;    5604  1
;    5605  2         begin
;    5606  2
;    5607  2
;    5608  2         while not (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) do          ! WHILE O = 0
;    5609  3             begin
;    5610  3             IPKT_ADDR = ..IDCT_ADDR [RR_POLL] - 10;                          ! ADDRESS OF RESPONSE PACKET
;    5611  3
;    5612  3
;    5613  4     IF NOT (.IPKT_ADDR [CONNID] EQL CID_DUP)                 !               ZZZ
;    5614  3             THEN                                             !               ZZZ
;    5615  3             (CREDIT_BAL = .CREDIT_BAL + .IPKT_ADDR [CREDITS];);              !ZZZ
;    5616  3     !IT WAS NOTICE THAT DUP WAS SENDIND BACK CREDITS WHICH IT SHOULD NOT.     !ZZZ
;    5617  3             selectoneu .IPKT_ADDR [CONNID] of
;    5618  3                 set
;    5619  3
;    5620  3                 [CID_DISK] :        DISK_RSP ();
;    5621  3
;    5622  3                 [CID_DUP]  :        DUP_RSP  ();                             !ZZZ
;    5623  3
;    5624  3                 [otherwise] :       PRINTF (DBM20, .IPKT_ADDR [CONNID], .IRDRX_ADDR);
;    5625  3                                                                              ! "BAD CONN ID = XXXXX FROM XXXXXX"
;    5626  3                 tes;
;    5627  3
;    5628  3
;    5629  3             IPKT_ADDR [MSGLEN] = MSG_LEN * 2;                                ! RE-INIT PKT FIELDS; MESSAGE LENGTH
;    5630  3             IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;                  ! ADVANCE TO HI ORDER WORD OF RING SLOT
;    5631  3             .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];                      ! RETURN SLOT TO CONTROLLER
;    5632  3             .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED_OWN or ED_FLAG; ! OWNERSHIP TOO
;    5633  3             IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;                  ! ADVANCE TO NEXT RRING SLOT
;    5634  3
;    5635  3             if .IDCT_ADDR [RR_POLL] gtra .IDCT_ADDR [RR_END]                 ! IF BEYOND END OF RING
;    5636  3
;    5637  3             then
;    5638  3                 IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG];                   ! CYCLE TO TOP OF RING
```

```
;    5639  3
;    5640  2              end;                                    ! WHILE LOOP
;    5641  2
;    5642  2          ICOM_ADDR [RSP_INT] = 0;                    ! CLR RESPONSE INTERRUPT WRD IN RING HEADER
;    5643  1          end;


                                        .SBTTL  POLL.RRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          POLL.RRING::
                                        JSR     R1,$SAVE3                       ;                       5586
000004  013701  000100'                 MOV     IDCT.ADDR,R1                    ;                       5608
000010  062701  000014                  ADD     #14,R1
000014  011100                  1$:     MOV     (R1),R0
000016  016000  000002                  MOV     2(R0),R0
000022  042700  077777                  BIC     #77777,R0
000026  020027  100000                  CMP     R0,#-100000
000032  001504                          BEQ     6$
000034  017137  000000  000000G        MOV     @0(R1),IPKT.ADDR                ;                       5610
000042  162737  000012  000000G        SUB     #12,IPKT.ADDR
000050  013700  000000G                MOV     IPKT.ADDR,R0                    ;                       5613
000054  005002                          CLR     R2
000056  156002  000011                  BISB    11(R0),R2
000062  020227  000002                  CMP     R2,#2
000066  001406                          BEQ     2$
000070  116003  000010                  MOVB    10(R0),R3                       ;                       5615
000074  042703  177760                  BIC     #177760,R3
000100  060337  000000G                ADD     R3,CREDIT.BAL
000104  005702                  2$:     TST     R2                              ;                       5620
000106  001003                          BNE     3$
000110  004737  000000V                JSR     PC,DISK.RSP
000114  000421                          BR      5$                              ;                       5617
000116  020227  000002          3$:     CMP     R2,#2                           ;                       5622
000122  001003                          BNE     4$
000124  004737  025416'                JSR     PC,DUP.RSP
000130  000413                          BR      5$                              ;                       5617
000132  013746  000000G          4$:     MOV     IRDRX.ADDR,-(SP)                ;                       5624
000136  010246                          MOV     R2,-(SP)                        ;
000140  012746  000000G                MOV     #DBM20,-(SP)
000144  012746  000003                  MOV     #3,-(SP)
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104417                          TRAP    17
000154  062706  000010                  ADD     #10,SP
000160  013700  000000G          5$:     MOV     IPKT.ADDR,R0                    ;                       5629
000164  012760  000074  000006         MOV     #74,6(R0)
000172  013702  000100'                 MOV     IDCT.ADDR,R2                    ;                       5630
000176  010201                          MOV     R2,R1
000200  062701  000014                  ADD     #14,R1
000204  062711  000002                  ADD     #2,(R1)
000210  016071  000002  000000         MOV     2(R0),@0(R1)                    ;                       5631
000216  052771  140000  000000         BIS     #-40000,@0(R1)                  ;                       5632
000224  062711  000002                  ADD     #2,(R1)                         ;                       5633
000230  021162  000006                  CMP     (R1),6(R2)                      ;                       5635
000234  101667                          BLOS    1$
```

```
000236  016211  000004                        MOV     4(R2),(R1)                      ;                       5638
000242  000664                                BR      1$                              ;                       5608
000244  013700  000074'              6$:     MOV     ICOM.ADDR,R0                    ;                       5642
000250  005060  000006                        CLR     6(R0)
000254  000207                                RTS     PC                              ;                       5586

; Routine Size:  87 words,      Routine Base:  $CODE$ + 26242
; Maximum stack depth per invocation:  10 words


;   5644  1
```

```
;    5645  1
;    5646  1
;    5647  1     GLOBAL routine DISK_RSP : novalue =
;    5648  1
;    5649  1
;    5650  1     !+
;    5651  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
;    5652  1     !       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
;    5653  1     !       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
;    5654  1     !       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
;    5655  1     !       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
;    5656  1     !
;    5657  1     !       IMPLICIT INPUTS:
;    5658  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    5659  1     !                           MESSAGE
;    5660  1     !-
;    5661  1
;    5662  1
;    5663  1
;    5664  1         selectoneu .IPKT_ADDR [MSGTYP] of
;    5665  1
;    5666  1             set
;    5667  1
;    5668  1             [MT_SEQ] :      SEQUEN ();
;    5669  1
;    5670  1             [MT_DG] :       DATAGM ();
;    5671  1
;    5672  1             [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED"
;    5673  1             tes;
;    5674  1

                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                          DISK.RSP::
                                        MOV     R1,-(SP)                        ;                       5647
000002  013700  000000G                 MOV     IPKT.ADDR,R0                    ;                       5664
000006  116001  000010                  MOVB    10(R0),R1
000012  006201                          ASR     R1
000014  006201                          ASR     R1
000016  006201                          ASR     R1
000020  006201                          ASR     R1
000022  042701  177760                  BIC     #177760,R1
000026  001003                          BNE     1$                              ;                       5669
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      3$                              ;                       5664
000036  020127  000001          1$:     CMP     R1,#1                           ;                       5671
000042  001003                          BNE     2$
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      3$                              ;                       5664
000052  010146                  2$:     MOV     R1,-(SP)                        ;                       5673
000054  012746  000000G                 MOV     #DBM21,-(SP)
000060  012746  000002                  MOV     #2,-(SP)
000064  010600                          MOV     SP,R0                           ; SP,*
```

```
000066  104417                          TRAP    17
000070  062706  000006                  ADD     #6,SP
000074  012601                  3$:     MOV     (SP)+,R1                ;                       5647
000076  000207                          RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 26520
; Maximum stack depth per invocation:  6 words


;   5675  1
;   5676  1
;   5677  1
```

```
;    5678  1     GLOBAL routine SEQUEN : novalue =
;    5679  1
;    5680  1     !+
;    5681  1     !       THIS ROUTINE IS CALLED BY DISK_RSP FOR EACH DISK MSCP RESPONSE MESSAGE
;    5682  1     !       WITH THE "SEQUENTIAL" MESSAGE TYPE. ITS GENERAL PURPOSE IS TO COPY THE
;    5683  1     !       CONTENTS OF THE MESSAGE PACKET INTO A RETURN PACKET SO THAT THE
;    5684  1     !       PACKET CAN BE RETURNED TO THE CONTROLLER.   IN ADDITION,
;    5685  1     !       IF THE COMMAND WAS AN I/O TRANSFER (READ, WRITE, OR ACCESS), THEN SOME
;    5686  1     !       FIELDS OF THE COMMAND PACKET ARE COPIED INTO THE RETURN PACKET.
;    5687  1     !
;    5688  1     !       IMPLICIT INPUTS:
;    5689  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5690  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    5691  1     !-
;    5692  1
;    5693  2         begin
;    5694  2
;    5695  2         local
;    5696  2             P_INDEX : signed word initial (-1),                          ! ASSUME NO ASSOCIATED COMMAND PKT
;    5697  2             R_INDEX : signed word,
;    5698  2             SRC_ADDR,
;    5699  2             DST_ADDR,
;    5700  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    5701  2
;    5702  2         incr COUNT from 0 to PKT_CNT - 1 do
;    5703  2
;    5704  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and          ! IF THIS IS THE ASSOC CMD
;    5705  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5706  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;    5707  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;    5708  2                (.MSCP_PKT [.COUNT, MSGTYP] eql MT_SEQ) and
;    5709  2                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END) and
;    5710  3                (.PKT_USE [.COUNT] eql .ICTLR)
;    5711  2             then
;    5712  3                 begin
;    5713  3                 P_INDEX = .COUNT;                                       ! SET PKT NUMBER
;    5714  3                 exitloop;
;    5715  2                 end;
;    5716  2
;    5717  2         if .P_INDEX lss 0                                               ! IF COMMAND NOT FOUND
;    5718  2         then
;    5719  3             begin
;    5720  3             PRINTF (DBM108, .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);      ! UNKNOWN COMMAND REF. NUMBER
;    5721  3             return;
;    5722  2             end;
;    5723  2
;    5724  3         if .MSCP_PKT [.P_INDEX, OPCODE] neq (.IPKT_ADDR [OPCODE] and (not OP_END))  ! IF OPCODE MISMATCH
;    5725  2         then
;    5726  2             PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO])
;
;    5727  2
;    5728  3         if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
;    5729  2             (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
;    5730  3            ((.IPKT_ADDR [STATUS_CODE] eql ST_SUC) and
```

```
;   5731  2              (.IPKT_ADDR [STATUS_SUBCODE] eql 0)) and
;   5732  3             ((.MSCP_PKT [.P_INDEX, BC_LO] neq .IPKT_ADDR [BC_LO]) or
;   5733  3              (.MSCP_PKT [.P_INDEX, BC_HI] neq .IPKT_ADDR [BC_HI]))
;   5734  2          then
; P 5735  2              PRINTF (DBM112,
; P 5736  2                      .MSCP_PKT [.P_INDEX, BC_HI], .MSCP_PKT [.P_INDEX, BC_LO], .IPKT_ADDR [BC_HI], .IPKT_ADDR [BC_LO],
;   5737  2                      .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);
;   5738  2
;   5739  2          if .MSCP_PKT [.P_INDEX, RSP_RECEIVED]
;   5740  2          then
;   5741  3              begin
;   5742  3              PRINTF (DBM120, .MSCP_PKT [.P_INDEX, CRN_HI], .MSCP_PKT [.P_INDEX, CRN_LO]);
;   5743  3              PUT_PKT (.P_INDEX);
;   5744  3              return;
;   5745  3              end
;   5746  2          else
;   5747  2              MSCP_PKT [.P_INDEX, RSP_RECEIVED] = TRUE;                    ! MARK RESPONSE RECEIVED
;   5748  2
;   5749  2          if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                         ! IF RETPKT IS NOT AVAILABLE
;   5750  2          then
;   5751  3              begin
;   5752  3              PRINTF (DBM22);                                              ! "SEQUEN: RETPKT NOT AVAILABLE"
;   5753  3              PUT_PKT (.P_INDEX);
;   5754  3              return;
;   5755  3              end
;   5756  2          else
;   5757  3              begin
;   5758  3              SRC_ADDR = .IPKT_ADDR + 6;                                   ! SET UP COPY (SKIP OVER PKT DESC)
;   5759  3              R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);        ! START OF ALLOCATED RETPKT
;   5760  3
;   5761  3              incr COUNT from 1 to RP_LEN do
;   5762  4                  begin
;   5763  4                  .DST_ADDR = ..SRC_ADDR;                                  ! COPY 1 WORD
;   5764  4                  DST_ADDR = .DST_ADDR + 2;                                ! ADVANCE DESTINATION ADDR
;   5765  4                  SRC_ADDR = .SRC_ADDR + 2;                                ! ADVANCE SOURCE ADDR
;   5766  4
;   5767  5                  if .IPKT_ADDR [OPCODE] eql (OP_ONL or OP_END)            ! IF THIS IS THE ONLINE END MESSAGE
;   5768  4                  then
;   5769  4                      if .COUNT eql 10                                     ! SKIP OVER RESERVED WORDS
;   5770  4                      then
;   5771  4                          SRC_ADDR = .SRC_ADDR + 4;                        !       IN ONLINE END - MESSAGE
;   5772  3                  end;                                                     ! COPY LOOP
;   5773  3
;   5774  3              R_ADDR [CTLR] = .ICTLR;                                      ! LOAD CONTROLLER NUMBER INTO PKT
;   5775  3
;   5776  3              if .P_INDEX geq 0                                            ! IF ASSOC. CMD PKT WAS FOUND
;   5777  3              then
;   5778  3
;   5779  3                  if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or        ! IF END MESSAGE IS
;   5780  3                     (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or       !       READ, WRITE, OR
;   5781  4                     (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))          !       ACCESS
;   5782  3                  then
;   5783  4                      begin
```

```
;   5784  4                 R_ADDR [CMDMOD] = .MSCP_PKT [.P_INDEX, MODIFY];          ! COPY
;   5785  4                 R_ADDR [CBCNT_LO] = .MSCP_PKT [.P_INDEX, BC_LO];         ! RELEVANT
;   5786  4                 R_ADDR [CBCNT_HI] = .MSCP_PKT [.P_INDEX, BC_HI];         ! FIELDS
;   5787  4                 R_ADDR [LBN_LO] = .MSCP_PKT [.P_INDEX, LBN_L];           ! FROM
;   5788  4                 R_ADDR [LBN_HI] = .MSCP_PKT [.P_INDEX, LBN_H];           ! COMMAND
;   5789  4                 R_ADDR [BUFF_0] = .MSCP_PKT [.P_INDEX, BUF_0];           ! PACKET
;   5790  4                 R_ADDR [BUFF_1] = .MSCP_PKT [.P_INDEX, BUF_1];           ! TO RETPKT
;   5791  3                 end;                                                     ! IF ENDCODE WAS READ/WRITE/ACCESS
;   5792  3
;   5793  3             IN_IODQ (.R_INDEX);                                          ! PUT RETPKT INDEX INTO IODQ
;   5794  2             end;                                                         ! IF RETPKT WAS ALLOCATED
;   5795  2
;   5796  2         if (.IPKT_ADDR [STATUS_CODE] neq  ST_SUC) or
;   5797  3            (.IPKT_ADDR [STATUS_SUBCODE] neq 0)
;   5798  2         then
;   5799  2             LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_OCCURED                ! SAVE ERROR CONDITION
;   5800  2         else
;   5801  2             LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_NOT_OCCURED;           !
;   5802  2
;   5803  2         LAST_PKT [.ICTLR, LAST_CRN_LO] = .IPKT_ADDR [CRN_LO];            ! SAVE COMMAND REFERENCE NUMBER
;   5804  2         LAST_PKT [.ICTLR, LAST_CRN_HI] = .IPKT_ADDR [CRN_HI];            !
;   5805  2         SCAN_ERRLOG ();                                                  ! PRINT ANY ASSOCIATED ERROR-LOGS
;   5806  2
;   5807  2         if .P_INDEX geq 0                                                ! IF ASSOC CMD PKT WAS FOUND
;   5808  2         then
;   5809  2             PUT_PKT (.P_INDEX);                                          ! RETURN COMMAND PACKET TO POOL
;   5810  2
;   5811  1         end;                                                             ! ROUTINE DISK_RSP


                                        .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G                 SEQUEN::JSR     R1,$SAVE5                   ;                                   5678
000004  005746                                  TST     -(SP)
000006  012746  177777                          MOV     #-1,-(SP)                   ; *,P.INDEX                         5693
000012  013701  000000G                         MOV     IPKT.ADDR,R1                ;                                   5704
000016  005002                                  CLR     R2                          ; COUNT                             5702
000020  010246                          1$:     MOV     R2,-(SP)                    ; COUNT,*                           5704
000022  012746  000106                          MOV     #106,-(SP)
000026  004737  000000G                         JSR     PC,BL$MUL
000032  022626                                  CMP     (SP)+,(SP)+
000034  026061  000012G 000012                  CMP     MSCP.PKT+12(R0),12(R1)
000042  001030                                  BNE     2$
000044  026061  000014G 000014                  CMP     MSCP.PKT+14(R0),14(R1)      ;                                   5705
000052  001024                                  BNE     2$
000054  026011  000000G                         CMP     MSCP.PKT(R0),(R1)           ;                                   5706
000060  001421                                  BEQ     2$
000062  105760  000022G                         TSTB    MSCP.PKT+22(R0)             ;                                   5707
000066  100416                                  BMI     2$
000070  132760  000360  000010G                 BITB    #360,MSCP.PKT+10(R0)        ;                                   5708
000076  001012                                  BNE     2$
000100  105761  000022                          TSTB    22(R1)                      ;                                   5709
000104  100007                                  BPL     2$
000106  116200  000000G                         MOVB    PKT.USE(R2),R0              ; *(COUNT),*                        5710
```

```
000112  020037  000104'                        CMP     R0,ICTLR
000116  001002                                 BNE     2$
000120  010216                                 MOV     R2,(SP)         ; COUNT,P.INDEX          5713
000122  000405                                 BR      3$              ;                        5712
000124  005202                          2$:    INC     R2              ; COUNT                  5702
000126  020227  000013                         CMP     R2,#13          ; COUNT,*
000132  003732                                 BLE     1$
000134  005716                                 TST     (SP)            ; P.INDEX                5717
000136  002013                          3$:    BGE     4$
000140  016146  000012                         MOV     12(R1),-(SP)    ;                        5720
000144  016146  000014                         MOV     14(R1),-(SP)
000150  012746  000000G                        MOV     #DBM108,-(SP)
000154  012746  000003                         MOV     #3,-(SP)
000160  010600                                 MOV     SP,R0           ; SP,*
000162  104417                                 TRAP    17
000164  000545                                 BR      9$              ;                        5721
000166  011646                          4$:    MOV     (SP),-(SP)      ; P.INDEX,*              5724
000170  012746  000106                         MOV     #106,-(SP)
000174  004737  000000G                        JSR     PC,BL$MUL
000200  010001                                 MOV     R0,R1
000202  022626                                 CMP     (SP)+,(SP)+
000204  013700  000000G                        MOV     IPKT.ADDR,R0
000210  116003  000022                         MOVB    22(R0),R3
000214  042703  177600                         BIC     #177600,R3
000220  005002                                 CLR     R2
000222  156102  000022G                        BISB    MSCP.PKT+22(R1),R2
000226  020203                                 CMP     R2,R3
000230  001422                                 BEQ     5$
000232  016046  000012                         MOV     12(R0),-(SP)    ;                        5726
000236  016046  000014                         MOV     14(R0),-(SP)
000242  005046                                 CLR     -(SP)
000244  116016  000022                         MOVB    22(R0),(SP)
000250  005046                                 CLR     -(SP)
000252  116116  000022G                        MOVB    MSCP.PKT+22(R1),(SP)
000256  012746  000000G                        MOV     #DBM111,-(SP)
000262  012746  000005                         MOV     #5,-(SP)
000266  010600                                 MOV     SP,R0           ; SP,*
000270  104417                                 TRAP    17
000272  062706  000014                         ADD     #14,SP
000276  013700  000000G                 5$:    MOV     IPKT.ADDR,R0    ;                        5728
000302  126027  000022  000241                 CMPB    22(R0),#241
000310  001404                                 BEQ     6$
000312  126027  000022  000242                 CMPB    22(R0),#242     ;                        5729
000320  001045                                 BNE     8$
000322  012702  000024                  6$:    MOV     #24,R2          ;                        5730
000326  060002                                 ADD     R0,R2
000330  132712  000037                         BITB    #37,(R2)
000334  001037                                 BNE     8$
000336  032712  177740                         BIT     #177740,(R2)    ;                        5731
000342  001034                                 BNE     8$
000344  026160  000026G 000026                 CMP     MSCP.PKT+26(R1),26(R0)  ;                5732
000352  001004                                 BNE     7$
000354  026160  000030G 000030                 CMP     MSCP.PKT+30(R1),30(R0)  ;                5733
```

```
000362  001424                          BEQ     8$
000364  016046  000012          7$:     MOV     12(R0),-(SP)            ;                               5737
000370  016046  000014                  MOV     14(R0),-(SP)
000374  016046  000026                  MOV     26(R0),-(SP)
000400  016046  000030                  MOV     30(R0),-(SP)
000404  016146  000026G                 MOV     MSCP.PKT+26(R1),-(SP)
000410  016146  000030G                 MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G                 MOV     #DBM112,-(SP)
000420  012746  000007                  MOV     #7,-(SP)
000424  010600                          MOV     SP,R0                   ; SP,*
000426  104417                          TRAP    17
000430  062706  000020                  ADD     #20,SP
000434  132761  000400  000005G  8$:     BITB    #400,MSCP.PKT+5(R1)     ;                               5739
000442  001422                          BEQ     10$
000444  016146  000012G                 MOV     MSCP.PKT+12(R1),-(SP)   ;                               5742
000450  016146  000014G                 MOV     MSCP.PKT+14(R1),-(SP)
000454  012746  000000G                 MOV     #DBM120,-(SP)
000460  012746  000003                  MOV     #3,-(SP)
000464  010600                          MOV     SP,R0                   ; SP,*
000466  104417                          TRAP    17
000470  016616  000010                  MOV     10(SP),(SP)             ; P.INDEX,*                     5743
000474  004737  000000G                 JSR     PC,PUT.PKT
000500  062706  000010          9$:     ADD     #10,SP                  ;                               5744
000504  000137  027760'                 JMP     21$                     ;                               5741
000510  112761  000001  000005G  10$:    MOVB    #1,MSCP.PKT+5(R1)       ;                               5747
000516  013746  000104'                 MOV     ICTLR,-(SP)             ;                               5749
000522  004737  000000G                 JSR     PC,GET.RETPKT
000526  010066  000004                  MOV     R0,4(SP)                ; *,R.INDEX
000532  005726                          TST     (SP)+
000534  005766  000002                  TST     2(SP)                   ; R.INDEX
000540  002007                          BGE     11$
000542  012746  000000G                 MOV     #DBM22,-(SP)            ;                               5752
000546  012746  000001                  MOV     #1,-(SP)
000552  010600                          MOV     SP,R0                   ; SP,*
000554  104417                          TRAP    17
000556  000563                          BR      19$                     ;                               5753
000560  013704  000000G          11$:    MOV     IPKT.ADDR,R4            ; *,SRC.ADDR                    5758
000564  062704  000006                  ADD     #6,R4                   ; *,SRC.ADDR
000570  016646  000002                  MOV     2(SP),-(SP)             ; R.INDEX,*                     5759
000574  012746  000054                  MOV     #54,-(SP)
000600  004737  000000G                 JSR     PC,BL$MUL
000604  062700  000000G                 ADD     #RETPKT,R0
000610  010005                          MOV     R0,R5                   ; *,DST.ADDR
000612  013702  000000G                 MOV     IPKT.ADDR,R2            ;                               5767
000616  012703  000001                  MOV     #1,R3                   ; *,COUNT                       5761
000622  012425          12$:    MOV     (R4)+,(R5)+             ; SRC.ADDR,DST.ADDR             5763
000624  126227  000022  000211          CMPB    22(R2),#211             ;                               5767
000632  001005                          BNE     13$
000634  020327  000012                  CMP     R3,#12                  ; COUNT,*                       5769
000640  001002                          BNE     13$
000642  062704  000004                  ADD     #4,R4                   ; *,SRC.ADDR                    5771
000646  005203          13$:    INC     R3                      ; COUNT                         5761
000650  020327  000026                  CMP     R3,#26                  ; COUNT,*
```

```
000654  003762                                  BLE     12$
000656  013703  000104'                         MOV     ICTLR,R3                ;                               5774
000662  042703  177760                          BIC     #177760,R3
000666  142760  000017  000002                  BICB    #17,2(R0)               ; *,*(R.ADDR)
000674  150360  000002                          BISB    R3,2(R0)                ; *,*(R.ADDR)
000700  005766  000004                          TST     4(SP)                   ; P.INDEX                       5776
000704  002441                                  BLT     15$
000706  005003                                  CLR     R3                      ;                               5779
000710  156203  000022                          BISB    22(R2),R3
000714  020327  000241                          CMP     R3,#241
000720  001406                                  BEQ     14$
000722  020327  000242                          CMP     R3,#242                 ;                               5780
000726  001403                                  BEQ     14$
000730  020327  000220                          CMP     R3,#220                 ;                               5781
000734  001025                                  BNE     15$
000736  016160  000024G 000012          14$:    MOV     MSCP.PKT+24(R1),12(R0)  ; *,*(R.ADDR)                   5784
000744  016160  000026G 000044                  MOV     MSCP.PKT+26(R1),44(R0)  ; *,*(R.ADDR)                   5785
000752  016160  000030G 000046                  MOV     MSCP.PKT+30(R1),46(R0)  ; *,*(R.ADDR)                   5786
000760  016160  000046G 000050                  MOV     MSCP.PKT+46(R1),50(R0)  ; *,*(R.ADDR)                   5787
000766  016160  000050G 000052                  MOV     MSCP.PKT+50(R1),52(R0)  ; *,*(R.ADDR)                   5788
000774  016160  000032G 000024                  MOV     MSCP.PKT+32(R1),24(R0)  ; *,*(R.ADDR)                   5789
001002  016160  000034G 000026                  MOV     MSCP.PKT+34(R1),26(R0)  ; *,*(R.ADDR)                   5790
001010  016616  000006                  15$:    MOV     6(SP),(SP)              ; R.INDEX,*                     5793
001014  004737  000000G                         JSR     PC,IN.IODQ
001020  005726                                  TST     (SP)+                   ;                               5757
001022  013716  000104'                         MOV     ICTLR,(SP)              ;                               5799
001026  012746  000006                          MOV     #6,-(SP)
001032  004737  000000G                         JSR     PC,BL$MUL
001036  013701  000000G                         MOV     IPKT.ADDR,R1            ;                               5796
001042  012703  000024                          MOV     #24,R3
001046  060103                                  ADD     R1,R3
001050  132713  000037                          BITB    #37,(R3)
001054  001003                                  BNE     16$
001056  032713  177740                          BIT     #177740,(R3)            ;                               5797
001062  001404                                  BEQ     17$
001064  012760  000001  000122'          16$:    MOV     #1,LAST.PKT(R0)         ;                               5799
001072  000402                                  BR      18$                     ;                               5796
001074  005060  000122'                  17$:    CLR     LAST.PKT(R0)            ;                               5801
001100  016160  000012  000124'          18$:    MOV     12(R1),LAST.PKT+2(R0)   ;                               5803
001106  016160  000014  000126'                  MOV     14(R1),LAST.PKT+4(R0)   ;                               5804
001114  004737  000000V                         JSR     PC,SCAN.ERRLOG          ;                               5805
001120  005766  000004                          TST     4(SP)                   ; P.INDEX                       5807
001124  002404                                  BLT     20$
001126  016616  000004                  19$:    MOV     4(SP),(SP)              ; P.INDEX,*                     5809
001132  004737  000000G                         JSR     PC,PUT.PKT
001136  022626                          20$:    CMP     (SP)+,(SP)+             ;                               5693
001140  022626                          21$:    CMP     (SP)+,(SP)+             ;                               5678
001142  000207                                  RTS     PC

; Routine Size:  306 words,     Routine Base:  $CODE$ + 26620
; Maximum stack depth per invocation:  18 words
```

```
;    5812  1     GLOBAL routine SCAN_ERRLOG : novalue =
;    5813  1
;    5814  1     !+
;    5815  1     ! THIS ROUTINE SCANS THE ERROR-LOG SAVE AREA AND PRINTS ANY ERROR-LOGS RECEIVED FOR THE ASSOCIATED RESPONSE
;    5816  1     !-
;    5817  1
;    5818  2         begin
;    5819  2
;    5820  2         local
;    5821  2             TEMP_UNIT,
;    5822  2             SFT_ERR_PRINTED : byte initial (byte (FALSE));
;    5823  2
;    5824  2         incr index from 0 to EP_CNT do                                  ! SCAN ERROR-LOG PACKET SAVE AREA
;    5825  3             begin
;    5826  3
;    5827  3             if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;    5828  3                (.ELOG_PKT [.index, EL_CRN_LO] eql .IPKT_ADDR [CRN_LO]) and
;    5829  3                (.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5830  4                (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;    5831  3             then
;    5832  4                 begin                                                  ! ERROR-LOG PENDING THIS RESPONSE
;    5833  4
;    5834  4                 if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED  ! IF SOFT ERROR OCCURED
;    5835  4                 then
;    5836  4
;    5837  4                     if .ELOG_PKT [.index, EL_FORMAT] lequ 4
;    5838  4                     then
;    5839  5                         begin
;    5840  5                         SOFT_ERROR (.index);                           ! UPATE SOFT ERROR COUNT
;    5841  5                         TEMP_UNIT = .L$LUN;                            ! SAVE UNIT NUMBER AS KNOWN TO DRS
;    5842  5
;    5843  5                         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    5844  5
;    5845  5                             if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_PKT [.index, EL_DK_NUM]) and
;    5846  6                                (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    5847  5                             then
;    5848  6                                 begin
;    5849  6                                 L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];   ! CORECT UNIT NO. FOR ERROR MESSAGE
;    5850  6                                 exitloop;
;    5851  5                                 end;
;    5852  5
;    5853  5                         case .ELOG_PKT [.index, EL_FORMAT] from 0 to 4 of
;    5854  5                             set
;    5855  5
;    5856  5                             [0]:    if .APT_MODE                       ! CONTROLLER ERROR
;    5857  5                                     then
;    5858  5                                         ERR_SOFT_RTNE_APT (50, .index)
;    5859  5                                     else
;    5860  5                                         ERR_SOFT_RTNE (50);
;    5861  5
;    5862  5                             [1]:    if .APT_MODE                       ! HOST MEMORY ACCESS ERROR
;    5863  5                                     then
;    5864  5                                         ERR_SOFT_RTNE_APT (51, .index)
```

```
;    5865  5                                  else
;    5866  5                                      ERR_SOFT_RTNE (51);
;    5867  5
;    5868  5                              [2]:    if .APT_MODE                                    ! DISK TRANSFER ERROR
;    5869  5                                      then
;    5870  5                                          ERR_SOFT_RTNE_APT (52, .index)
;    5871  5                                      else
;    5872  5                                          ERR_SOFT_RTNE (52);
;    5873  5
;    5874  5                              [3]:    if .APT_MODE                                    ! SDI ERROR
;    5875  5                                      then
;    5876  5                                          ERR_SOFT_RTNE_APT (53, .index)
;    5877  5                                      else
;    5878  5                                          ERR_SOFT_RTNE (53);
;    5879  5
;    5880  5                              [4]:    if .APT_MODE                                    ! SMALL DISK ERROR
;    5881  5                                      then
;    5882  5                                          ERR_SOFT_RTNE_APT (54, .index)
;    5883  5                                      else
;    5884  5                                          ERR_SOFT_RTNE (54);
;    5885  5                              tes;
;    5886  5
;    5887  5                          L$LUN = .TEMP_UNIT;                                         ! RESTORE UNIT NUMBER
;    5888  5                          SFT_ERR_PRINTED = TRUE;                                     ! SOFT ERROR PRINTOUT OCCURED
;    5889  5                          end
;    5890  4                      else
;    5891  4                          PRINTF (DBM109, .ELOG_PKT [.index, EL_FORMAT]);             ! UNKNOWN ERROR-LOG FORMAT
;    5892  4
;    5893  5                  if not (.SFT_ERR_PRINTED)
;    5894  4                  then
;    5895  4                      PRINTB (CRLF);                                                  ! EXTRA CARRIAGE-RETURN/LINE-FEED
;    5896  4
;    5897  4                      EMS_EL (.index);                                                ! PRINT ERROR-LOG CONTENTS
;    5898  4                      end
;    5899  3                  else
;    5900  3
;    5901  3                      if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;    5902  4                         ((.ELOG_PKT [.index, EL_CRN_HI] lssu .IPKT_ADDR [CRN_HI]) or
;    5903  5                          ((.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5904  3                           (.ELOG_PKT [.index, EL_CRN_LO] lssu .IPKT_ADDR [CRN_LO]))) and
;    5905  4                         (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;    5906  3                      then
;    5907  4                          begin
;    5908  4                          PRINTB (CRLF);                                              ! CARRIAGE-RETURN/LINE-FEED
;    5909  4                          EMS_EL (.index);                                            ! PRINT ERROR-LOG CONTENTS
;    5910  3                          end;
;    5911  3
;    5912  2                  end;                                                                ! ERROR-LOG SAVE AREA SCAN
;    5913  2
;    5914  1          end;


                                        .SBTTL  SCAN.ERRLOG RDRX INTERRUPT SERVICE ROUTINES
```

F2


```
000000  004137  000000G                SCAN.ERRLOG::
                                        JSR     R1,$SAVE5                       ;                               5812
000004  005746                          TST     -(SP)
000006  105005                          CLRB    R5                              ; SFT.ERR.PRINTED               5818
000010  005002                          CLR     R2                              ; INDEX                         5824
000012  010246                  1$:     MOV     R2,-(SP)                        ; INDEX,*                       5827
000014  012746  000102                  MOV     #102,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  010001                          MOV     R0,R1
000026  022626                          CMP     (SP)+,(SP)+
000030  012703  000000G                 MOV     #ELOG.PKT,R3
000034  060103                          ADD     R1,R3
000036  005004                          CLR     R4
000040  005000                          CLR     R0
000042  151300                          BISB    (R3),R0
000044  020037  000104'                 CMP     R0,ICTLR
000050  001016                          BNE     2$
000052  005204                          INC     R4
000054  013700  000000G                 MOV     IPKT.ADDR,R0                    ;                               5828
000060  026160  000006G 000012          CMP     ELOG.PKT+6(R1),12(R0)
000066  001007                          BNE     2$
000070  026160  000010G 000014          CMP     ELOG.PKT+10(R1),14(R0)          ;                               5829
000076  001003                          BNE     2$
000100  126327  000001  000001          CMPB    1(R3),#1                        ;                               5830
000106  001402                  2$:     BEQ     3$
000110  000137  030524'                 JMP     25$
000114  013746  000104'         3$:     MOV     ICTLR,-(SP)                     ;                               5834
000120  012746  000006                  MOV     #6,-(SP)
000124  004737  000000G                 JSR     PC,BL$MUL
000130  022626                          CMP     (SP)+,(SP)+
000132  005760  000122'                 TST     LAST.PKT(R0)
000136  001161                          BNE     23$
000140  126127  000016G 000004          CMPB    ELOG.PKT+16(R1),#4              ;                               5837
000146  101142                          BHI     21$
000150  010246                          MOV     R2,-(SP)                        ; INDEX,*                       5840
000152  004737  000000V                 JSR     PC,SOFT.ERROR
000156  013766  000000G 000002          MOV     L$LUN,2(SP)                     ; *,TEMP.UNIT                   5841
000164  012703  000006                  MOV     #6,R3                           ; *,OFFSET                      5843
000170  010300                  4$:     MOV     R3,R0                           ; OFFSET,*                      5845
000172  063700  000076'                 ADD     ICST.ADDR,R0
000176  016146  000012G                 MOV     ELOG.PKT+12(R1),-(SP)
000202  111004                          MOVB    (R0),R4
000204  042704  177760                  BIC     #177760,R4
000210  020426                          CMP     R4,(SP)+
000212  001012                          BNE     5$
000214  032710  040000                  BIT     #40000,(R0)                     ;                               5846
000220  001407                          BEQ     5$
000222  011004                          MOV     (R0),R4                         ;                               5849
000224  000304                          SWAB    R4
000226  042704  177760                  BIC     #177760,R4
000232  010437  000000G                 MOV     R4,L$LUN
000236  000405                          BR      6$                              ;                               5848
000240  062703  000024          5$:     ADD     #24,R3                          ; *,OFFSET                      5843
```

```
000244  020327  000102                          CMP     R3,#102                         ; OFFSET,*
000250  003747                                   BLE     4$
000252  005000                          6$:      CLR     R0                              ;                       5856
000254  153700  001166'                         BISB    APT.MODE,R0
000260  116101  000016G                         MOVB    ELOG.PKT+16(R1),R1              ;                       5853
000264  042701  177400                          BIC     #177400,R1
000270  006301                                   ASL     R1
000272  066107  000000'                         ADD     P.AAA(R1),PC                    ; Case dispatch
000276  032700  000001                  8$:     BIT     #1,R0                           ;                       5856
000302  001403                                   BEQ     9$
000304  012716  000062                          MOV     #62,(SP)                        ;                       5858
000310  000442                                   BR      17$
000312  012716  000062                  9$:     MOV     #62,(SP)                        ;                       5860
000316  000446                                   BR      19$
000320  032700  000001                  10$:    BIT     #1,R0                           ;                       5862
000324  001403                                   BEQ     11$
000326  012716  000063                          MOV     #63,(SP)                        ;                       5864
000332  000431                                   BR      17$
000334  012716  000063                  11$:    MOV     #63,(SP)                        ;                       5866
000340  000435                                   BR      19$
000342  032700  000001                  12$:    BIT     #1,R0                           ;                       5868
000346  001403                                   BEQ     13$
000350  012716  000064                          MOV     #64,(SP)                        ;                       5870
000354  000420                                   BR      17$
000356  012716  000064                  13$:    MOV     #64,(SP)                        ;                       5872
000362  000424                                   BR      19$
000364  032700  000001                  14$:    BIT     #1,R0                           ;                       5874
000370  001403                                   BEQ     15$
000372  012716  000065                          MOV     #65,(SP)                        ;                       5876
000376  000407                                   BR      17$
000400  012716  000065                  15$:    MOV     #65,(SP)                        ;                       5878
000404  000413                                   BR      19$
000406  006000                          16$:    ROR     R0                              ;                       5880
000410  103007                                   BCC     18$
000412  012716  000066                          MOV     #66,(SP)                        ;                       5882
000416  010246                          17$:    MOV     R2,-(SP)                        ; INDEX,*
000420  004737  000000V                         JSR     PC,ERR.SOFT.RTNE.APT
000424  005726                                   TST     (SP)+
000426  000404                                   BR      20$                             ;                       5880
000430  012716  000066                  18$:    MOV     #66,(SP)                        ;                       5884
000434  004737  000000V                 19$:    JSR     PC,ERR.SOFT.RTNE
000440  016637  000002  000000G         20$:    MOV     2(SP),L$LUN                     ; TEMP.UNIT,*           5887
000446  112705  000001                          MOVB    #1,R5                           ; *,SFT.ERR.PRINTED     5888
000452  000412                                   BR      22$                             ;                       5837
000454  005046                          21$:    CLR     -(SP)                           ;                       5891
000456  116116  000016G                         MOVB    ELOG.PKT+16(R1),(SP)
000462  012746  000000G                         MOV     #DBM109,-(SP)
000466  012746  000002                          MOV     #2,-(SP)
000472  010600                                   MOV     SP,R0                           ; SP,*
000474  104417                                   TRAP    17
000476  022626                                   CMP     (SP)+,(SP)+
000500  005726                          22$:    TST     (SP)+                           ;                       5837
000502  032705  000001                  23$:    BIT     #1,R5                           ; *,SFT.ERR.PRINTED     5893
```

```
000506  001007                          BNE     24$
000510  012746  000000G                 MOV     #CRLF,-(SP)             ;                               5895
000514  012746  000001                  MOV     #1,-(SP)
000520  010600                          MOV     SP,R0                   ; SP,*
000522  104414                          TRAP    14
000524  022626                          CMP     (SP)+,(SP)+
000526  010246                  24$:    MOV     R2,-(SP)                ; INDEX,*                       5897
000530  004737  000000G                 JSR     PC,EMS.EL
000534  005726                          TST     (SP)+                   ;                               5832
000536  000433                          BR      27$                     ;                               5827
000540  006004                  25$:    ROR     R4                      ;                               5901
000542  103031                          BCC     27$
000544  013700  000000G                 MOV     IPKT.ADDR,R0            ;                               5902
000550  026160  000010G 000014          CMP     ELOG.PKT+10(R1),14(R0)
000556  103405                          BLO     26$
000560  001022                          BNE     27$                     ;                               5903
000562  026160  000006G 000012          CMP     ELOG.PKT+6(R1),12(R0)   ;                               5904
000570  103016                          BHIS    27$
000572  126327  000001  000001  26$:    CMPB    1(R3),#1                ;                               5905
000600  001012                          BNE     27$
000602  012746  000000G                 MOV     #CRLF,-(SP)             ;                               5908
000606  012746  000001                  MOV     #1,-(SP)
000612  010600                          MOV     SP,R0                   ; SP,*
000614  104414                          TRAP    14
000616  010216                          MOV     R2,(SP)                 ; INDEX,*                       5909
000620  004737  000000G                 JSR     PC,EMS.EL
000624  022626                          CMP     (SP)+,(SP)+             ;                               5907
000626  005202                  27$:    INC     R2                      ; INDEX                         5824
000630  020227  000014                  CMP     R2,#14                  ; INDEX,*
000634  003002                          BGT     28$
000636  000137  027776'                 JMP     1$
000642  005726                  28$:    TST     (SP)+                   ;                               5812
000644  000207                          RTS     PC

; Routine Size:  211 words,    Routine Base:  $CODE$ + 27764
; Maximum stack depth per invocation:  12 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:                                  ; CASE Table for SCAN.ERRLOG+0272       5853
000000  000000                  7$:     .WORD   0                       ; [8$]
000002  000022                          .WORD   22                      ; [10$]
000004  000044                          .WORD   44                      ; [12$]
000006  000066                          .WORD   66                      ; [14$]
000010  000110                          .WORD   110                     ; [16$]
```

```
;    5915  1     GLOBAL routine DATAGM : novalue =
;    5916  1
;    5917  1     !+
;    5918  1     !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;    5919  1     !       THE RDRX
;    5920  1     !
;    5921  1     !       IMPLICIT INPUTS:
;    5922  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
;    5923  1     !                           MESSAGE
;    5924  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;    5925  1     !-
;    5926  1
;    5927  2         begin
;    5928  2
;    5929  2         local
;    5930  2             index : signed word initial (-1),
;    5931  2             SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;    5932  2             SRC_ADDR,
;    5933  2             DST_ADDR,
;    5934  2             TEMP_UNIT,
;    5935  2             SFT_ERR_PRINTED : byte initial (byte (FALSE)),
;    5936  2             PACKET_LEN : word;
;    5937  2
;    5938  2     !
;    5939  2     ! FIND AN EMPTY SLOT IN THE ERROR-LOG PACKET SAVE AREA
;    5940  2     !
;    5941  2
;    5942  2        incr COUNT from 0 to EP_CNT - 1 do
;    5943  2
;    5944  2             if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY                    ! IF EMPTY SLOT FOUND
;    5945  2             then
;    5946  3                 begin
;    5947  3                 index = .COUNT;                                             ! SAVE INDEX INTO THE SAVE AREA
;    5948  3                 exitloop;
;    5949  2                 end;
;    5950  2
;    5951  2          if .index lss 0
;    5952  2          then
;    5953  2              index = EP_CNT;                                                ! IF NO SLOT FOUND, USE LAST SPARE SLOT
;    5954  2
;    5955  2     !
;    5956  2     ! SAVE THE PACKET CONTENTS
;    5957  2     !
;    5958  2
;    5959  2          SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                     ! ADDRESS OF THE SAVE AREA
;    5960  2          SAVE_ADDR [EL_CONTENTS ] = FULL;                                  ! MARK IT FULL
;    5961  2          SAVE_ADDR [EL_CNTR] = .ICTLR;                                     ! OWNERSHIP
;    5962  2          SRC_ADDR = .IPKT_ADDR + 6;                                        ! SETUP COPY ADDRESSES
;    5963  2          DST_ADDR = .SAVE_ADDR + 2;                                        !
;    5964  2          PACKET_LEN = ((.IPKT_ADDR [MSGLEN] + 1) / 2) * 2;                 ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
;    5965  2
;    5966  2          if .PACKET_LEN gtru EP_LEN - 1
;    5967  2          then
```

```
;    5968  2              PACKET_LEN = EP_LEN - 1;                                    ! ADJUST LENGTH, IF TOO LONG
;    5969  2
;    5970  2          incr COUNT from 1 to .PACKET_LEN do
;    5971  3              begin
;    5972  3              .DST_ADDR = ..SRC_ADDR;                                     ! COPY A WORD
;    5973  3              SRC_ADDR = .SRC_ADDR + 2;                                   ! UPDATE ADDRESS POINTERS
;    5974  3              DST_ADDR = .DST_ADDR + 2;                                   !
;    5975  2              end;
;    5976  2
;    5977  2          !
;    5978  2          ! CHECK IF THE CORRESPONDING RESPONSE HAS ALREADY BEEN RECEIVED
;    5979  2          !
;    5980  2
;    5981  2          if (.SAVE_ADDR [EL_CRN_LO] eql .LAST_PKT [.ICTLR, LAST_CRN_LO]) and
;    5982  3             (.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI])
;    5983  2          then
;    5984  3              begin                                                       ! LOG REFERS TO THE LAST RESPONSE RECEIVED
;    5985  3
;    5986  3              if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED     ! IF SOFT ERROR HAD OCCURED
;    5987  3              then
;    5988  3
;    5989  3                  if .SAVE_ADDR [EL_FORMAT] lequ 4
;    5990  3                  then
;    5991  4                      begin
;    5992  4                      SOFT_ERROR (.index);                                ! UPDATE SOFT ERROR COUNT
;    5993  4                      TEMP_UNIT = .L$LUN;                                 ! SAVE UNIT NUMBER AS KNOWN TO DRS
;    5994  4
;    5995  4                      incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    5996  4
;    5997  4                          if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;    5998  5                             (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    5999  4                          then
;    6000  5                              begin
;    6001  5                              L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT]; ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;    6002  5                              exitloop;
;    6003  4                              end;
;    6004  4
;    6005  4                      case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;    6006  4                          set
;    6007  4
;    6008  4
;    6009  4                          [0] :       if .APT_MODE                       ! CONTROLLER ERROR
;    6010  4                                      then
;    6011  4                                          ERR_SOFT_RTNE_APT (50, .index)
;    6012  4                                      else
;    6013  4                                          ERR_SOFT_RTNE (50);
;    6014  4
;    6015  4                          [1] :       if .APT_MODE                       ! HOST MEMORY ACCESS ERROR
;    6016  4                                      then
;    6017  4                                          ERR_SOFT_RTNE_APT (51, .index)
;    6018  4                                      else
;    6019  4                                          ERR_SOFT_RTNE (51);
;    6020  4
```

```
;  6021  4                        [2] :       if .APT_MODE                                    ! DISK TRANSFER ERROR
;  6022  4                                    then
;  6023  4                                        ERR_SOFT_RTNE_APT (52, .index)
;  6024  4                                    else
;  6025  4                                        ERR_SOFT_RTNE (52);
;  6026  4
;  6027  4                        [3] :       if .APT_MODE                                    ! SDI ERROR
;  6028  4                                    then
;  6029  4                                        ERR_SOFT_RTNE_APT (53, .index)
;  6030  4                                    else
;  6031  4                                        ERR_SOFT_RTNE (53);
;  6032  4
;  6033  4                        [4] :       if .APT_MODE                                    ! SMALL DISK ERROR
;  6034  4                                    then
;  6035  4                                        ERR_SOFT_RTNE_APT (54, .index)
;  6036  4                                    else
;  6037  4                                        ERR_SOFT_RTNE (54);
;  6038  4                        tes;
;  6039  4
;  6040  4
;  6041  4                    L$LUN = .TEMP_UNIT;                                             ! RESTORE UNIT NUMBER
;  6042  4                    SFT_ERR_PRINTED = TRUE;                                         ! SOFT ERROR PRINTOUT OCCURED
;  6043  4                    end
;  6044  4
;  6045  3                else
;  6046  3                    PRINTF (DBM109, .SAVE_ADDR [EL_FORMAT]);                        ! ERROR LOG FORMAT UNKNOWN
;  6047  3
;  6048  4            if not (.SFT_ERR_PRINTED)
;  6049  4
;  6050  3                then
;  6051  3                    PRINTB (CRLF);                                                  ! EXTRA CARRIEGE-RETURN/LINE-FEED
;  6052  3
;  6053  3                EMS_EL (.index);                                                    ! PRINT PACKET CONTENTS
;  6054  3                end                                                             ! CORRESPONDING RESPONSE RECEIVED
;  6055  3
;  6056  2            else
;  6057  2
;  6058  2                if (.SAVE_ADDR [EL_CRN_HI] lssu .LAST_PKT [.ICTLR, LAST_CRN_HI]) or
;  6059  3                   ((.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI]) and
;  6060  3                    (.SAVE_ADDR [EL_CRN_LO] lssu .LAST_PKT [.ICTLR, LAST_CRN_LO]))
;  6061  3
;  6062  2                then
;  6063  3                    begin                                                       ! LOG REFERS TO SOME PREVIOUS RESPONSE
;  6064  3                    PRINTB (CRLF);                                              ! CARRIAGE-RETURN/LINE-FEED
;  6065  3                    EMS_EL (.index);                                            ! PRINT PACKET CONTENTS
;  6066  2                    end;
;  6067  2
;  6068  1            end;


                                        .SBTTL  DATAGM RDRX INTERRUPT SERVICE ROUTINES
030632                                  .PSECT  $CODE$,  RO
```

```
000000  004137  000000G                DATAGM::JSR     R1,$SAVE5                       ;                                       5915
000004  012704  177777                          MOV     #-1,R4                          ; *,INDEX                               5927
000010  105046                                  CLRB    -(SP)                           ; SFT.ERR.PRINTED
000012  005001                                  CLR     R1                              ; COUNT                                 5942
000014  010146                          1$:     MOV     R1,-(SP)                        ; COUNT,*                               5944
000016  012746  000102                        MOV     #102,-(SP)
000022  004737  000000G                       JSR     PC,BL$MUL
000026  022626                                CMP     (SP)+,(SP)+
000030  105760  000001G                       TSTB    ELOG.PKT+1(R0)
000034  001002                                BNE     2$
000036  010104                                MOV     R1,R4                           ; COUNT,INDEX                           5947
000040  000405                                BR      3$                              ;                                       5946
000042  005201                          2$:     INC     R1                              ; COUNT                                 5942
000044  020127  000013                        CMP     R1,#13                          ; COUNT,*
000050  003761                                BLE     1$
000052  005704                                TST     R4                              ; INDEX                                 5951
000054  002002                          3$:     BGE     4$
000056  012704  000014                        MOV     #14,R4                          ; *,INDEX                               5953
000062  010446                          4$:     MOV     R4,-(SP)                        ; INDEX,*                               5959
000064  012746  000102                        MOV     #102,-(SP)
000070  004737  000000G                       JSR     PC,BL$MUL
000074  062700  000000G                       ADD     #ELOG.PKT,R0
000100  010001                                MOV     R0,R1                           ; *,SAVE.ADDR
000102  111761  000001                        MOVB    (PC),1(R1)                      ; *,*(SAVE.ADDR)                        5960
000106  113711  000104'                      MOVB    ICTLR,(R1)                      ; *,SAVE.ADDR                           5961
000112  013700  000000G                       MOV     IPKT.ADDR,R0                    ;                                       5962
000116  012705  000006                        MOV     #6,R5                           ; *,SRC.ADDR
000122  060005                                ADD     R0,R5                           ; *,SRC.ADDR
000124  012703  000002                        MOV     #2,R3                           ; *,DST.ADDR                            5963
000130  060103                                ADD     R1,R3                           ; SAVE.ADDR,DST.ADDR
000132  016016  000006                        MOV     6(R0),(SP)                      ;                                       5964
000136  005216                                INC     (SP)
000140  012746  000002                        MOV     #2,-(SP)
000144  004737  000000G                       JSR     PC,BL$DIV
000150  062700  000002                        ADD     #2,R0
000154  020027  000040                        CMP     R0,#40                          ; PACKET.LEN,*                          5966
000160  101402                                BLOS    5$
000162  012700  000040                        MOV     #40,R0                          ; *,PACKET.LEN                          5968
000166  005002                          5$:     CLR     R2                              ; COUNT                                 5970
000170  000401                                BR      7$
000172  012523                          6$:     MOV     (R5)+,(R3)+                     ; SRC.ADDR,DST.ADDR                     5972
000174  005202                          7$:     INC     R2                              ; COUNT                                 5970
000176  020200                                CMP     R2,R0                           ; COUNT,PACKET.LEN
000200  003774                                BLE     6$
000202  013716  000104'                      MOV     ICTLR,(SP)                      ;                                       5981
000206  012746  000006                        MOV     #6,-(SP)
000212  004737  000000G                       JSR     PC,BL$MUL
000216  005726                                TST     (SP)+
000220  026160  000006  000124'               CMP     6(R1),LAST.PKT+2(R0)            ; *(SAVE.ADDR),*
000226  001402                                BEQ     8$
000230  000137  031474'                       JMP     30$
000234  026160  000010  000126'         8$:     CMP     10(R1),LAST.PKT+4(R0)           ; *(SAVE.ADDR),*                        5982
000242  001402                                BEQ     9$
```

```
000244  000137  031502'                  JMP     31$
000250  005760  000122'          9$:     TST     LAST.PKT(R0)                    ;                                   5986
000254  001153                           BNE     28$
000256  005003                           CLR     R3                              ;                                   5989
000260  156103  000016                   BISB    16(R1),R3                       ; *(SAVE.ADDR),*
000264  020327  000004                   CMP     R3,#4
000270  101135                           BHI     27$
000272  010416                           MOV     R4,(SP)                         ; INDEX,*                           5992
000274  004737  000000V                  JSR     PC,SOFT.ERROR
000300  013705  000000G                  MOV     L$LUN,R5                        ; *,TEMP.UNIT                       5993
000304  012702  000006                   MOV     #6,R2                           ; *,OFFSET                          5995
000310  010200                   10$:    MOV     R2,R0                           ; OFFSET,*                          5997
000312  063700  000076'                  ADD     ICST.ADDR,R0
000316  016146  000012                   MOV     12(R1),-(SP)                    ; *(SAVE.ADDR),*
000322  111046                           MOVB    (R0),-(SP)
000324  042716  177760                   BIC     #177760,(SP)
000330  022626                           CMP     (SP)+,(SP)+
000332  001012                           BNE     11$
000334  032710  040000                   BIT     #40000,(R0)                     ;                                   5998
000340  001407                           BEQ     11$
000342  011046                           MOV     (R0),-(SP)                      ;                                   6001
000344  000316                           SWAB    (SP)
000346  042716  177760                   BIC     #177760,(SP)
000352  012637  000000G                  MOV     (SP)+,L$LUN
000356  000405                           BR      12$                             ;                                   6000
000360  062702  000024           11$:    ADD     #24,R2                          ; *,OFFSET                          5995
000364  020227  000102                   CMP     R2,#102                         ; OFFSET,*
000370  003747                           BLE     10$
000372  005000                   12$:    CLR     R0                              ;                                   6009
000374  153700  001166'                  BISB    APT.MODE,R0
000400  006303                           ASL     R3                              ;                                   6005
000402  066307  000012'                  ADD     P.AAB(R3),PC                    ; Case dispatch
000406  032700  000001           14$:    BIT     #1,R0                           ;                                   6009
000412  001403                           BEQ     15$
000414  012716  000062                   MOV     #62,(SP)                        ;                                   6011
000420  000442                           BR      23$
000422  012716  000062           15$:    MOV     #62,(SP)                        ;                                   6013
000426  000446                           BR      25$
000430  032700  000001           16$:    BIT     #1,R0                           ;                                   6015
000434  001403                           BEQ     17$
000436  012716  000063                   MOV     #63,(SP)                        ;                                   6017
000442  000431                           BR      23$
000444  012716  000063           17$:    MOV     #63,(SP)                        ;                                   6019
000450  000435                           BR      25$
000452  032700  000001           18$:    BIT     #1,R0                           ;                                   6021
000456  001403                           BEQ     19$
000460  012716  000064                   MOV     #64,(SP)                        ;                                   6023
000464  000420                           BR      23$
000466  012716  000064           19$:    MOV     #64,(SP)                        ;                                   6025
000472  000424                           BR      25$
000474  032700  000001           20$:    BIT     #1,R0                           ;                                   6027
000500  001403                           BEQ     21$
000502  012716  000065                   MOV     #65,(SP)                        ;                                   6029
```

```
000506  000407                          BR      23$
000510  012716  000065          21$:    MOV     #65,(SP)                ;                               6031
000514  000413                          BR      25$
000516  006000                  22$:    ROR     R0                      ;                               6033
000520  103007                          BCC     24$
000522  012716  000066                  MOV     #66,(SP)                ;                               6035
000526  010446                  23$:    MOV     R4,-(SP)                ; INDEX,*
000530  004737  000000V                 JSR     PC,ERR.SOFT.RTNE.APT
000534  005726                          TST     (SP)+
000536  000404                          BR      26$                     ;                               6033
000540  012716  000066          24$:    MOV     #66,(SP)                ;                               6037
000544  004737  000000V          25$:    JSR     PC,ERR.SOFT.RTNE
000550  010537  000000G          26$:    MOV     R5,L$LUN                ; TEMP.UNIT,*                   6041
000554  112766  000001  000006          MOVB    #1,6(SP)                ; *,SFT.ERR.PRINTED             6042
000562  000410                          BR      28$                     ;                               5989
000564  010316                  27$:    MOV     R3,(SP)                 ;                               6046
000566  012746  000000G                 MOV     #DBM109,-(SP)
000572  012746  000002                  MOV     #2,-(SP)
000576  010600                          MOV     SP,R0                   ; SP,*
000600  104417                          TRAP    17
000602  022626                          CMP     (SP)+,(SP)+
000604  032766  000001  000006  28$:    BIT     #1,6(SP)                ; *,SFT.ERR.PRINTED             6048
000612  001007                          BNE     29$
000614  012716  000000G                 MOV     #CRLF,(SP)              ;                               6051
000620  012746  000001                  MOV     #1,-(SP)
000624  010600                          MOV     SP,R0                   ; SP,*
000626  104414                          TRAP    14
000630  005726                          TST     (SP)+
000632  010416                  29$:    MOV     R4,(SP)                 ; INDEX,*                       6053
000634  004737  000000G                 JSR     PC,EMS.EL
000640  000426                          BR      33$                     ;                               5981
000642  026160  000010  000126'  30$:    CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*                6058
000650  103410                  31$:    BLO     32$
000652  026160  000010  000126'          CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*                6059
000660  001016                          BNE     33$
000662  026160  000006  000124'          CMP     6(R1),LAST.PKT+2(R0)    ; *(SAVE.ADDR),*                6060
000670  103012                          BHIS    33$
000672  012716  000000G          32$:    MOV     #CRLF,(SP)              ;                               6064
000676  012746  000001                  MOV     #1,-(SP)
000702  010600                          MOV     SP,R0                   ; SP,*
000704  104414                          TRAP    14
000706  010416                          MOV     R4,(SP)                 ; INDEX,*                       6065
000710  004737  000000G                 JSR     PC,EMS.EL
000714  005726                          TST     (SP)+                   ;                               6063
000716  062706  000010          33$:    ADD     #10,SP                  ;                               5915
000722  000207                          RTS     PC

; Routine Size:  234 words,     Routine Base:  $CODE$ + 30632
; Maximum stack depth per invocation:  14 words


000012                                  .PSECT  $PLIT$,  RO ,  D
```

```
                                P.AAB:                                  ; CASE Table for DATAGM+0402                    6005
000012  000000                  13$:    .WORD   0                       ; [14$]
000014  000022                          .WORD   22                      ; [16$]
000016  000044                          .WORD   44                      ; [18$]
000020  000066                          .WORD   66                      ; [20$]
000022  000110                          .WORD   110                     ; [22$]


;   6069  1
;   6070  1
;   6071  1
```

```
;    6072  1     GLOBAL routine SOFT_ERROR (index) : novalue =
;    6073  1
;    6074  1     !+
;    6075  1     !       THIS ROUTINE UPDATES THE SOFT ERROR COUNT IN THE TALLY TABLE FOR EACH
;    6076  1     !       ERROR LOG MESSAGE RECEIVED
;    6077  1     !
;    6078  1     !       IMPLICIT INPUTS:
;    6079  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;    6080  1     !-
;    6081  1
;    6082  2         begin
;    6083  2
;    6084  2         local
;    6085  2             FOUND: byte initial (byte (FALSE)),
;    6086  2             SOFT_OCCURED : byte initial (byte (FALSE)),
;    6087  2             UNIT: word,
;    6088  2             ERROR_CODE : byte,
;    6089  2             ERROR_SUB : word,
;    6090  2             RETRIES : word,
;    6091  2             TALLY_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    6092  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;    6093  2
;    6094  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                   ! ADDR OF ERROR PKT
;    6095  2         ERROR_CODE = .ELOG_ADDR [EL_CODE];                              ! ERROR CODE
;    6096  2         ERROR_SUB = .ELOG_ADDR [EL_SUBCODE];                            ! ERROR SUBCODE
;    6097  2
;    6098  2         if (BIT_TST (SWP_FLAGS, SWF_TRY)) and
;    6099  3            (.ELOG_ADDR [EL_FORMAT] eql 2)
;    6100  2         then
;    6101  2             RETRIES = .ELOG_ADDR [EL_RETRY]                             ! COUNT EACH RETRY
;    6102  2         else
;    6103  2             RETRIES = 1;                                                ! IGNORE RETRIES
;    6104  2
;    6105  2         if .RETRIES eql 0                                               ! IN CASE OF A BUG
;    6106  2         then
;    6107  2             RETRIES = 1;
;    6108  2
;    6109  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    6110  2
;    6111  2             if (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and  ! DISK TO UNIT NO.
;    6112  3                (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_ADDR [EL_DK_NUM])
;    6113  2             then
;    6114  3                 begin
;    6115  3                 FOUND = TRUE;
;    6116  3                 UNIT = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];          ! DISK'S UNIT NO.
;    6117  3                 exitloop;
;    6118  2                 end;
;    6119  2
;    6120  2     !   if (.ELOG_ADDR [EL_SUCCESS]) or
;    6121  2     !      (.ELOG_ADDR [EL_CONTINUE])
;    6122  2     !   then
;    6123  2             SOFT_OCCURED = TRUE;                                        ! SOFT ERROR FLAG
;    6124  2
```

```
;   6125  2          if .FOUND                                                                   ! IF UNIT FOUND
;   6126  2          then
;   6127  3              begin
;   6128  3              TALLY_ADDR = TALLY + (.UNIT * TALLY_LEN * 2);                           ! ADDR OF TALLY TBL
;   6129  3
;   6130  3              if .SOFT_OCCURED                                                        ! FOR SOFT ERRORRS
;   6131  3              then
;   6132  3                  selectoneu .ERROR_CODE of
;   6133  3                      set
;   6134  3
;   6135  3                      [ST_MFE]:       TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES;! SOFT-MEDIA FORMAT
;   6136  3
;   6137  3                      [ST_DAT]:       if .ERROR_SUB eql 2                             ! SOFT-DATA
;   6138  3                                      then
;   6139  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;   6140  3                                      else
;   6141  3                                          TALLY_ADDR [ERR_SFT_DAT] = .TALLY_ADDR [ERR_SFT_DAT] + .RETRIES;
;   6142  3
;   6143  3                      [ST_HST]:       TALLY_ADDR [ERR_SFT_HST] = .TALLY_ADDR [ERR_SFT_HST] + .RETRIES;! SOFT-HOST ACCESS
;   6144  3
;   6145  3                      [ST_CNT]:       C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + .RETRIES;
;   6146  3                                                                                      ! SOFT-CONTROLLER
;   5147  3
;   6148  3                      [ST_DRV]:       if .ERROR_SUB eql 3                             ! SOFT_DRIVE
;   6149  3                                      then
;   6150  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;   6151  3                                      else
;   6152  3                                          TALLY_ADDR [ERR_SFT_DRV] = .TALLY_ADDR [ERR_SFT_DRV] + .RETRIES;
;   6153  3                      tes
;   6154  3              else
;   6155  3
;   6156  3                  if (.ELOG_ADDR [EL_CRN_LO] eql 0) and
;   6157  4                     (.ELOG_ADDR [EL_CRN_HI] eql 0)
;   6158  3                  then
;   6159  3                      selectoneu .ERROR_CODE of
;   6160  3                          set
;   6161  3
;   6162  3                          [ST_MFE]:   TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1;       ! HARD-MEDIA FORMAT
;   6163  3
;   6164  3                          [ST_DAT]:   if .ERROR_SUB eql 2                             ! HARD-DATA
;   6165  3                                      then
;   6166  3                                          TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
;   6167  3                                      else
;   6168  3                                          TALLY_ADDR [ERR_HRD_DAT] = .TALLY_ADDR [ERR_HRD_DAT] + 1;
;   6169  3
;   6170  3                          [ST_HST]:   TALLY_ADDR [ERR_HRD_HST] = .TALLY_ADDR [ERR_HRD_HST] + 1;       ! HARD-HOST ACCESS
;   6171  3
;   6172  3                          [ST_CNT]:   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;   6173  3                                                                                      ! HARD-CONTROLLER
;   6174  3
;   6175  3                          [ST_DRV]:   if .ERROR_SUB eql 3                             ! HARD-DRIVE
;   6176  3                                      then
;   6177  3                                          TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
```

```
:   6178  3                                   else
:   6179  3                                       TALLY_ADDR [ERR_HRD_DRV] = .TALLY_ADDR [ERR_HRD_DRV] + 1;
:   6180  3                           tes;
:   6181  3
:   6182  3               end
:   6183  2           else                                                    ! UNIT NOT FOUND
:   6184  2
:   6185  2               if .SOFT_OCCURED
:   6186  2               then
:   6187  2                   C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + 1
:   6188  2               else
:   6189  2                   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
:   6190  2
:   6191  1           end;                                                    ! RTNE SOFT_ERROR


                                    .SBTTL  SOFT.ERROR RDRX INTERRUPT SERVICE ROUTINES
031556                              .PSECT  $CODE$,  RO

000000  004137  000000G       SOFT.ERROR::
                                    JSR     R1,$SAVE5       ;                                   6072
000004  005746                      TST     -(SP)
000006  105046                      CLRB    -(SP)           ; FOUND                             6082
000010  105046                      CLRB    -(SP)           ; SOFT.OCCURED
000012  016646  000024              MOV     24(SP),-(SP)    ; INDEX,*                           6094
000016  012746  000102              MOV     #102,-(SP)
000022  004737  000000G             JSR     PC,BL$MUL
000026  062700  000000G             ADD     #ELOG.PKT,RO
000032  010001                      MOV     RO,R1           ; *,ELOG.ADDR
000034  116100  000020              MOVB    20(R1),RO       ; *(ELOG.ADDR),*                    6095
000040  042700  177740              BIC     #177740,RO
000044  105003                      CLRB    R3              ; ERROR.CODE
000046  050003                      BIS     RO,R3           ; *,ERROR.CODE
000050  016105  000020              MOV     20(R1),R5       ; *(ELOG.ADDR),ERROR.SUB            6096
000054  006205                      ASR     R5              ; ERROR.SUB
000056  006205                      ASR     R5              ; ERROR.SUB
000060  006205                      ASR     R5              ; ERROR.SUB
000062  006205                      ASR     R5              ; ERROR.SUB
000064  006205                      ASR     R5              ; ERROR.SUB
000066  042705  174000              BIC     #174000,R5      ; *,ERROR.SUB
000072  013700  000000G             MOV     SWP.FLAGS,RO    ;                                   6098
000076  042700  077777              BIC     #77777,RO
000102  020027  100000              CMP     RO,#-100000
000106  001010                      BNE     1$
000110  126127  000016  000002      CMPB    16(R1),#2       ; *(ELOG.ADDR),*                    6099
000116  001004                      BNE     1$
000120  005004                      CLR     R4              ; RETRIES                           6101
000122  156104  000051              BISB    51(R1),R4       ; *(ELOG.ADDR),RETRIES
000126  000402                      BR      2$              ;                                   6098
000130  012704  000001      1$:     MOV     #1,R4           ; *,RETRIES                         6103
000134  005704              2$:     TST     R4              ; RETRIES                           6105
000136  001002                      BNE     3$
000140  012704  000001              MOV     #1,R4           ; *,RETRIES                         6107
```

```
000144  012702  000006                  3$:     MOV     #6,R2                   ; *,OFFSET                      6109
000150  010200                          4$:     MOV     R2,R0                   ; OFFSET,*                      6111
000152  063700  000076'                         ADD     ICST.ADDR,R0
000156  032710  040000                          BIT     #40000,(R0)
000162  001421                                  BEQ     5$
000164  016146  000012                          MOV     12(R1),-(SP)            ; *(ELOG.ADDR),*                6112
000170  111046                                  MOVB    (R0),-(SP)
000172  042716  177760                          BIC     #177760,(SP)
000176  022626                                  CMP     (SP)+,(SP)+
000200  001012                                  BNE     5$
000202  112766  000001  000006                  MOVB    #1,6(SP)                ; *,FOUND                       6115
000210  011046                                  MOV     (R0),-(SP)              ;                               6116
000212  000316                                  SWAB    (SP)
000214  042716  177760                          BIC     #177760,(SP)
000220  012666  000010                          MOV     (SP)+,10(SP)            ; *,UNIT
000224  000405                                  BR      6$                      ;                               6114
000226  062702  000024                  5$:     ADD     #24,R2                  ; *,OFFSET                      6109
000232  020227  000102                          CMP     R2,#102                 ; OFFSET,*
000236  003744                                  BLE     4$
000240  112766  000001  000004          6$:     MOVB    #1,4(SP)                ; *,SOFT.OCCURED                6123
000246  032766  000001  000006                  BIT     #1,6(SP)                ; *,FOUND                       6125
000254  001002                                  BNE     7$
000256  000137  032444'                         JMP     22$
000262  016616  000010                  7$:     MOV     10(SP),(SP)             ; UNIT,*                        6128
000266  012746  000066                          MOV     #66,-(SP)
000272  004737  000000G                         JSR     PC,BL$MUL
000276  062700  000000G                         ADD     #TALLY,R0
000302  032766  000001  000006                  BIT     #1,6(SP)                ; *,SOFT.OCCURED                6130
000310  001503                                  BEQ     14$
000312  120327  000005                          CMPB    R3,#5                   ; ERROR.CODE,*                  6135
000316  001462                                  BEQ     12$
000320  120327  000010                          CMPB    R3,#10                  ; ERROR.CODE,*                  6137
000324  001022                                  BNE     9$
000326  012702  000052                          MOV     #52,R2                  ;                               6139
000332  060002                                  ADD     R0,R2                   ; TALLY.ADDR,*
000334  020527  000002                          CMP     R5,#2                   ; ERROR.SUB,*                   6137
000340  001005                                  BNE     8$
000342  005001                                  CLR     R1                      ;                               6139
000344  151201                                  BISB    (R2),R1
000346  060401                                  ADD     R4,R1                   ; RETRIES,*
000350  110112                                  MOVB    R1,(R2)
000352  000543                                  BR      21$                     ;                               6137
000354  005001                          8$:     CLR     R1                      ;                               6141
000356  156201  000001                          BISB    1(R2),R1
000362  060401                                  ADD     R4,R1                   ; RETRIES,*
000364  110162  000001                          MOVB    R1,1(R2)
000370  000534                                  BR      21$                     ;                               6132
000372  120327  000011                  9$:     CMPB    R3,#11                  ; ERROR.CODE,*                  6143
000376  001007                                  BNE     10$
000400  005001                                  CLR     R1
000402  156001  000055                          BISB    55(R0),R1               ; *(TALLY.ADDR),*
000406  060401                                  ADD     R4,R1                   ; RETRIES,*
000410  110160  000055                          MOVB    R1,55(R0)               ; *,*(TALLY.ADDR)
```

```
000414  000522                          BR      21$                             ;                               6132
000416  120327  000012          10$:    CMPB    R3,#12                          ; ERROR.CODE,*                  6145
000422  001012                          BNE     11$
000424  013702  000104'                  MOV     ICTLR,R2
000430  006302                          ASL     R2
000432  005001                          CLR     R1
000434  156201  000001G                 BISB    C.ERR.TBL+1(R2),R1
000440  060401                          ADD     R4,R1                           ; RETRIES,*
000442  110162  000001G                 MOVB    R1,C.ERR.TBL+1(R2)
000446  000505                          BR      21$                             ;                               6132
000450  120327  000013          11$:    CMPB    R3,#13                          ; ERROR.CODE,*                  6148
000454  001102                          BNE     21$
000456  020527  000003                  CMP     R5,#3                           ; ERROR.SUB,*
000462  001007                          BNE     13$
000464  005001                  12$:    CLR     R1                              ;                               6150
000466  156001  000052                  BISB    52(R0),R1                       ; *(TALLY.ADDR),*
000472  060401                          ADD     R4,R1                           ; RETRIES,*
000474  110160  000052                  MOVB    R1,52(R0)                       ; *,*(TALLY.ADDR)
000500  000470                          BR      21$                             ;                               6148
000502  005001                  13$:    CLR     R1                              ;                               6152
000504  156001  000054                  BISB    54(R0),R1                       ; *(TALLY.ADDR),*
000510  060401                          ADD     R4,R1                           ; RETRIES,*
000512  110160  000054                  MOVB    R1,54(R0)                       ; *,*(TALLY.ADDR)
000516  000461                          BR      21$                             ;                               6132
000520  005761  000006          14$:    TST     6(R1)                           ; *(ELOG.ADDR)                  6156
000524  001056                          BNE     21$
000526  005761  000010                  TST     10(R1)                          ; *(ELOG.ADDR)                  6157
000532  001053                          BNE     21$
000534  120327  000005                  CMPB    R3,#5                           ; ERROR.CODE,*                  6162
000540  001443                          BEQ     19$
000542  120327  000010                  CMPB    R3,#10                          ; ERROR.CODE,*                  6164
000546  001013                          BNE     16$
000550  012704  000046                  MOV     #46,R4                          ;                               6166
000554  060004                          ADD     R0,R4                           ; TALLY.ADDR,*
000556  020527  000002                  CMP     R5,#2                           ; ERROR.SUB,*                   6164
000562  001002                          BNE     15$
000564  105214                          INCB    (R4)                            ;                               6166
000566  000435                          BR      21$                             ;                               6164
000570  105264  000001          15$:    INCB    1(R4)                           ;                               6168
000574  000432                          BR      21$                             ;                               6159
000576  120327  000011          16$:    CMPB    R3,#11                          ; ERROR.CODE,*                  6170
000602  001003                          BNE     17$
000604  105260  000051                  INCB    51(R0)                          ; *(TALLY.ADDR)
000610  000424                          BR      21$                             ;                               6159
000612  120327  000012          17$:    CMPB    R3,#12                          ; ERROR.CODE,*                  6172
000616  001006                          BNE     18$
000620  013702  000104'                  MOV     ICTLR,R2
000624  006302                          ASL     R2
000626  105262  000000G                 INCB    C.ERR.TBL(R2)
000632  000413                          BR      21$                             ;                               6159
000634  120327  000013          18$:    CMPB    R3,#13                          ; ERROR.CODE,*                  6175
000640  001010                          BNE     21$
000642  020527  000003                  CMP     R5,#3                           ; ERROR.SUB,*
```

H3


```
000646  001003                          BNE     20$
000650  105260  000046         19$:     INCB    46(R0)          ; *(TALLY.ADDR)                 6177
000654  000402                          BR      21$             ;                               6175
000656  105260  000050         20$:     INCB    50(R0)          ; *(TALLY.ADDR)                 6179
000662  005726                 21$:     TST     (SP)+           ;                               6127
000664  000415                          BR      24$             ;                               6125
000666  013700  000104'         22$:     MOV     ICTLR,R0        ;                               6187
000672  006300                          ASL     R0
000674  062700  000000G                 ADD     #C.ERR.TBL,R0
000700  032766  000001  000004          BIT     #1,4(SP)        ; *,SOFT.OCCURED                6185
000706  001403                          BEQ     23$
000710  105260  000001                  INCB    1(R0)           ;                               6187
000714  000401                          BR      24$             ;                               6185
000716  105210                 23$:     INCB    (R0)            ;                               6189
000720  062706  000012         24$:     ADD     #12,SP          ;                               6072
000724  000207                          RTS     PC

; Routine Size:  235 words,     Routine Base:  $CODE$ + 31556
; Maximum stack depth per invocation:  13 words
```

```
;    6192  1     routine ERR_HRD_RTNE (ERRNUM) : novalue =
;    6193  1
;    6194  1     !+
;    6195  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    6196  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6197  1     !-
;    6198  1
;    6199  2         begin
;    6200  2
;    6201  2         local
;    6202  2             CUR_PRIORITY : word;
;    6203  2
;    6204  2         builtin
;    6205  2             PC;
;    6206  2
;    6207  2         GETPRI (CUR_PRIORITY);
;    6208  2         SETPRI (PRI04);                                 ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    6209  2
;    6210  2         if (.ERRNUM lequ 34) or                         ! FOR NON-BAD BLOCK TYPE ERRORS
;    6211  2            (.ERRNUM gtru 38) or
;    6212  2            (.ERRNUM eql 36) or
;    6213  3            (.ERRNUM eql 37)
;    6214  2         then
;    6215  2
;    6216  3             if BIT_TST (SWP_FLAGS, SWF_HRD)             ! IF ERRORS TO BE TREATED NORMALLY
;    6217  2             then
;    6218  2
;    6219  2     !ZZZ        case .ERRNUM from 31 to 45 of
;    6220  2                 case .ERRNUM from 31 to 73 of           !INCLUDE DUP NUMBERS (60-73)          ZZZ
;    6221  2             set
;    6222  2
;    6223  2                     [31]:   ERRHRD (31, EGH_30, EMS_30);    ! INVALID COMMAND
;    6224  2
;    6225  2                     [32]:   ERRHRD (32, EGH_30, EMS_30);    ! COMMAND ABORTED
;    6226  2
;    6227  2                     [33]:   ;                               !
;    6228  2
;    6229  2                     [34]:   ;                               !
;    6230  2
;    6231  2                     [35]:   ;                               ! MEDIA FORMAT ERROR
;    6232  2
;    6233  2                     [36]:   ERRHRD (36, EGH_30, EMS_30);    ! WRITE PROTECTED
;    6234  2
;    6235  2                     [37]:   ERRHRD (37, EGH_30, EMS_30);    ! COMPARE ERROR
;    6236  2
;    6237  2                     [38]:   ;                               ! DATA ERROR
;    6238  2
;    6239  2                     [39]:   ERRHRD (39, EGH_30, EMS_30);    ! HOST BUFFER ACCESS ERROR
;    6240  2
;    6241  2                     [40]:   ERRHRD (40, EGH_30, EMS_30);    ! CONTROLLER ERROR
;    6242  2
;    6243  2                     [41]:   ERRHRD (41, EGH_30, EMS_30);    ! DRIVE ERROR
;    6244  2
```

```
;   6245  2                    [42]:   ERRHRD (42, EGH_30, 0);          ! HOST WRITE COMPARE ERROR
;   6246  2
;   6247  2                    [43]:   ERRHRD (43, EGH_30, EMS_30);     ! MESSAGE FROM INTERNAL DIAGNOSTICS
;   6248  2
;   6249  2                    [44]:   ERRHRD (44, EGH_30, EMS_30);     ! DUPLICATE UNIT NUMBER
;   6250  2
;   6251  2                    [45]:   ERRHRD (45, EGH_30, EMS_30);     ! INVALID END CODE
;   6252  2
;   6253  2                    [46]:   ;                                !LEAVE ROOM FOR SOFT ERROR NUMBERS AND SOME PADDING  ZZZ
;   6254  2                    [47]:   ;                                !
;   6255  2                    [48]:   ;                                !
;   6256  2                    [49]:   ;                                !
;   6257  2                    [50]:   ;                                !
;   6258  2                    [51]:   ;                                !
;   6259  2                    [52]:   ;                                !
;   6260  2                    [53]:   ;                                !
;   6261  2                    [54]:   ;                                !
;   6262  2                    [55]:   ;                                !
;   6263  2                    [56]:   ;                                !
;   6264  2                    [57]:   ;                                !
;   6265  2                    [58]:   ;                                !
;   6266  2                    [59]:   ;                                !
;   6267  2
;   6268  2                    [60]:  ERRHRD (60, EH_12, EMS_30);       !NOT USED                                          ZZZ
;   6269  2                    [61]:  ERRHRD (61, EH_13, EMS_30);       !SUCCESSFUL MESSAGE                                ZZZ
;   6270  2                    [62]:  ERRHRD (62, EH_13, EMS_30);       !ILLEGAL UNIT NUMBER                               ZZZ
;   6271  2                    [63]:  ERRHRD (63, EH_13, EMS_30);       !ILLEGAL RELATIVE OR PHYSICAL BLOCK                ZZZ
;   6272  2                    [64]:  ERRHRD (64, EH_13, EMS_30);       !DEVICE ERROR                                      ZZZ
;   6273  2                    [65]:  ERRHRD (65, EH_13, EMS_30);       !ZERO LENGTH MESSAGE                               ZZZ
;   6274  2                    [66]:  ERRHRD (66, EH_8,  EMS_30);       !DUP UNKNOWN STATUS CODE                           ZZZ
;   6275  2                    [67]:  ERRHRD (67, EH_7,  EMS_30);       !INVALID COMMAND                                   ZZZ
;   6276  2                    [68]:  ERRHRD (68, EH_7,  EMS_30);       !NO REGION AVAILABLE                               ZZZ
;   6277  2                    [69]:  ERRHRD (69, EH_7,  EMS_30);       !NO REGION SUITABLE                                ZZZ
;   6278  2                    [70]:  ERRHRD (70, EH_7,  EMS_30);       !PROGRAM NOT KNOWN                                 ZZZ
;   6279  2                    [71]:  ERRHRD (71, EH_7,  EMS_30);       !LOAD FAILURE                                      ZZZ
;   6280  2                    [72]:  ERRHRD (72, EH_7,  EMS_30);       !STANDALONE                                        ZZZ
;   6281  2                    [73]:  ERRHRD (73, EH_8,  EMS_30);       !DUP UNKNOWN STATUS CODE                           ZZZ
;   6282  2
;   6283  2                tes
;   6284  2            else
;   6285  3                begin
;   6286  3                !****increment error count                   ! INCREMENT TOTAL ERROR COUNT
;   6287  3                PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR MESSAGE JUST LIKE DRS
;   6288  3
;   6289  3                if .ERRNUM neq 42
;   6290  3                then
;   6291  4                    begin
;   6292  4                    PRINTB (HRD_SUB);                        ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;   6293  4                    EMS_ERR ();                              ! PRINT REST OF THE INFORMATION
;   6294  3                    end;
;   6295  2                end;
;   6296  2
;   6297  2            if (.ERRNUM eql 35) or                           ! FOR BAD-BLOCK TYPE ERRORS
```

```
;   6298  3          (.ERRNUM eql 38)
;   6299  2      then
;   6300  2
;   6301  3          if BIT_TST (SWP_FLAGS, SWF_BLK)                ! IF ERRORS TO BE TREATED NORMALLY
;   6302  2          then
;   6303  2
;   6304  2              selectoneu .ERRNUM of
;   6305  2                  set
;   6306  2
;   6307  2                  [35]:   ERRHRD (35, EGH_30, EMS_30);   ! MEDIA FORMAT ERROR
;   6308  2
;   6309  2                  [38]:   ERRHRD (38, EGH_30, EMS_30);   ! DATA ERROR
;   6310  2                  tes
;   6311  2          else
;   6312  3              begin
;   6313  3              !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;   6314  3              PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);    ! PRINT ERROR LINE JUST LIKE DRS
;   6315  3              PRINTB (HRD_SUB);                          ! PRINT NEXT LINE TOO
;   6316  3              EMS_ERR ();                                ! PRINT REST OF THE INFORMATION
;   6317  2              end;
;   6318  2
;   6319  2      SETPRI (.CUR_PRIORITY);                            ! PRIORITY BACK TO NORMAL
;   6320  2
;   6321  1      end;


                                        .SBTTL  ERR.HRD.RTNE RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          ERR.HRD.RTNE:
                                        JSR     R1,$SAVE2                       ;                       6192
000004  104440                          TRAP    40                              ;                       6207
000006  010002                          MOV     R0,R2                           ; *.CUR.PRIORITY
000010  012700  000200                  MOV     #200,R0                         ;                       6208
000014  104441                          TRAP    41
000016  016601  000010                  MOV     10(SP),R1                       ; ERRNUM,*              6210
000022  020127  000042                  CMP     R1,#42
000026  101411                          BLOS    1$
000030  020127  000046                  CMP     R1,#46                          ;                       6211
000034  101006                          BHI     1$
000036  020127  000044                  CMP     R1,#44                          ;                       6212
000042  001403                          BEQ     1$
000044  020127  000045                  CMP     R1,#45                          ;                       6213
000050  001176                          BNE     27$
000052  032737  010000  000000G  1$:     BIT     #10000,SWP.FLAGS                ;                       6216
000060  001002                          BNE     2$
000062  000137  033200'                  JMP     31$
000066  010100                  2$:     MOV     R1,R0                           ;                       6220
000070  162700  000037                  SUB     #37,R0
000074  006300                          ASL     R0
000076  066007  000024'                  ADD     P.AAC(R0),PC                    ; Case dispatch
000102  104456                  4$:     TRAP    56                              ;                       6223
000104  000037                          .WORD   37
000106  000000G                          .WORD   EGH.30
000110  000000G                          .WORD   EMS.30
```

```
000112  000567                  BR      30$                             ;       6220
000114  104456          5$:     TRAP    56                              ;       6225
000116  000040                  .WORD   40
000120  000000G                 .WORD   EGH.30
000122  000000G                 .WORD   EMS.30
000124  000562                  BR      30$                             ;       6220
000126  104456          6$:     TRAP    56                              ;       6233
000130  000044                  .WORD   44
000132  000000G                 .WORD   EGH.30
000134  000000G                 .WORD   EMS.30
000136  000555                  BR      30$                             ;       6220
000140  104456          7$:     TRAP    56                              ;       6235
000142  000045                  .WORD   45
000144  000000G                 .WORD   EGH.30
000146  000000G                 .WORD   EMS.30
000150  000550                  BR      30$                             ;       6220
000152  104456          8$:     TRAP    56                              ;       6239
000154  000047                  .WORD   47
000156  000000G                 .WORD   EGH.30
000160  000000G                 .WORD   EMS.30
000162  000574                  BR      33$                             ;       6220
000164  104456          9$:     TRAP    56                              ;       6241
000166  000050                  .WORD   50
000170  000000G                 .WORD   EGH.30
000172  000000G                 .WORD   EMS.30
000174  000567                  BR      33$                             ;       6220
000176  104456          10$:    TRAP    56                              ;       6243
000200  000051                  .WORD   51
000202  000000G                 .WORD   EGH.30
000204  000000G                 .WORD   EMS.30
000206  000562                  BR      33$                             ;       6220
000210  104456          11$:    TRAP    56                              ;       6245
000212  000052                  .WORD   52
000214  000000G                 .WORD   EGH.30
000216  000000                  .WORD   0
000220  000555                  BR      33$                             ;       6220
000222  104456          12$:    TRAP    56                              ;       6247
000224  000053                  .WORD   53
000226  000000G                 .WORD   EGH.30
000230  000000G                 .WORD   EMS.30
000232  000550                  BR      33$                             ;       6220
000234  104456          13$:    TRAP    56                              ;       6249
000236  000054                  .WORD   54
000240  000000G                 .WORD   EGH.30
000242  000000G                 .WORD   EMS.30
000244  000543                  BR      33$                             ;       6220
000246  104456          14$:    TRAP    56                              ;       6251
000250  000055                  .WORD   55
000252  000000G                 .WORD   EGH.30
000254  000000G                 .WORD   EMS.30
000256  000536                  BR      33$                             ;       6220
000260  104456          15$:    TRAP    56                              ;       6268
000262  000074                  .WORD   74
```

```
000264  000000G                         .WORD   EH.12
000266  000000G                         .WORD   EMS.30
000270  000531                          BR      33$                             ;       6220
000272  104456                  16$:    TRAP    56                              ;       6269
000274  000075                          .WORD   75
000276  000000G                         .WORD   EH.13
000300  000000G                         .WORD   EMS.30
000302  000524                          BR      33$                             ;       6220
000304  104456                  17$:    TRAP    56                              ;       6270
000306  000076                          .WORD   76
000310  000000G                         .WORD   EH.13
000312  000000G                         .WORD   EMS.30
000314  000517                          BR      33$                             ;       6220
000316  104456                  18$:    TRAP    56                              ;       6271
000320  000077                          .WORD   77
000322  000000G                         .WORD   EH.13
000324  000000G                         .WORD   EMS.30
000326  000512                          BR      33$                             ;       6220
000330  104456                  19$:    TRAP    56                              ;       6272
000332  000100                          .WORD   100
000334  000000G                         .WORD   EH.13
000336  000000G                         .WORD   EMS.30
000340  000505                          BR      33$                             ;       6220
000342  104456                  20$:    TRAP    56                              ;       6273
000344  000101                          .WORD   101
000346  000000G                         .WORD   EH.13
000350  000000G                         .WORD   EMS.30
000352  000500                          BR      33$                             ;       6220
000354  104456                  21$:    TRAP    56                              ;       6274
000356  000102                          .WORD   102
000360  000000G                         .WORD   EH.8
000362  000000G                         .WORD   EMS.30
000364  000473                          BR      33$                             ;       6220
000366  104456                  22$:    TRAP    56                              ;       6275
000370  000103                          .WORD   103
000372  000000G                         .WORD   EH.7
000374  000000G                         .WORD   EMS.30
000376  000466                          BR      33$                             ;       6220
000400  104456                  23$:    TRAP    56                              ;       6276
000402  000104                          .WORD   104
000404  000000G                         .WORD   EH.7
000406  000000G                         .WORD   EMS.30
000410  000461                          BR      33$                             ;       6220
000412  104456                  24$:    TRAP    56                              ;       6277
000414  000105                          .WORD   105
000416  000000G                         .WORD   EH.7
000420  000000G                         .WORD   EMS.30
000422  000454                          BR      33$                             ;       6220
000424  104456                  25$:    TRAP    56                              ;       6278
000426  000106                          .WORD   106
000430  000000G                         .WORD   EH.7
000432  000000G                         .WORD   EMS.30
000434  000447                          BR      33$                             ;       6220
```

```
000436  104456                          26$:    TRAP    56                              ;                               6279
000440  000107                                  .WORD   107
000442  000000G                                 .WORD   EH.7
000444  000000G                                 .WORD   EMS.30
000446  000442                          27$:    BR      33$                             ;                               6220
000450  104456                          28$:    TRAP    56                              ;                               6280
000452  000110                                  .WORD   110
000454  000000G                                 .WORD   EH.7
000456  000000G                                 .WORD   EMS.30
000460  000435                                  BR      33$                             ;                               6220
000462  104456                          29$:    TRAP    56                              ;                               6281
000464  000111                                  .WORD   111
000466  000000G                                 .WORD   EH.8
000470  000000G                                 .WORD   EMS.30
000472  000430                          30$:    BR      33$                             ;                               6216
000474  010746                          31$:    MOV     PC,-(SP)                        ; PC,*                          6287
000476  013746  000000G                         MOV     L$LUN,-(SP)
000502  010146                                  MOV     R1,-(SP)
000504  012746  000000G                         MOV     #HRD.MSG,-(SP)
000510  012746  000004                         MOV     #4,-(SP)
000514  010600                                  MOV     SP,R0                           ; SP,*
000516  104414                                  TRAP    14
000520  020127  000052                         CMP     R1,#52                          ;                               6289
000524  001411                                  BEQ     32$
000526  012716  000000G                         MOV     #HRD.SUB,(SP)                   ;                               6292
000532  012746  000001                         MOV     #1,-(SP)
000536  010600                                  MOV     SP,R0                           ; SP,*
000540  104414                                  TRAP    14
000542  004737  000000G                         JSR     PC,EMS.ERR                      ;                               6293
000546  005726                                  TST     (SP)+                           ;                               6291
000550  062706  000012                  32$:    ADD     #12,SP                          ;                               6285
000554  020127  000043                  33$:    CMP     R1,#43                          ;                               6297
000560  001403                                  BEQ     34$
000562  020127  000046                         CMP     R1,#46                          ;                               6298
000566  001050                                  BNE     37$
000570  032737  040000  000000G         34$:    BIT     #40000,SWP.FLAGS                ;                               6301
000576  001420                                  BEQ     36$
000600  020127  000043                         CMP     R1,#43                          ;                               6307
000604  001005                                  BNE     35$
000606  104456                                  TRAP    56
000610  000043                                  .WORD   43
000612  000000G                                 .WORD   EGH.30
000614  000000G                                 .WORD   EMS.30
000616  000434                                  BR      37$                             ;                               6304
000620  020127  000046                  35$:    CMP     R1,#46                          ;                               6309
000624  001031                                  BNE     37$
000626  104456                                  TRAP    56
000630  000046                                  .WORD   46
000632  000000G                                 .WORD   EGH.30
000634  000000G                                 .WORD   EMS.30
000636  000424                                  BR      37$                             ;                               6301
000640  010746                          36$:    MOV     PC,-(SP)                        ; PC,*                          6314
000642  013746  000000G                         MOV     L$LUN,-(SP)
```

```
000646  010146                          MOV     R1,-(SP)
000650  012746  000000G                 MOV     #HRD.MSG,-(SP)
000654  012746  000004                  MOV     #4,-(SP)
000660  010600                          MOV     SP,R0                   ; SP,*
000662  104414                          TRAP    14
000664  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                               6315
000670  012746  000001                  MOV     #1,-(SP)
000674  010600                          MOV     SP,R0                   ; SP,*
000676  104414                          TRAP    14
000700  004737  000000G                 JSR     PC,EMS.ERR              ;                               6316
000704  062706  000014                  ADD     #14,SP                  ;                               6312
000710  010200                  37$:    MOV     R2,R0                   ; CUR.PRIORITY,*                6319
000712  104441                          TRAP    41
000714  000207                          RTS     PC                      ;                               6192

; Routine Size:  231 words,     Routine Base:  $CODE$ + 32504
; Maximum stack depth per invocation:  11 words


000024                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAC:
000024  000000                  3$:     .WORD   0                       ; CASE Table for ERR.HRD.RTNE+0076      6220
000026  000012                          .WORD   12                      ; [4$]
000030  000452                          .WORD   452                     ; [5$]
000032  000452                          .WORD   452                     ; [33$]
000034  000452                          .WORD   452                     ; [33$]
000036  000024                          .WORD   24                      ; [33$]
000040  000036                          .WORD   36                      ; [6$]
000042  000452                          .WORD   452                     ; [7$]
000044  000050                          .WORD   50                      ; [33$]
000046  000062                          .WORD   62                      ; [8$]
000050  000074                          .WORD   74                      ; [9$]
000052  000106                          .WORD   106                     ; [10$]
000054  000120                          .WORD   120                     ; [11$]
000056  000132                          .WORD   132                     ; [12$]
000060  000144                          .WORD   144                     ; [13$]
000062  000452                          .WORD   452                     ; [14$]
000064  000452                          .WORD   452                     ; [33$]
000066  000452                          .WORD   452                     ; [33$]
000070  000452                          .WORD   452                     ; [33$]
000072  000452                          .WORD   452                     ; [33$]
000074  000452                          .WORD   452                     ; [33$]
000076  000452                          .WORD   452                     ; [33$]
000100  000452                          .WORD   452                     ; [33$]
000102  000452                          .WORD   452                     ; [33$]
000104  000452                          .WORD   452                     ; [33$]
000106  000452                          .WORD   452                     ; [33$]
000110  000452                          .WORD   452                     ; [33$]
000112  000452                          .WORD   452                     ; [33$]
000114  000452                          .WORD   452                     ; [33$]
000116  000156                          .WORD   156                     ; [15$]
000120  000170                          .WORD   170                     ; [16$]
```

```
000122  000202                          .WORD   202             ; [17$]
000124  000214                          .WORD   214             ; [18$]
000126  000226                          .WORD   226             ; [19$]
000130  000240                          .WORD   240             ; [20$]
000132  000252                          .WORD   252             ; [21$]
000134  000264                          .WORD   264             ; [22$]
000136  000276                          .WORD   276             ; [23$]
000140  000310                          .WORD   310             ; [24$]
000142  000322                          .WORD   322             ; [25$]
000144  000334                          .WORD   334             ; [26$]
000146  000346                          .WORD   346             ; [28$]
000150  000360                          .WORD   360             ; [29$]
```

```
;    6322  1     routine ERR_SOFT_RTNE (ERRNUM) : novalue =
;    6323  1
;    6324  1     !+
;    6325  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6326  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6327  1     !-
;    6328  1
;    6329  2         begin
;    6330  2
;    6331  2         builtin
;    6332  2             PC;
;    6333  2
;    6334  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                 ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    6335  2         then
;    6336  2
;    6337  2             case .ERRNUM from 50 to 54 of
;    6338  2                 set
;    6339  2
;    6340  2                 [50]:       ERRSOFT (50, 0, 0);         ! CONTROLLER ERROR
;    6341  2
;    6342  2                 [51]:       ERRSOFT (51, 0, 0);         ! HOST MEMORY ACCESS ERROR
;    6343  2
;    6344  2                 [52]:       ERRSOFT (52, 0, 0);         ! DISK TRANSFER ERROR
;    6345  2
;    6346  2                 [53]:       ERRSOFT (53, 0, 0);         ! SDI ERROR
;    6347  2
;    6348  2                 [54]:       ERRSOFT (54, 0, 0);         ! SMALL DISK ERROR
;    6349  2                 tes
;    6350  2         else
;    6351  3             begin
;    6352  3             !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;    6353  3             PRINTB (SFT_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR LINE JUST LIKE DRS
;    6354  2             end;
;    6355  2
;    6356  1         end;


                                        .SBTTL  ERR.SOFT.RTNE RDRX INTERRUPT SERVICE ROUTINES
033422                                  .PSECT  $CODE$,  RO

000000  032737  020000  000000G  ERR.SOFT.RTNE:
                                        BIT     #20000,SWP.FLAGS            ;                   6334
000006  001440                          BEQ     7$
000010  016600  000002                  MOV     2(SP),RO                    ; ERRNUM,*          6337
000014  162700  000062                  SUB     #62,RO
000020  006300                          ASL     RO
000022  066007  000152'                 ADD     P.AAD(RO),PC                ; Case dispatch
000026  104457                  2$:     TRAP    57                          ;                   6340
000030  000062                          .WORD   62
000032  000000                          .WORD   0
000034  000000                          .WORD   0
000036  000207                          RTS     PC                          ;                   6337
000040  104457                  3$:     TRAP    57                          ;                   6342
```

```
000042  000063                          .WORD   63
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  000207                          RTS     PC                              ;                               6337
000052  104457                  4$:     TRAP    57                              ;                               6344
000054  000064                          .WORD   64
000056  000000                          .WORD   0
000060  000000                          .WORD   0
000062  000207                          RTS     PC                              ;                               6337
000064  104457                  5$:     TRAP    57                              ;                               6346
000066  000065                          .WORD   65
000070  000000                          .WORD   0
000072  000000                          .WORD   0
000074  000207                          RTS     PC                              ;                               6337
000076  104457                  6$:     TRAP    57                              ;                               6348
000100  000066                          .WORD   66
000102  000000                          .WORD   0
000104  000000                          .WORD   0
000106  000207                          RTS     PC                              ;                               6334
000110  010746                  7$:     MOV     PC,-(SP)                        ; PC,*                          6353
000112  013746  000000G                 MOV     L$LUN,-(SP)
000116  016646  000006                  MOV     6(SP),-(SP)                     ; ERRNUM,*
000122  012746  000000G                 MOV     #SFT.MSG,-(SP)
000126  012746  000004                  MOV     #4,-(SP)
000132  010600                          MOV     SP,R0                           ; SP,*
000134  104414                          TRAP    14
000136  062706  000012                  ADD     #12,SP                          ;                               6351
000142  000207                          RTS     PC                              ;                               6322

; Routine Size:  50 words,      Routine Base:  $CODE$ + 33422
; Maximum stack depth per invocation:  7 words


000152                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAD:                                          ; CASE Table for ERR.SOFT.RTNE+0022     6337
000152  000000                 1$:     .WORD   0                               ; [2$]
000154  000012                          .WORD   12                              ; [3$]
000156  000024                          .WORD   24                              ; [4$]
000160  000036                          .WORD   36                              ; [5$]
000162  000050                          .WORD   50                              ; [6$]
```

```
;    6357  1     routine ERR_HRD_RTNE_APT (ERRNUM) : novalue =
;    6358  1
;    6359  1
;    6360  1     !+
;    6361  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    6362  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6363  1     !-
;    6364  1
;    6365  1
;    6366  2         begin
;    6367  2
;    6368  2
;    6369  2         local
;    6370  2             CUR_PRIORITY;
;    6371  2
;    6372  2
;    6373  2         builtin
;    6374  2             PC;
;    6375  2
;    6376  2         GETPRI (CUR_PRIORITY);
;    6377  2         SETPRI (PRI04);                          ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    6378  2
;    6379  2
;    6380  2         if .APT_MODE
;    6381  2         then
;    6382  2
;    6383  3             begin
;    6384  3             .MAIL_BOX_TESTNUM = .RP_ADDR [LBN_LO];       ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    6385  3             .MAIL_BOX_SUBTST = .RP_ADDR [DISK];          ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER UNDER APT ONLY
;    6386  2             end;
;    6387  2
;    6388  2
;    6389  2         if (.ERRNUM lequ 34) or                      ! FOR NON-BAD BLOCK TYPE ERRORS
;    6390  2            (.ERRNUM gtru 38) or
;    6391  2            (.ERRNUM eql 36) or
;    6392  3            (.ERRNUM eql 37)
;    6393  3
;    6394  2         then
;    6395  2
;    6396  3             if BIT_TST (SWP_FLAGS, SWF_HRD)          ! IF ERRORS TO BE TREATED NORMALLY
;    6397  2             then
;    6398  2
;    6399  2                 case .ERRNUM from 31 to 45 of
;    6400  2                     set
;    6401  2
;    6402  2                     [31]:   ERRDF (31, EGH_30, EMS_30);     ! INVALID COMMAND
;    6403  2
;    6404  2                     [32]:   ERRDF (32, EGH_30, EMS_30);     ! COMMAND ABORTED
;    6405  2
;    6406  2                     [33]:   ;                               !
;    6407  2
;    6408  2                     [34]:   ;                               !
;    6409  2
```

```
;  6410  2                    [35]:   ;                                   ! MEDIA FORMAT ERROR
;  6411  2
;  6412  2                    [36]:   ERRDF (36, EGH_30, EMS_30);         ! WRITE PROTECTED
;  6413  2
;  6414  2                    [37]:   ERRDF (37, EGH_30, EMS_30);         ! COMPARE ERROR
;  6415  2
;  6416  2                    [38]:   ;                                   ! DATA ERROR
;  6417  2
;  6418  2                    [39]:   ERRDF (39, EGH_30, EMS_30);         ! HOST BUFFER ACCESS ERROR
;  6419  2
;  6420  2                    [40]:   ERRDF (40, EGH_30, EMS_30);         ! CONTROLLER ERROR
;  6421  2
;  6422  2                    [41]:   ERRDF (41, EGH_30, EMS_30);         ! DRIVE ERROR
;  6423  2
;  6424  2                    [42]:   ERRDF (42, EGH_30, 0);              ! HOST WRITE COMPARE ERROR
;  6425  2
;  6426  2                    [43]:   ERRDF (43, EGH_30, EMS_30);         ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  6427  2
;  6428  2                    [44]:   ERRDF (44, EGH_30, EMS_30);         ! DUPLICATE UNIT NUMBER
;  6429  2
;  6430  2                    [45]:   ERRDF (45, EGH_30, EMS_30);         ! INVALID END CODE
;  6431  2                    tes
;  6432  2
;  6433  2                else
;  6434  2
;  6435  3                    begin
;  6436  3                    !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;  6437  3                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR MESSAGE JUST LIKE DRS
;  6438  3
;  6439  3
;  6440  3                    if .ERRNUM neq 42
;  6441  3
;  6442  3                    then
;  6443  4                        begin
;  6444  4                        PRINTB (HRD_SUB);                       ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;  6445  4                        EMS_ERR ();                             ! PRINT REST OF THE INFORMATION
;  6446  3                        end;
;  6447  2                    end;
;  6448  2
;  6449  2                if (.ERRNUM eql 35) or                          ! FOR BAD-BLOCK TYPE ERRORS
;  6450  3                   (.ERRNUM eql 38)
;  6451  3
;  6452  2                then
;  6453  2
;  6454  3                    if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF ERRORS TO BE TREATED NORMALLY
;  6455  2                    then
;  6456  2
;  6457  2                        selectoneu .ERRNUM of
;  6458  2                            set
;  6459  2
;  6460  2                            [35]:   ERRDF (35, EGH_30, EMS_30); ! MEDIA FORMAT ERROR
;  6461  2
;  6462  2                            [38]:   ERRDF (38, EGH_30, EMS_30); ! DATA ERROR
```

```
;   6463  2                     tes
;   6464  2
;   6465  2             else
;   6466  2
;   6467  3                 begin
;   6468  3                 !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;   6469  3                 PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR LINE JUST LIKE DRS
;   6470  3                 PRINTB (HRD_SUB);                           ! PRINT NEXT LINE TOO
;   6471  3                 EMS_ERR ();                                 ! PRINT REST OF THE INFORMATION
;   6472  2                 end;
;   6473  2
;   6474  2
;   6475  2         SETPRI (.CUR_PRIORITY);                             ! PRIORITY BACK TO NORMAL
;   6476  2
;   6477  2
;   6478  1         end;


                                        .SBTTL  ERR.HRD.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
033566                                  .PSECT  $CODE$,  RO

000000  004137  000000G          ERR.HRD.RTNE.APT:
                                        JSR     R1,$SAVE2                   ;                       6357
000004  104440                          TRAP    40                          ;                       6376
000006  010002                          MOV     R0,R2                       ; *,CUR.PRIORITY
000010  012700  000200                  MOV     #200,R0                     ;                       6377
000014  104441                          TRAP    41
000016  032737  000001  001166'         BIT     #1,APT.MODE                 ;                       6380
000024  001412                          BEQ     1$
000026  013700  000000G                 MOV     RP.ADDR,R0                  ;                       6384
000032  016077  000050  001170'         MOV     50(R0),@MAIL.BOX.TESTNUM
000040  013700  000000G                 MOV     RP.ADDR,R0                  ;                       6385
000044  016077  000010  001172'         MOV     10(R0),@MAIL.BOX.SUBTST
000052  016601  000010          1$:     MOV     10(SP),R1                   ; ERRNUM,*              6389
000056  020127  000042                  CMP     R1,#42
000062  101411                          BLOS    2$
000064  020127  000046                  CMP     R1,#46                      ;                       6390
000070  101006                          BHI     2$
000072  020127  000044                  CMP     R1,#44                      ;                       6391
000076  001403                          BEQ     2$
000100  020127  000045                  CMP     R1,#45                      ;                       6392
000104  001131                          BNE     17$
000106  032737  010000  000000G 2$:     BIT     #10000,SWP.FLAGS            ;                       6396
000114  001475                          BEQ     15$
000116  010100                          MOV     R1,R0                       ;                       6399
000120  162700  000037                  SUB     #37,R0
000124  006300                          ASL     R0
000126  066007  000164'                 ADD     P.AAE(R0),PC                ; Case dispatch
000132  104455                  4$:     TRAP    55                          ;                       6402
000134  000037                          .WORD   37
000136  000000G                         .WORD   EGH.30
000140  000000G                         .WORD   EMS.30
000142  000512                          BR      17$                         ;                       6399
```

```
000144  104455                  5$:     TRAP    55                              ;                       6404
000146  000040                          .WORD   40
000150  000000G                         .WORD   EGH.30
000152  000000G                         .WORD   EMS.30
000154  000505                          BR      17$                             ;                       6399
000156  104455                  6$:     TRAP    55                              ;                       6412
000160  000044                          .WORD   44
000162  000000G                         .WORD   EGH.30
000164  000000G                         .WORD   EMS.30
000166  000500                          BR      17$                             ;                       6399
000170  104455                  7$:     TRAP    55                              ;                       6414
000172  000045                          .WORD   45
000174  000000G                         .WORD   EGH.30
000176  000000G                         .WORD   EMS.30
000200  000473                          BR      17$                             ;                       6399
000202  104455                  8$:     TRAP    55                              ;                       6418
000204  000047                          .WORD   47
000206  000000G                         .WORD   EGH.30
000210  000000G                         .WORD   EMS.30
000212  000466                          BR      17$                             ;                       6399
000214  104455                  9$:     TRAP    55                              ;                       6420
000216  000050                          .WORD   50
000220  000000G                         .WORD   EGH.30
000222  000000G                         .WORD   EMS.30
000224  000461                          BR      17$                             ;                       6399
000226  104455                  10$:    TRAP    55                              ;                       6422
000230  000051                          .WORD   51
000232  000000G                         .WORD   EGH.30
000234  000000G                         .WORD   EMS.30
000236  000454                          BR      17$                             ;                       6399
000240  104455                  11$:    TRAP    55                              ;                       6424
000242  000052                          .WORD   52
000244  000000G                         .WORD   EGH.30
000246  000000                          .WORD   0
000250  000447                          BR      17$                             ;                       6399
000252  104455                  12$:    TRAP    55                              ;                       6426
000254  000053                          .WORD   53
000256  000000G                         .WORD   EGH.30
000260  000000G                         .WORD   EMS.30
000262  000442                          BR      17$                             ;                       6399
000264  104455                  13$:    TRAP    55                              ;                       6428
000266  000054                          .WORD   54
000270  000000G                         .WORD   EGH.30
000272  000000G                         .WORD   EMS.30
000274  000435                          BR      17$                             ;                       6399
000276  104455                  14$:    TRAP    55                              ;                       6430
000300  000055                          .WORD   55
000302  000000G                         .WORD   EGH.30
000304  000000G                         .WORD   EMS.30
000306  000430                          BR      17$                             ;                       6396
000310  010746                  15$:    MOV     PC,-(SP)                        ; PC,*                  6437
000312  013746  000000G                 MOV     L$LUN,-(SP)
000316  010146                          MOV     R1,-(SP)
```

```
000320  012746  000000G                        MOV     #DF.MSG,-(SP)
000324  012746  000004                         MOV     #4,-(SP)
000330  010600                                 MOV     SP,R0                   ; SP,*
000332  104414                                 TRAP    14
000334  020127  000052                         CMP     R1,#52                  ;                       6440
000340  001411                                 BEQ     16$
000342  012716  000000G                        MOV     #HRD.SUB,(SP)           ;                       6444
000346  012746  000001                         MOV     #1,-(SP)
000352  010600                                 MOV     SP,R0                   ; SP,*
000354  104414                                 TRAP    14
000356  004737  000000G                        JSR     PC,EMS.ERR              ;                       6445
000362  005726                                 TST     (SP)+                   ;                       6443
000364  062706  000012              16$:       ADD     #12,SP                  ;                       6435
000370  020127  000043              17$:       CMP     R1,#43                  ;                       6449
000374  001403                                 BEQ     18$
000376  020127  000046                         CMP     R1,#46                  ;                       6450
000402  001050                                 BNE     21$
000404  032737  040000  000000G     18$:       BIT     #40000,SWP.FLAGS        ;                       6454
000412  001420                                 BEQ     20$
000414  020127  000043                         CMP     R1,#43                  ;                       6460
000420  001005                                 BNE     19$
000422  104455                                 TRAP    55
000424  000043                                 .WORD   43
000426  000000G                                .WORD   EGH.30
000430  000000G                                .WORD   EMS.30
000432  000434                                 BR      21$                     ;                       6457
000434  020127  000046              19$:       CMP     R1,#46                  ;                       6462
000440  001031                                 BNE     21$
000442  104455                                 TRAP    55
000444  000046                                 .WORD   46
000446  000000G                                .WORD   EGH.30
000450  000000G                                .WORD   EMS.30
000452  000424                                 BR      21$                     ;                       6454
000454  010746              20$:               MOV     PC,-(SP)                ; PC,*                  6469
000456  013746  000000G                        MOV     L$LUN,-(SP)
000462  010146                                 MOV     R1,-(SP)
000464  012746  000000G                        MOV     #DF.MSG,-(SP)
000470  012746  000004                         MOV     #4,-(SP)
000474  010600                                 MOV     SP,R0                   ; SP,*
000476  104414                                 TRAP    14
000500  012716  000000G                        MOV     #HRD.SUB,(SP)           ;                       6470
000504  012746  000001                         MOV     #1,-(SP)
000510  010600                                 MOV     SP,R0                   ; SP,*
000512  104414                                 TRAP    14
000514  004737  000000G                        JSR     PC,EMS.ERR              ;                       6471
000520  062706  000014                         ADD     #14,SP                  ;                       6467
000524  010200              21$:               MOV     R2,R0                   ; CUR.PRIORITY,*        6475
000526  104441                                 TRAP    41
000530  000207                                 RTS     PC                      ;                       6357

; Routine Size:  173 words,    Routine Base:  $CODE$ + 33566
; Maximum stack depth per invocation:  11 words
```

```
000164                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAE:                                  ; CASE Table for ERR.HRD.RTNE.AP+0126   6399
000164  000000                3$:     .WORD   0                       ; [4$]
000166  000012                        .WORD   12                      ; [5$]
000170  000236                        .WORD   236                     ; [17$]
000172  000236                        .WORD   236                     ; [17$]
000174  000236                        .WORD   236                     ; [17$]
000176  000024                        .WORD   24                      ; [6$]
000200  000036                        .WORD   36                      ; [7$]
000202  000236                        .WORD   236                     ; [17$]
000204  000050                        .WORD   50                      ; [8$]
000206  000062                        .WORD   62                      ; [9$]
000210  000074                        .WORD   74                      ; [10$]
000212  000106                        .WORD   106                     ; [11$]
000214  000120                        .WORD   120                     ; [12$]
000216  000132                        .WORD   132                     ; [13$]
000220  000144                        .WORD   144                     ; [14$]


;   6479  1
;   6480  1
```

```
;    6481  1     routine ERR_SOFT_RTNE_APT (ERRNUM, index) : novalue =
;    6482  1
;    6483  1     !+
;    6484  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6485  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6486  1     !-
;    6487  1
;    6488  2         begin
;    6489  2
;    6490  2         local
;    6491  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;    6492  2
;    6493  2         builtin
;    6494  2             PC;
;    6495  2
;    6496  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);       ! ADDRESS OF THE SAVED ERROR-LOG INFORMATION
;    6497  2
;    6498  2         if .APT_MODE
;    6499  2         then
;    6500  3             begin
;    6501  3             .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];      ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    6502  3             .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
;    6503  2             end;
;    6504  2
;    6505  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                     ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    6506  2         then
;    6507  2
;    6508  2             case .ERRNUM from 50 to 54 of
;    6509  2                 set
;    6510  2
;    6511  2                 [50]:       ERRDF (50, 0, 0);               ! CONTROLLER ERROR
;    6512  2
;    6513  2                 [51]:       ERRDF (51, 0, 0);               ! HOST MEMORY ACCESS ERROR
;    6514  2
;    6515  2                 [52]:       ERRDF (52, 0, 0);               ! DISK TRANSFER ERROR
;    6516  2
;    6517  2                 [53]:       ERRDF (53, 0, 0);               ! SDI ERROR
;    6518  2
;    6519  2                 [54]:       ERRDF (54, 0, 0);               ! SMALL DISK ERROR
;    6520  2                 tes
;    6521  2         else
;    6522  3             begin
;    6523  3             !****increment error count                      ! INCREMENT TOTAL ERROR COUNT
;    6524  3             PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);         ! PRINT ERROR LINE JUST LIKE DRS
;    6525  2             end;
;    6526  2
;    6527  1         end;


                                        .SBTTL  ERR.SOFT.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
034320                                  .PSECT  $CODE$,  RO

000000  016646  000002                 ERR.SOFT.RTNE.APT:
```

```
                                        MOV     2(SP),-(SP)             ; INDEX,*                       6496
000004  012746  000102                  MOV     #102,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  062700  000000G                 ADD     #ELOG.PKT,R0
000020  032737  000001  001166'         BIT     #1,APT.MODE             ;                               6498
000026  001406                          BEQ     1$
000030  016077  000056  001170'         MOV     56(R0),@MAIL.BOX.TESTNUM ; *(ELOG.ADDR),*               6501
000036  016077  000012  001172'         MOV     12(R0),@MAIL.BOX.SUBTST ; *(ELOG.ADDR),*                6502
000044  032737  020000  000000G 1$:     BIT     #20000,SWP.FLAGS        ;                               6505
000052  001440                          BEQ     8$
000054  016600  000010                  MOV     10(SP),R0               ; ERRNUM,*                      6508
000060  162700  000062                  SUB     #62,R0
000064  006300                          ASL     R0
000066  066007  000222'                 ADD     P.AAF(R0),PC            ; Case dispatch
000072  104455              3$:         TRAP    55                      ;                               6511
000074  000062                          .WORD   62
000076  000000                          .WORD   0
000100  000000                          .WORD   0
000102  000441                          BR      9$                      ;                               6508
000104  104455              4$:         TRAP    55                      ;                               6513
000106  000063                          .WORD   63
000110  000000                          .WORD   0
000112  000000                          .WORD   0
000114  000434                          BR      9$                      ;                               6508
000116  104455              5$:         TRAP    55                      ;                               6515
000120  000064                          .WORD   64
000122  000000                          .WORD   0
000124  000000                          .WORD   0
000126  000427                          BR      9$                      ;                               6508
000130  104455              6$:         TRAP    55                      ;                               6517
000132  000065                          .WORD   65
000134  000000                          .WORD   0
000136  000000                          .WORD   0
000140  000422                          BR      9$                      ;                               6508
000142  104455              7$:         TRAP    55                      ;                               6519
000144  000066                          .WORD   66
000146  000000                          .WORD   0
000150  000000                          .WORD   0
000152  000415                          BR      9$                      ;                               6505
000154  010716              8$:         MOV     PC,(SP)                 ; PC,*                          6524
000156  013746  000000G                 MOV     L$LUN,-(SP)
000162  016646  000012                  MOV     12(SP),-(SP)            ; ERRNUM,*
000166  012746  000000G                 MOV     #DF.MSG,-(SP)
000172  012746  000004                  MOV     #4,-(SP)
000176  010600                          MOV     SP,R0                   ; SP,*
000200  104414                          TRAP    14
000202  062706  000010                  ADD     #10,SP                  ;                               6522
000206  022626              9$:         CMP     (SP)+,(SP)+             ;                               6488
000210  000207                          RTS     PC                      ;                               6481

; Routine Size:  69 words,      Routine Base:  $CODE$ + 34320
; Maximum stack depth per invocation:  8 words
```

```
000222                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAF:                                  ; CASE Table for ERR.SOFT.RTNE.A+0066   6508
000222  000000                  2$:     .WORD   0                       ; [3$]
000224  000012                          .WORD   12                      ; [4$]
000226  000024                          .WORD   24                      ; [5$]
000230  000036                          .WORD   36                      ; [6$]
000232  000050                          .WORD   50                      ; [7$]


;   6528  1
;   6529  1
;   6530  1     end
;   6531  1
;   6532  0     eludom


;                               OTS external references
                                        .GLOBL  $SAVE5, $SAVE4, $SAVE3, $SAVE2
                                        .GLOBL  BL$SHF, BL$DIV, BL$MOD, BL$MUL


;                               PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;         $GGG$                           326       RO ,  I  ,  LCL,  REL,  CON
;         $CODE$                         7341       RO ,  I  ,  LCL,  REL,  CON
;         $PLIT$                           78       RO ,  D  ,  LCL,  REL,  CON


;                           Library Statistics
;
;                                    -------- Symbols --------      Pages       Processing
;       File                         Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAE0.L16;18    405       332        81          21        00:00.1


;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAE0.BL2/LIST=ZRQAE0.LS2/OBJECT=ZRQAE0.OB2/SOURCE=PAGE:53
```

```
;    6533  0     module ZRQAM4 (
;    6534  0
;    6535  0     %title 'RD/RX EXERCISER'
;    6536  0                 ident = 'V01.8',
;    6537  0                 addressing_mode (absolute),
;    6538  0                 environment (noeis)
;    6539  0                 ) =
;    6540  0
;    6541  1     begin
;    6542  1
;    6543  1     %sbtti 'LASTAD AND SETUP'
;    6544  1
;    6545  1     library 'ZRQAE0.L16';
;    6546  1
;    6547  1     require 'BLSMAC.REQ';                          ! DIAGNOSTIC SUPERVISOR LIBRARY
;    8038  1
;    8039  2     LASTAD
;    8040  2
;    8041  2     BGNSETUP (4)                                                        !ZZZ
;    8042  2
; P  8043  2             BGNPTAB
; P  8044  2                 INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000060', 0, 0, RD52_MAX_LBN, 0  !ZZZ
; P  8045  2                     ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
; P  8046  2
;    8047  2             ENDPTAB
;    8048  2
; P  8049  2             BGNPTAB
; P  8050  2                 INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000001', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  8051  2                     ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    8052  2             ENDPTAB
;    8053  2
; P  8054  2             BGNPTAB
; P  8055  2                 INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000002', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  8056  2                     ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    8057  2             ENDPTAB
; P  8058  2             BGNPTAB
; P  8059  2                 INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000003', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  8060  2                     !HERE'S ONE FOR THE 4TH DRIVE                                                !ZZZ
;    8061  2             ENDPTAB                                                                              !ZZZ
;    8062  2
;    8063  1     ENDSETUP


                                        .TITLE  ZRQAM4 RD/RX EXERCISER
                                        .IDENT  /V01.8/
                                        .ENABL  AMA


000000                                  .PSECT  $XYZ$,  RO
000000  000124'                  BL$LAS::.WORD   T$FREE
000002  000000C                          .WORD   <<T$FREE-<BL$LAS+4>>/2>
000004  000034'                  P.AAA:  .WORD   L$LAST+30
000006  000010                          .WORD   10                                  ; Plit count word
```

```
000010  172150                  P.AAB:  .WORD   -5630
000012  000154                          .WORD   154
000014  000004                          .WORD   4
000016  000060                          .WORD   60
000020  000000                          .WORD   0
000022  000000                          .WORD   0
000024  150477                          .WORD   -27301
000026  000000                          .WORD   0
000030  000060'                 P.AAC:  .WORD   L$LAST+54
000032  000010                          .WORD   10                      ; Plit count word
000034  172150                  P.AAD:  .WORD   -5630
000036  000154                          .WORD   154
000040  000004                          .WORD   4
000042  000001                          .WORD   1
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  001437                          .WORD   1437
000052  000000                          .WORD   0
000054  000104'                 P.AAE:  .WORD   L$LAST+100
000056  000010                          .WORD   10                      ; Plit count word
000060  172150                  P.AAF:  .WORD   -5630
000062  000154                          .WORD   154
000064  000004                          .WORD   4
000066  000002                          .WORD   2
000070  000000                          .WORD   0
000072  000000                          .WORD   0
000074  001437                          .WORD   1437
000076  000000                          .WORD   0
000100  000000                  P.AAG:  .WORD   0
000102  000010                          .WORD   10                      ; Plit count word
000104  172150                  P.AAH:  .WORD   -5630
000106  000154                          .WORD   154
000110  000004                          .WORD   4
000112  000003                          .WORD   3
000114  000000                          .WORD   0
000116  000000                          .WORD   0
000120  001437                          .WORD   1437
000122  000000                          .WORD   0
000124  000000                  T$FREE::.WORD   0


        000004'                 L$LAST==        BL$LAS+4
        000004                  T$PTHV==        4
        000004'                 $LAS5=          P.AAA
        000010'                 $REM5=          P.AAB
        000030'                 $LAS4=          P.AAC
        000034'                 $REM4=          P.AAD
        000054'                 $LAS3=          P.AAE
        000060'                 $REM3=          P.AAF
        000100'                 $$LAS1=         P.AAG
        000104'                 $REM2=          P.AAH
```

D5

```
;   0026  0          MAX_UNITS        = MAX_CTLR * UNITS_PER_CNTR,          ! MAXIMUM NUMBER OF UNITS TO TEST
;   0027  0                                                                                                  SEQ 0469
ZRQAM4          RD/RX EXERCISER                                  9-Jul-1984 08:27:11    VAX-11 Bliss-16 V4.0-579          Page 233
V01.8           LASTAD AND SETUP                                 6-Jul-1984 11:20:41    DISK$USER2:[POWERS]ZRQAE0.BL2;22      (62)

                                                .SBTTL  $END.LINK LASTAD AND SETUP
000000  000207                          $END.LINK::
                                                RTS     PC                                ;                                  8037

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0126
; Maximum stack depth per invocation:  0 words


;   8064  1     end
;   8065  1
;   8066  0     eludom


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;        $XYZ$                            44       RO ,  I  ,  LCL,  REL,  CON



;                                Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAE0.L16;18         405        7         1         21         00:00.1




;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAE0.BL2/LIST=ZRQAE0.LS2/OBJECT=ZRQAE0.OB2/SOURCE=PAGE:53

; Size:          7342 code + 447 data words
; Run Time:         03:46.6
; Elapsed Time:     05:04.0
; Lines/CPU Min:     2136
; Lexemes/CPU-Min: 20307
; Memory Used:  549 pages
; Compilation Complete
                                                                  9-Jul-1984 08:23:00    VAX-11 Bliss-16 V4.0-579          Page   1
                                                                 28-Jun-1984 15:12:21    DISK$USER2:[POWERS]ZRQAE0.REQ;5       (1)

;   0001  0     !LAST EDITED 28-JUN-84
;   0002  0     !*****************************************************************************
;   0003  0     !                                                                           *
;   0004  0     !                        L I T E R A L S                                    *
;   0005  0     !                                                                           *
;   0006  0     !*****************************************************************************
;   0007  0     !
;   0008  0     LITERAL
;   0009  0     !
;   0010  0     !****** ODT TRAP VECTOR LOCATION
;   0011  0     !
;   0012  0             O_TVEC          = %o'14',
;   0013  0     !
;   0014  0     !****** HARDWARE ADDRESSES ETC.
;   0015  0     !
;   0016  0             INIT_INTR_VECT  = %o'154',                          ! VECTOR ADDRESS
```

```
;    0015  0     !
;    0016  0             INIT_INTR_VECT  = %o'154',                          ! VECTOR ADDRESS
;    0017  0             INIT_IP_ADDR    = %o'172150',                       ! IP REGISTER ADDRESS
;    0018  0             INIT_BR_LEVEL   = %o'4',                            ! BUS REQUEST LEVEL
;    0019  0     !
;    0026  0             RD51_MAX_TRACK  = 1200,                             ! MAXIMUM NUMBER OF TRACKS FOR RD51
;    0028  0             TODR_CLOCK      = %o'177546',                       ! TIME-OF-CLOCK ADDRESS
;    0029  0             RD51_SEC_PER_TRK= 18.,                              ! NUMBER OF SECTORS PER TRACK FOR RD51
;    0030  0     !
;    0031  0             RD51_MAX_LBN    = RD51_MAX_TRACK * RD51_SEC_PER_TRK - 1.! MAX LBN FOR RD51
;    0032  0     !****** HARDWARE-LIMITS
;    0033  0     !
;    0034  0             RD52_MAX_TRACK  = 2976.,                            ! MAXIMUM NUMBER OF TRACKS FOR RD52
;    0035  0             MAX_CTLR        = 4,                                ! MAXIMUM NUMBER OF LED CONTROLLERS ALLOWED
;    0036  0             RD52_SEC_PER_TRK= 18.,                              ! NUMBER OF SECTORS PER TRACK FOR RD52
;    0037  0             MAX_UNITS_PER_CNTR = 8,                             ! MAXIMUM UNITS PER CONTROLLER
;    0038  0             RD52_MAX_LBN    = RD52_MAX_TRACK * RD52_SEC_PER_TRK - 1.! MAX LBN FOR RD52
;    0039  0
;    0040  0             RX50_MAX_TRACK  = 80.,                              ! MAXIMUM NUMBER OF TRACKS FOR RX50
;    0041  0             RX50_SEC_PER_TRK= 10.,                              ! NUMBER OF SECTORS PER TRACK FOR RX50
;    0042  0             RX50_MAX_LBN    = RX50_MAX_TRACK * RX50_SEC_PER_TRK - 1.! MAX LBN FOR RX50
;    0043  0
;    0044  0             BYTES_PER_SECT  = 512.,                             ! BYTES/SECTOR (AT PRESENT SAME FOR RDs AND RXs)
;    0045  0     !       MAX_XFER_SIZE   = 2 * BYTES_PER_SECT,               ! ARTBITRARY MAX SIZE OF EACH DISK I/O
;    0046  0             MAX_XFER_SIZE   = BYTES_PER_SECT * 3 / 2,
;    0047  0     !
;    0048  0     ! NOTE - BOTH OF THESE NUMBERS ARE NOW ARBITRARILY CHOSEN AS THE NUMBER OF LBNS CONTAINED PER UNIT/10 .
;    0049  0     !
;    0050  0     !****** RING SIZES
;    0051  0     !
;    0052  0             CR_LOG          = 2.,                               ! LOG2 LENGTH OF COMMAND RING
;    0053  0             RR_LOG          = 2.,                               ! LOG2 LENGTH OF RESPONSE RING
;    0054  0             CRING_LEN       = 1 ^ CR_LOG,                       ! COMMAND RING LENGTH
;    0055  0             RRING_LEN       = 1 ^ RR_LOG,                       ! RESPONSE RING LENGTH
;    0056  0     !
```

```
;    0054  0      !****** OFFSETS (IN WORDS)
;    0055  0      !
;    0056  0              OF_UN           = 3,                                    ! OFFSET FROM START OF CST TO FIRST UNIT
;    0057  0              OF_DATA         = 0,                                    ! OFFSET TO DISK UNIT FLAGS   WITHIN UNIT'S CST
;    0058  0              OF_BEG          = 1,                                    ! OFFSET TO BEGINNING BLK NO. WITHIN UNIT'S CST
;    0059  0              OF_BEG1         = 2,        !OFFSET TO START BK HI      ZZZ
;    0060  0              OF_END          = 3,        !OFFSET TO END BLOCK LO     ZZZ
;    0061  0              OF_END1         = 4,        !OFFSET TO END BK HI        ZZZ
;    0062  0              OF_NAME_0       = 5,        !OFFSET TO 1st 2 CHARS OF NAME  ZZZ
;    0063  0              OF_NAME_2       = 6,        !OFFSET TO 2nd 2 CHARS OF NAME  ZZZ
;    0064  0              OF_DUPFLAGS     = 8,        !OFFSET TO DUP FLAGS        ZZZ
;    0065  0              OF_COUNT        = 9,        !OFFSET TO MSCP FUNCTION COUNTER ZZZ
;    0066  0              OF_DBN          = 8,        !OFFSET TO RELATIVE DBN     ZZZ
;    0067  0      !
;    0068  0      !
;    0069  0      !****** TABLE AND OTHER STRUCTURE SIZES
;    0070  0      !
;    0071  0              LBNADR_LEN      = 2,                                    !MAX_LBN'S ARE 2 WD ADDRESSES
 ZZZ
;    0072  0              HWPT_LEN        = 8,                                    ! SIZE (WORDS) OF HW P-TABLE
 ZZZ
;    0073  0              COMM_LEN        = (RRING_LEN * 2) + (CRING_LEN * 2) + 4,! SIZE (WORDS) OF COMMUNICATION AREA PER CONTROLLER
;    0074  0              UNIT_SIZE       = 10,                                   ! SIZE (WORDS) OF CST UNIT ENTRY
 ZZZ
;    0075  0              CST_LEN         = UNITS_PER_CNTR * UNIT_SIZE + OF_UN,   ! SIZE (WORDS) OF A CONTROLLER STATUS TABLE
;    0076  0              TALLY_CLEAR     = 7,                                    ! SIZE (WORDS) OF STATISTICS TBL CLEARED EVERY PASS
;    0077  0              TALLY_TOTALS    = 20,                                   ! SIZE (WORDS) OF STATISTICS TABLE FOR TOTALS
 ZZZ
;    0078  0              TALLY_LEN       = TALLY_CLEAR + TALLY_TOTALS,           ! SIZE (WORDS) OF A STATISTICS TABLE
;    0079  0              C_ERR_LEN       = 1,                                    ! SIZE (WORDS) OF CONTROLLER ERROR TABLE
;    0080  0              RP_LEN          = 22,                                   ! SIZE (WORDS) OF A RETURN PACKET
;    0081  0              MSG_LEN         = 30,                                   ! SIZE (WORDS) OF AN MSCP MESSAGE (TEXT PORTION)
;    0082  0              PKT_LEN         = MSG_LEN + 5,                          ! SIZE (WORDS) OF AN MSCP PACKET
;    0083  0              DCT_LEN         = 9,                                    ! SIZE (WORDS) OF A DRIVER CONTROLLER TABLE
;    0084  0              RDM_LEN         = 16,                                   ! SIZE (WORDS) OF THE RANDOM NUMBER TABLE
;    0085  0              MAX_UDP_CNT     = 16,                                   ! MAX SIZE OF USER DATA PATTERN
;    0086  0              MAX_BUF_CNT     = (CRING_LEN * 2) * MAX_CTLR,           ! MAX NO. OF I/O BUFFERS (BUFF_ADDR & BUFF_OWN)
;    0087  0              PKT_CNT         = ((CRING_LEN * 2) + RRING_LEN) * MAX_CTLR,
;    0088  0                                                                      ! NO. OF MSCP PACKETS IN POOL
;    0089  0              RP_CNT          = PKT_CNT - (RRING_LEN * MAX_CTLR),     ! NO. OF RETURN PACKETS IN POOL
;    0090  0              IODQ_LEN        = RP_CNT,                               ! NO. OF ENTRIES IN I/O DONE QUEUE (IODQ)
;    0091  0              OUTC_CNT        = CRING_LEN * 2,                        ! NO. OF ENTRIES/CONTROLLER'S OUTSTANDING CMD LIST
;    0092  0              DP_CNT          = 21,                                   ! NO. OF PRE-DEFINED DATA PATTERNS
;    0093  0              EP_CNT          = MAX_CTLR * RRING_LEN * 3,             ! NO. OF ERROR-LOG PACKET SAVE BUFFERS
;    0094  0              EP_LEN          = PKT_LEN - 3 + 1,                      ! LENGTH OF EACH ERROR-LOG SAVE BUFFER
;    0095  0              LAST_PKT_LEN    = 3,                                    ! BUFFER LENGTH TO SAVE INFO. ABOUT LAST RESPONSE
;    0096  0              TOO_MANY_READS  = 2,                                    ! FOR READ/WRITE BALANCE WITH HOST READ COMPARES  ZZ
Z
;    0097  0      !
;    0098  0      !****** SW P-TABLE FLAGS (SWP_FLAGS)
;    0099  0      !
;    0100  0              SWF_TRC         = %o'000001',                           ! DIAGNOSTIC TRACE
;    0101  0              SWF_RDM         = %o'000002',                           ! RANDOM SEEK MODE
;    0102  0              SWF_CRC         = %o'000004',                           ! READ-COMPARE AT CONTROLLER
;    0103  0      !       SWF_DCC         = %o'000010',                           ! DRIVE COMPLEMENT COMPLETE
;    0104  0              SWF_CWC         = %o'000020',                           ! WRITE-COMPARE AT CONTROLLER
;    0105  0              SWF_HWC         = %o'000040',                           ! WRITE-COMPARE AT HOST
;    0106  0              SWF_UDP         = %o'000100',                           ! USER-DEFINED DATA PATTERN
```

```
0          SWF_CST        = %o'000200',                          ! CLEAR STATISTICAL TABLES
0    !     SWF_DIA        = %o'000400',                          ! DIAGNOSTIC PACKAGE, WHEN THIS IS SELECTED
0                                                                ! ALL INTERRUPTS ARE WAITED FOR, E.G. ONLY
0                                                                ! ONE MSCP PACKET IS OUTSTANDING AT A TIME
0          SWF_SEQ        = %o'001000',                          ! RANDOM OR FIXED SEQUENTIAL STEPPING
0    !     SWF_DUP        = %o'002000',                          ! RUN DUP DIAGNOSTIC
0          SWF_FER        = %o'004000',                          ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT DETECTED
0          SWF_HRD        = %o'010000',                          ! HALT ON HARD ERRORS ALSO WITH 'HOE' DRS FLAG?
0          SWF_SFT        = %o'020000',                          ! HALT ON SOFT ERRORS ALSO WITH 'HOE' DRS FLAG?
0          SWF_BLK        = %o'040000',                          ! HALT ON BAD-BLOCK ERRORS ALSO WITH 'HOE' DRS FLAG?
0          SWF_TRY        = %o'100000',                          ! COUNT EACH RETRY AS ANOTHER EXTRA SOFT-ERROR
0    !
0    !****** FLAGS FOR DUP EXERCISER (DUP_FLAGS)                                ZZZ
0    !                                                                         ZZZ
0          SWP_DINT       = %o'2',                !DUP CAUSED INIT           ZZZ
0
0    !
0    !****** ENTRY_REASON VALUES
0    !              (HOW PROGRAM WAS INVOKED)
0    !
0          START          = 1,                                   ! START
0          RESTART        = 2,                                   ! RESTART
0          CONT           = 3,                                   ! CONTINUE
0          PWR_FAIL       = 4,                                   ! POWER FAIL
0          NEW_PASS       = 5,                                   ! NEW PASS
0    !
0    !****** DROP UNIT REASONS
0    !              (LOADED INTO DUR VECTOR)
0    !
0          DU_USER        = 0,                                   ! USER COMMAND
0          DU_CONF        = 1,                                   ! CONFIGURATION ERROR
0          DU_INIT        = 2,                                   ! INITIALIZATION ERROR
0          DU_XFER        = 3,                                   ! TRANSFER LIMIT REACHED
0          DU_HERR        = 4,                                   ! HARD ERROR LIMIT REACHED
0          DU_DFATAL      = 5,                                   ! UNRECOVERABLE DEVICE ERROR
0          DU_CFATAL      = 6,                                   ! UNRECOVERABLE CONTROLLER ERROR
0          DU_ONLINE      = 7,                                   ! ONLINE FAILED
0          DU_ACCESS      = 8,                                   ! ACCESS TO LAST TRACK FAILED
0          DU_PROTECT     = 9,                                   ! WRITE PROTECT CONFLICT
0          DU_TIME        = 10,                                  ! COMMAND TIME OUT
0    !
0    !****** MISCELLANEOUS LITERALS
0    !
0          INI_ATT        = 2,                                   ! NO. OF HW INIT ATTEMPTS BEFORE FAILURE IS ASSUMED
0          WR_RING        = ((%o'200') or (CR_LOG + 3) or (RR_LOG)),
0                                                                ! WR-BIT-AND-RING-LENGTH (STEP 1 WRITE/STEP 2 READ)
0          QIO_PER_CTLR   = CRING_LEN * 2,                       ! MAXIMUM NUMBER OF OUTSTANDING QIOS PER CONTROLLER
0    !     MAX_XFER       = 256,                                 ! MAXIMUM SIZE (WORDS) OF AN I/O TRANSFER
0          REMOVABLE_BIT  = %o'0',                               ! BIT IN HARDWARE TABLES MARKING A REMOVABLE DISK
0          FIXED_BIT      = %o'20',                              ! BIT IN HARDWARE TABLES MARKING A FIXED     DISK
0          REMOVABLE      = 0,                                   ! NUMBER FOR REMOVABLE DISK WHEN SHIFTED RIGHT
0          FIXED          = 1,                                   ! NUMBER FOR FIXED     DISK WHEN SHIFTED RIGHT
0          RX_50          = 0,                    !D_TYPE FLAG = 0 FOR RX50 ZZZ
```

```
;   0160  0             RD_51           = 1,                    !D_TYPE FLAG = 1 FOR RD51 ZZZ
;   0161  0             RD_52           = 2,                    !              2 FOR RD52 ZZZ
;   0162  0     !
;   0163  0     !****** MSCP PACKET DESCRIPTOR
;   0164  0     !
;   0165  0             ED_OWN          = %o'100000',                   ! OWNERSHIP BIT
;   0166  0             ED_FLAG         = %o'040000',                   ! FLAG BIT
;   0167  0     !
;   0168  0     !****** MSCP COMMAND PACKET OPCODES
;   0169  0     !
;   0170  0             OP_MSK          = %o'177',                      ! OPCODE MASK
;   0171  0             OP_END          = %o'200',                      ! ENDCODE DESIGNATOR
;   0172  0             OP_ACC          = %o'20',                       ! ACCESS COMMAND
;   0173  0             OP_ONL          = %o'11',                       ! ONLINE COMMAND
;   0174  0             OP_RD           = %o'41',                       ! READ COMMAND
;   0175  0             OP_SCC          = %o'4',                        ! SET CONTROLLER CHARACTERISTICS COMMAND
;   0176  0             OP_WRT          = %o'42',                       ! WRITE COMMAND
;   0177  0             OP_GDS          = %o'1',            !get dust status            ZZZ
;   0178  0             OP_ESP          = %o'2',            !execute supplied prog      ZZZ
;   0179  0             OP_ELP          = %o'3',            !execute local program      ZZZ
;   0180  0             OP_SDD          = %o'4',            !send data                  ZZZ
;   0181  0             OP_RCD          = %o'5',            !receive data               ZZZ
;   0182  0             OP_ABT          = %o'6',            !abort program              ZZZ
;   0183  0
;   0184  0     !
;   0185  0     !****** PACKET SIZES
;   0186  0     !
;   0187  0             SZ_ACC = %decimal '32',                         ! ACCESS
;   0188  0             SZ_ONL = %decimal '36',                         ! ON LINE COMMAND
;   0189  0             SZ_RD  = %decimal '32',                         ! READ
;   0190  0             SZ_SCC = %decimal '32',                         ! SET CONTROLLER CHARACTERISTICS
;   0191  0             SZ_WRT = %decimal '32',                         ! WRITE
;   0192  0             SZ_GEN = %decimal '32',                         ! GENERAL PACKET SIZE
;   0193  0             SZ_REC = %DECIMAL '28',             !                       ZZZ
;   0194  0             SZ_SEN = %DECIMAL '28',             !                       ZZZ
;   0195  0             SZ_ELP = %DECIMAL '18',             !                       ZZZ
;   0196  0             SZ_ABT = %DECIMAL '12',             !                       ZZZ
;   0197  0             SZ_GDS = %DECIMAL '12',             !                       ZZZ
;   0198  0     !
;   0199  0     !****** MSCP COMMAND MODIFIERS
;   0200  0     !
;   0201  0             MD_CMP          = %o'040000',                   ! COMPARE
;   0202  0             MD_EXP          = %o'100000',                   ! EXPRESS REQUEST
;   0203  0     !
;   0204  0     !****** CONNECTION ID VALUES (MSCP_PKT, RETPKT)
;   0205  0     !               (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
;   0206  0     !
;   0207  0             CID_DISK        = 0,                            ! DISK MSCP
;   0208  0             CID_MSCP        = 0,                            !DISK MSCP
 !ZZZ
;   0209  0             CID_TAPE        = 1,                            ! TAPE MSCP
;   0210  0             CID_DUP         = 2,                            ! DIAGNOSTIC AND UTILITIES PROTOCOL
;   0211  0             CID_DRIVER      = 3,                            ! EXERCISER "DRIVER"
;   0212  0     !
```

```
;  0213  0     !****** MESSAGE TYPE VALUES
;  0214  0     !
;  0215  0             MT_SEQ          = 0,                        ! SEQUENTIAL (FROM PORT)
;  0216  0             MT_DG           = 1,                        ! DATAGRAM (FROM PORT)
;  0217  0             MT_CRD          = 2,                        ! CREDIT NOTIFICATION (FROM PORT)
;  0218  0             MT_FATAL        = 3,                        ! FATAL DEVICE ERROR (FROM "DRIVER")
;  0219  0             MT_TIMEOUT      = 4,                        ! COMMAND TIMEOUT (FROM "DRIVER")
;  0220  0     !
;  0221  0     !****** CONTROLLER FLAGS
;  0222  0     !               (IN SET CONTROLLER CHARACTERISTICS COMMAND AND RESPONSE)
;  0223  0     !
;  0224  0             CF_ATN          = %o'000200',               ! ENABLE ATTENTION MESSAGES
;  0225  0             CF_MSC          = %o'000100',               ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
;  0226  0             CF_OTH          = %o'000040',               ! ENABLE OTHER HOST'S ERROR LOG MESSAGES
;  0227  0             CF_THS          = %o'000020',               ! ENABLE THIS HOST'S ERROR LOG MESSAGES
;  0228  0     !       CF_MASK         = CF_ATN or CF_MSC or CF_THS,
;  0229  0             CF_MASK         = CF_MSC or CF_THS,         ! RELEVENT BITS IN CTLR FLAGS WORD
;  0230  0     !
;  0231  0     !****** UNIT FLAGS
;  0232  0     !               (IN ONLINE COMMAND AND RESPONSE)
;  0233  0     !
;  0234  0             UF_REMOVABLE    = %o'000200',               ! REMOVABLE MEDIA
;  0235  0             UF_WPH          = %o'020000',               ! WRITE PROTECT (HARDWARE)
;  0236  0     !
;  0237  0     !****** STATUS / EVENT CODE DEFINITIONS
;  0238  0     !
;  0239  0             ST_SUC          = %o'0',                    ! SUCCESS
;  0240  0             ST_CMD          = %o'1',                    ! INVALID COMMAND
;  0241  0             ST_ABO          = %o'2',                    ! COMMAND ABORTED
;  0242  0             ST_OFL          = %o'3',                    ! UNIT OFFLINE
;  0243  0             ST_AVL          = %o'4',                    ! DRIVE AVAILABLE
;  0244  0             ST_MFE          = %o'5',                    ! MEDIA FORMAT ERROR
;  0245  0             ST_WPT          = %o'6',                    ! WRITE PROTECTED
;  0246  0             ST_CMP          = %o'7',                    ! COMPARE ERROR
;  0247  0             ST_DAT          = %o'10',                   ! DATA ERROR
;  0248  0             ST_HST          = %o'11',                   ! HOST BUFFER ACCESS ERROR
;  0249  0             ST_CNT          = %o'12',                   ! CONTROLLER ERROR
;  0250  0             ST_DRV          = %o'13',                   ! DRIVE ERROR
;  0251  0             ST_DIA          = %o'37',                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  0252  0     !
;  0253  0     !****** END MESSAGE FLAGS
;  0254  0     !
;  0255  0             EF_BBR          = %o'200',                  ! BAD BLOCK REPORTED
;  0256  0             EF_BBU          = %o'100',                  ! BAD BLOCK NOT REPORTED
;  0257  0     !
;  0258  0     !****** RDRX LITERALS
;  0259  0     !
;  0260  0             RCIP            = 0,                        ! IP REGISTER
;  0261  0             RCSA            = 1,                        ! SA REGISTER
;  0262  0     !
;  0263  0     !****** COMMON SA REGISTER BIT DEFINITIONS
;  0264  0     !
;  0265  0             SA_S1           = %o'004000',               ! STEP 1 STATUS BIT
```

```
0          SA_S2          = %o'010000',                    !   :  2
0          SA_S3          = %o'020000',                    !   :  3
0          SA_S4          = %o'040000',                    !   V  4
0          SA_ERR         = %o'100000',                    ! ERROR INDICATOR
0          SA_INT         = %o'000200',                    ! INTERRUPT ENABLE DURING INITIALIZATION
0          SA_GO          = %o'000001',                    ! GO BIT TO START FIRMWARE
0     !
0     !****** INITIALIZATION STEP READ MASKS
0     !
0          S1_MASK        = %o'176000',                    ! STEP 1 READ BITS
0          S2_MASK        = %o'174377',                    !   :  2
0          S3_MASK        = %o'174377',                    !   :  3
0          S4_MASK        = %o'174000',                    !   V  4
0     !
0     !****** COMMAND TYPES
0     !
0          IMM_CMD        = 0,                             ! IMMEDIATE COMMAND
0          SEQ_CMD        = 1,                             ! SEQUENTIAL COMMAND
0          NON_SEQ_CMD    = 2,                             ! NON-SEQUENTIAL COMMAND
0     !
0     !****** ERROR-LOG FORMAT TYPES
0     !
0          FORMAT_CNTR    = %o'0',                         ! CONTROLLER ERROR
0          FORMAT_HOST    = %o'1',                         ! HOST MEMORY ACCESS ERROR
0          FORMAT_XFER    = %o'2',                         ! DISK TRANSFER ERROR
0          FORMAT_SDI     = %o'3',                         ! 'STANDARD DISK INTECONNECT' ERROR
0          FORMAT_SDE     = %o'4',                         ! SMALL DISK ERROR
0     !
0     !****** ERROR-LOG BLOCK NUMBER INFORMATION
0     !
0          TYPE_LBN       = %o'0000',                      ! LOGICAL BLOCK NUMBER
0          TYPE_RBN       = %o'0110',                      ! REPLACEMENT BLOCK NUMBER
0     !
0     !****** MSCP DISK MODEL CODES
0     !
0     !    MODEL_RX50     = 7,                             ! RX50
0     !    MODEL_RD51     = 6,                             ! RD51
0     !    MODEL_RD52     = 8,                             ! RD52
0     !
0     !****** LITERALS FOR READABILITY
0     !
0          YES            = 1,
0          NO             = 0,
0          TRUE           = 1,
0          FALSE          = 0,
0          SUCCESS        = 1,
0          FAILURE        = 0,
0          FOUND          = 1,
0          NOT_FOUND      = 0,
0          PRESENT        = 1,                             ! DISK IS PRESENT IN CONTROLLER
0          NOT_PRESENT    = 0,                             ! DISK IS NOT PRESENT IN CONTROLLER
0          UNPROTECTED    = 1,                             ! DISK HAS UNPROTECTED CUSTOMER LBN'S
0          PROTECTED      = 0,                             ! DISK HAS PROTECTED CUSTOMER LBN'S
```

```
;    0319  0          ONLINE          = 1,
;    0320  0          OFFLINE         = 0,
;    0321  0          IDLE            = 0,                    !IDLE
 !ZZZ
;    0322  0          ACTIVE          = 1,                    !ACTIVE
 !ZZZ
;    0323  0          FULL            = 1,                    ! ERROR-LOG SAVE PACKET FILLED
;    0324  0          EMPTY           = 0,                    ! ERROR-LOG SAVE PACKET PRINTED
;    0325  0          HRD_OCCURED     = 1,                    ! HARD ERROR DETECTED IN RESPONSE PACKET
;    0326  0          HRD_NOT_OCCURED = 0,                    ! HARD ERROR NOT DETECTED
;    0327  0          ALL_ONES        = %o'177777';
```

```
;  0328  0     !*****************************************************************************
;  0329  0     !                                                                            *
;  0330  0     !                        F I E L D S                                         *
;  0331  0     !                                                                            *
;  0332  0     !*****************************************************************************
;  0333  0     !
;  0334  0     FIELD
;  0335  0     !
;  0336  0     !****** HARDWARE P-TABLE FIELDS
;  0337  0     !
;  0338  0     HWP_FIELDS =
;  0339  0             set
;  0340  0             HWP_IP_ADDR     = [0, 0, 16, 0],                ! IP ADDRESS
;  0341  0             HWP_VECTOR      = [1, 0, 16, 0],                ! VECTOR ADDRESS
;  0342  0             HWP_BR_LEVEL    = [2, 0, 16, 0],                ! BUS REQUEST LEVEL
;  0343  0             HWP_DISK        = [3, 0, 16, 0],                ! DISK (ALL FIELDS)
;  0344  0             HWP_DISK_NUM    = [3, 0, 4, 0],                 ! DISK NUMBER
;  0345  0             HWP_DISK_TYPE   = [3, 4, 1, 0],                 ! DISK TYPE
;  0346  0             HWP_DISK_DUPEX  = [3, 5, 1, 0],                 !RUN DUP EXERCISER              !ZZZ
;  0347  0             HWP_DISK_DUPWT  = [3, 6, 1, 0],                 !DUP WRITE FLAG                 !ZZZ
;  0348  0             HWP_ENTIRE      = [3, 7, 1, 0],                 !TEST ENTIRE DISK               !ZZZ
;  0349  0             HWP_DISK_CP     = [3, 15, 1, 0],                ! PROTECT CUSTOMER DATA BIT
;  0350  0             HWP_BEG_TRK     = [4, 0, 16, 0],                ! BEGINNING TRACK LO            !ZZZ
;  0351  0             HWP_BEG_TRK1    = [5, 0, 16, 0],                ! BEGINNING TRACK HI            !ZZZ
;  0352  0             HWP_END_TRK     = [6, 0, 16, 0],                ! ENDING TRACK LO               !ZZZ
;  0353  0             HWP_END_TRK1    = [7, 0, 16, 0]                 ! ENDING TRACK HI               !ZZZ
;  0354  0             tes,
;  0355  0     !
;  0356  0     !****** COMMUNICATION AREA HEADER FIELDS
;  0357  0     !
;  0358  0     COM_FIELDS =
;  0359  0             set
;  0360  0             ADAP_CH         = [1, 8, 8, 0],                 ! ADAPTER CHANNEL NUMBER FOR PURGES
;  0361  0             CMD_INT         = [2, 0, 16, 0],                ! COMMAND RING INTERRUPT
;  0362  0             RSP_INT         = [3, 0, 16, 0]                 ! RESPONSE RING INTERRUPT
;  0363  0             tes,
;  0364  0
;  0365  0     !                                                                               ZZZ
;  0366  0     !       DUP BUFFER FIELD                                                        ZZZ
;  0367  0     !                                                                               ZZZ
;  0368  0     DP_FIELDS =                                                                     !ZZZ
;  0369  0             SET                                                                     !ZZZ
;  0370  0             DUPBF0          = [0, 0, 16, 0],                                        !ZZZ
;  0371  0             DUPBF1          = [1, 0, 16, 0],                                        !ZZZ
;  0372  0             DUPBF2          = [2, 0, 16, 0],                                        !ZZZ
;  0373  0             DUPTYPE         = [0, 12, 4, 0],                                        !ZZZ
;  0374  0             DUPMESG         = [0, 0, 12, 0]                                         !ZZZ
;  0375  0             TES,                                                                    !ZZZ
;  0376  0
;  0377  0
;  0378  0     !****** CONTROLLER STATUS TABLE (CST) FIELDS
;  0379  0     !
;  0380  0     CST_FIELDS =
```

```
;  0381  0          set
;  0382  0          IP_ADDR         = [0, 0, 16, 0],                ! IP ADDRESS
;  0383  0          VEC_ADDR        = [1, 0, 9, 0],                 ! VECTOR ADDRESS
;  0384  0          STATE           = [1, 15, 1, 0],                ! CONTROLLER STATUS
;  0385  0          BR_LEV          = [2, 0, 8, 0],                 ! BUS REQUEST LEVEL
;  0386  0          U_CNT           = [2, 8, 8, 0]                  ! NUMBER OF UNITS (DISKS) FOR THIS CONTROLLER
;  0387  0      !
;  0388  0      !   DO_ALL          = [3, 0, 16, 0],                ! DISK 0 (ALL FIELDS)
;  0389  0      !   DO_DISK_NUM     = [3, 0, 4, 0],                 ! DISK NUMBER
;  0390  0      !   DO_TYPE         = [3, 4, 1, 0],                 ! DISK TYPE
;  0391  0      !   DO_UNIT         = [3, 8, 4, 0],                 ! DISK 0 UNIT NUMBER (DRS UNIT)
;  0392  0      !   DO_FATAL        = [3, 12, 1, 0],                ! DISK 0 FATAL ERROR BIT
;  0393  0      !   DO_STAT         = [3, 13, 1, 0],                ! DISK 0 STATUS BIT
;  0394  0      !   DO_PRES         = [3, 14, 1, 0],                ! DISK 0 PRESENT BIT
;  0395  0      !   DO_PROT         = [3, 15, 1, 0],                ! DK 0 PROTECT CUSTOMER DATA
;  0396  0      !   DO_BEG0         = [4, 0, 16, 0],    !DK 0 BEGIN TK LO          ZZZ
;  0397  0      !   DO_BEG1         = [5, 0, 16, 0],    !DK 0 BEGIN TK HI          ZZZ
;  0398  0      !   DO_END0         = [6, 0, 16, 0],    !DK 0 END TK LO            ZZZ
;  0399  0      !   DO_END1         = [7, 0, 16, 0],    !DK 0 END TK HI            ZZZ
;  0400  0      !   DO_NAME0        = [8, 0, 8, 0],     !DK 0 MANE BYTE 0          ZZZ
;  0401  0      !   DO_NAME1        = [8, 8, 8, 0],     !DK 0 NAME BYTE 1          ZZZ
;  0402  0      !   DO_NAME2        = [9, 0, 8, 0],     !DK 0 NAME BYTE 2          ZZZ
;  0403  0      !   DO_NAME3        = [9, 8, 8, 0],     !DK 0 NAME BYTE 3          ZZZ
;  0404  0      !   DO_NUL          = [10, 0, 16, 0]    !NUL AFTER NAME            ZZZ
;  0405  0      !   DO_DBN          = [11, 0, 8, 0],    !DK 0 RELATIVE DBN         ZZZ
;  0406  0      !   DO_WRITE        = [11, 12, 1, 0],   !DK 0 DUP WRITE FLAG       ZZZ
;  0407  0      !   DO_ACTIVE       = [11, 13, 1, 0],   !DK 0 ACTIVE FLAG          ZZZ
;  0408  0      !   DO_DUPERROR     = [11, 14, 1, 0],   !DK 0 DUP ERROR FLAG       ZZZ
;  0409  0      !   DONODUPMED      = [11, 15, 1, 0],   !DK 0 NO DUP MEDIA FLAG    ZZZ
;  0410  0      !   DO_COUNT        = [12, 0, 16, 0],   !DK 0 RELATIVE MSCP FUN-   ZZZ
;  0411  0      !                                           CTION COUNTER          ZZZ
;  0412  0      !                                                                  ZZZ
;  0413  0      !       ! REPEAT WORDS 3 THROUGH 12 ABOVE AS: !                    ZZZ
;  0414  0      !       ! WORDS 13 THROUGH 21 FOR DRIVE 1     !                    ZZZ
;  0415  0      !       ! WORDS 22 THROUGH 30 FOR DRIVE 2     !                    ZZZ
;  0416  0      !       ! WORDS 31 THROUGH 39 FOR DRIVE 3     !                    ZZZ
;  0417  0      !       !                                     !                    ZZZ
;  0418  0      !                                                                  ZZZ
;  0419  0          tes,
;  0420  0      !
;  0421  0      !****** MSCP PACKET FIELDS
;  0422  0      !           (NOTE: BASE ADDRESS OF PACKET REFERENCES THE PACKET'S OWN
;  0423  0      !           BUFFER DESCRIPTOR, RATHER THAN THE MESSAGE BODY (TEXT + 0).
;  0424  0      !           SEE DOCUMENTATION FOR LAYOUT OF MSCP PACKETS.)
;  0425  0      !
;  0426  0      PKT_FIELDS =
;  0427  0          set
;  0428  0      !
;  0429  0      !   HEADER FIELDS
;  0430  0      !
;  0431  0          PKT_LO          = [0, 0, 16, 0],                ! PACKET DESCRIPTOR (LO ORDER)
;  0432  0          PKT_HI          = [1, 0, 16, 0],                ! PACKET DESCRIPTOR (HI ORDER - ALL FIELDS)
;  0433  0          PKT_U           = [1, 0, 2, 0],                 ! PACKET DESCRIPTOR (HI ORDER UNIBUS BITS)
```

```
;  0434  0          PKT_Q           = [1, 2, 4, 0],                 ! PACKET DESCRIPTOR (HI ORDER Q-BUS BITS)
;  0435  0          PKT_F           = [1, 14, 1, 0],                ! PACKET DESCRIPTOR FLAG BIT
;  0436  0          PKT_O           = [1, 15, 1, 0],                ! PACKET DESCRIPTOR OWNERSHIP BIT
;  0437  0          CMD_TYPE        = [2, 0, 8, 0],                 ! COMMAND TYPE
;  0438  0          RSP_RECEIVED    = [2, 8, 8, 0],                 ! FLAG SET IF RESPONSE TO COMMAND RECEIVED
;  0439  0          MSGLEN          = [3, 0, 16, 0],                ! MESSAGE LENGTH
;  0440  0          CREDITS         = [4, 0, 4, 0],                 ! CREDITS
;  0441  0          MSGTYP          = [4, 4, 4, 0],                 ! MESSAGE TYPE
;  0442  0          CONNID          = [4, 8, 8, 0],                 ! CONNECTION ID
;  0443  0      !
;  0444  0      !   GENERIC COMMAND PACKET AND END PACKET HEADER FILEDS
;  0445  0      !
;  0446  0          CRN_LO          = [5, 0, 16, 0],                ! COMMAND REF NUMBER (LO ORDER)
;  0447  0          CRN_HI          = [6, 0, 16, 0],                ! COMMAND REF NUMBER (HI ORDER)
;  0448  0          DK_NUM          = [7, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0449  0          OPCODE          = [9, 0, 8, 0],                 ! OPCODE AND ENDCODE
;  0450  0          MODIFY          = [10, 0, 16, 0],               ! COMMAND MODIFIERS
;  0451  0          STATUS_CODE     = [10, 0, 5, 0],                ! STATUS (PART OF RESPONSE PACKET)
;  0452  0          STATUS_SUBCODE  = [10, 5, 11, 0],               ! SUBCODE (PART OF RESPONSE PACKET)
;  0453  0      !
;  0454  0      !   READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)
;  0455  0      !
;  0456  0          BC_LO           = [11, 0, 16, 0],               ! BYTE COUNT (LO ORDER)
;  0457  0          BC_HI           = [12, 0, 16, 0],               ! BYTE COUNT (HI ORDER)
;  0458  0          BUF_0           = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR
;  0459  0          BUF_1           = [14, 0, 16, 0],               !       :
;  0460  0          BUF_2           = [15, 0, 16, 0],               !       :
;  0461  0          BUF_3           = [16, 0, 16, 0],               !       :
;  0462  0          BUF_4           = [17, 0, 16, 0],               !       :
;  0463  0          BUF_5           = [18, 0, 16, 0],               !       V
;  0464  0          LBN_L           = [19, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0465  0          LBN_H           = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0466  0
;  0467  0      !
;  0468  0      ! DUP PROGRAM LETTER FIELDS (FOR EXECUTE LOCAL PROGRAM CMD)          ZZZ
;  0469  0      !                                                                    ZZZ
;  0470  0          L1              = [11, 0, 8, 0],        !LETTER NO 1            ZZZ
;  0471  0          L2              = [11, 8, 8, 0],        !LETTER NO 2            ZZZ
;  0472  0          L3              = [12, 0, 8, 0],        !LETTER NO 3            ZZZ
;  0473  0          L4              = [12, 8, 8, 0],        !LETTER NO 4            ZZZ
;  0474  0          L5              = [13, 0, 8, 0],        !LETTER NO 5            ZZZ
;  0475  0          L6              = [13, 8, 8, 0],        !LETTER NO 6            ZZZ
;  0476  0      !
;  0477  0      !   SET CONTROLLER CHARACTERISTICS COMMAND FIELDS
;  0478  0      !
;  0479  0          C_FLAGS         = [12, 0, 16, 0],               ! CONTROLLER FLAGS
;  0480  0      !
;  0481  0      !   ONLINE COMMAND FIELDS
;  0482  0      !
;  0483  0          U_FLAGS         = [12, 0, 16, 0],               ! UNIT FLAGS
;  0484  0          DDPAR           = [19, 0, 16, 0]                ! DEVICE-DEPENDENT PARAMETERS
;  0485  0          tes,
;  0486  0      !
```

```
;  0487  0     !****** RETURN PACKET (RETPKT) FIELDS
;  0488  0     !               (SIMILAR, BUT NOT IDENTICAL, TO MSCP PACKET FIELDS)
;  0489  0     !
;  0490  0     RP_FIELDS =
;  0491  0             set
;  0492  0     !
;  0493  0     !       COMMON TO ALL RETURN PACKETS FROM DISK MSCP
;  0494  0     !
;  0495  0             MESLEN          = [0, 0, 16, 0],                ! MESSAGE LENGTH
;  0496  0             CTLR            = [1, 0, 4, 0],                 ! CONTROLLER NUMBER (CREDITS OVERWRITTEN)
;  0497  0             MESTYP          = [1, 4, 4, 0],                 ! MESSAGE TYPE
;  0498  0             CONID           = [1, 8, 8, 0],                 ! CONNECTION ID
;  0499  0             CRF_LO          = [2, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (LO ORDER)
;  0500  0             CRF_HI          = [3, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (HI ORDER)
;  0501  0             DISK            = [4, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0502  0             CMDMOD          = [5, 0, 16, 0],                ! COMMAND MODIFIERS
;  0503  0             ENDCOD          = [6, 0, 8, 0],                 ! END CODE
;  0504  0             FLAGS           = [6, 8, 8, 0],                 ! FLAGS
;  0505  0             STATUS          = [7, 0, 16, 0],                ! STATUS AND SUB-CODE
;  0506  0             STSCOD          = [7, 0, 5, 0],                 ! STATUS CODE
;  0507  0             SUBCOD          = [7, 5, 11, 0],                ! SUB-CODE
;  0508  0     !
;  0509  0     !       READ, WRITE, AND ACCESS COMMAND RETURN PACKETS
;  0510  0     !
;  0511  0             BCNT_LO         = [8, 0, 16, 0],                ! BYTE COUNT (LO ORDER)
;  0512  0             BCNT_HI         = [9, 0, 16, 0],                ! BYTE COUNT (HI ORDER)
;  0513  0             BUFF_0          = [10, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 0)
;  0514  0             BUFF_1          = [11, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 1)
;  0515  0             BUFF_2          = [12, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 2)
;  0516  0             BUFF_3          = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 3)
;  0517  0             BUFF_4          = [14, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 4)
;  0518  0             BUFF_5          = [15, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 5)
;  0519  0             BBLK_LO         = [16, 0, 16, 0],               ! FIRST BAD BLOCK (LO ORDER)
;  0520  0             BBLK_HI         = [17, 0, 16, 0],               ! FIRST BAD BLOCK (HI ORDER)
;  0521  0             CBCNT_LO        = [18, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (LO ORDER)
;  0522  0             CBCNT_HI        = [19, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (HI ORDER)
;  0523  0             LBN_LO          = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0524  0             LBN_HI          = [21, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0525  0     !
;  0526  0     !       SET CONTROLLER CHARACTERISTICS RETURN PACKET
;  0527  0     !
;  0528  0             C_FLGS          = [9, 0, 16, 0],                ! CONTROLLER FLAGS
;  0529  0             C_TIME          = [10, 0, 16, 0],               ! CONTROLLER TIMEOUT
;  0530  0     !
;  0531  0     !       UNIT ONLINE RETURN PACKET
;  0532  0     !
;  0533  0             U_FLGS          = [9, 0, 16, 0],                ! UNIT FLAGS
;  0534  0             R_MODEL         = [13, 0, 8, 0],                ! 2 DIGIT MODEL NUMBER                          ZZZ
;  0535  0             NAME_NUM        = [14, 0, 6, 0],                ! MODEL NAME - 2 DIGIT NUMBER
;  0536  0             NAME_1_LO       = [14, 12, 4, 0],               ! MODEL NAME - 2ND CHARACTER (LOW  ORDER 4 BITS)
;  0537  0             NAME_1_HI       = [15, 0, 1, 0],                ! MODEL NAME - 2ND CHARACTER (HIGH ORDER 1 BIT)
;  0538  0             NAME_0          = [15, 1, 5, 0],                ! MODEL NAME - IST CHARACTER
;  0539  0     !ZZZ    USIZ_LO         = [18, 0, 16, 0],               ! UNIT SIZE (LO ORDER)
```

```
;  0540  0     !ZZZ    USIZ_HI         = [19, 0, 16, 0],          ! UNIT SIZE (HI ORDER)
;  0541  0             SIZE0           = [18, 0, 16, 0],          !LOWER WD OF MAX LBNS OR UNIT SIZE      ZZZ
;  0542  0             SIZE1           = [19, 0, 16, 0]           !UPPER WD     "          "      "      ZZZ
;  0543  0             tes,
;  0544  0     !
;  0545  0     !****** STATISTICS TABLE (TALLY) FIELDS
;  0546  0     !
;  0547  0     T_FIELDS =
;  0548  0             set
;  0549  0             BYTES_READ_LO   = [0, 0, 16, 0],           ! NUMBER OF BYTES READ (LO ORDER)
;  0550  0             BYTES_READ_HI   = [1, 0, 16, 0],           ! NUMBER OF BYTES READ (HI ORDER)
;  0551  0             MBYTES_READ     = [2, 0, 16, 0],           ! MEGABYTES READ
;  0552  0             BYTES_WRIT_LO   = [3, 0, 16, 0],           ! NUMBER OF BYTES WRITTEN (LO ORDER)
;  0553  0             BYTES_WRIT_HI   = [4, 0, 16, 0],           ! NUMBER OF BYTES WRITTEN (HI ORDER)
;  0554  0             MBYTES_WRIT     = [5, 0, 16, 0],           ! MEGABYTES WRITTEN
;  0555  0             ERR_HARD        = [6, 0, 16, 0],           ! NUMBER OF HARD ERRORS
;  0556  0     !
;  0557  0             TOT_READS_LO    = [7, 0, 16, 0],           ! TOTAL NUMBER OF READS (LO ORDER)
;  0558  0             TOT_READS_HI    = [8, 0, 16, 0],           ! TOTAL NUMBER OF READS (HI ORDER)
;  0559  0             TOT_WRITES_LO   = [10, 0, 16, 0],          ! TOTAL NUMBER OF WRITES (LO ORDER)
;  0560  0             TOT_WRITES_HI   = [11, 0, 16, 0],          ! TOTAL NUMBER OF WRITES (HI ORDER)
;  0561  0             TOT_BYT_RED_LO  = [13, 0, 16, 0],          ! TOTAL BYTES READ (LO ORDER)
;  0562  0             TOT_BYT_RED_HI  = [14, 0, 16, 0],          ! TOTAL BYTES READ (HI ORDER)
;  0563  0             MTOT_BYT_RED    = [15, 0, 16, 0],          ! TOTAL MEGABYTES READ
;  0564  0             TOT_BYT_WRT_LO  = [16, 0, 16, 0],          ! TOTAL BYTES WRITTEN (LO ORDER)
;  0565  0             TOT_BYT_WRT_HI  = [17, 0, 16, 0],          ! TOTAL BYTES WRITTEN (HI ORDER)
;  0566  0             MTOT_BYT_WRT    = [18, 0, 16, 0],          ! TOTAL MEGABYTES WRITTEN
;  0567  0             ERR_HRD_SEK     = [19, 0, 8, 0],           ! TOTAL HARD ERRORS - SEEK
;  0568  0             ERR_HRD_DAT     = [19, 8, 8, 0],           ! TOTAL HARD ERRORS - DATA
;  0569  0             ERR_HRD_DRV     = [20, 0, 8, 0],           ! TOTAL HARD ERRORS - DRIVE
;  0570  0             ERR_HRD_HST     = [20, 8, 8, 0],           ! TOTAL HARD ERRORS - HOST
;  0571  0             ERR_SFT_SEK     = [21, 0, 8, 0],           ! TOTAL SOFT ERRORS - SEEK
;  0572  0             ERR_SFT_DAT     = [21, 8, 8, 0],           ! TOTAL SOFT ERRORS - DATA
;  0573  0             ERR_SFT_DRV     = [22, 0, 8, 0],           ! TOTAL SOFT ERRORS - DRIVE
;  0574  0             ERR_SFT_HST     = [22, 8, 8, 0],           ! TOTAL SOFT ERRORS - HOST
;  0575  0             T_BLK_WT        = [23, 0,16, 0],           !                         ZZZ
;  0576  0             T_DBN_WT        = [24, 0,16, 0],           !DBNS WRITTEN             ZZZ
;  0577  0             T_BLK_RD        = [25, 0,16, 0],           !                         ZZZ
;  0578  0             T_DBN_RD        = [26, 0,16, 0]            !DBNS READ                ZZZ
;  0579  0
;  0580  0             tes,
;  0581  0     !
;  0582  0     !****** CONTROLLER ERROR TALLY FIELDS
;  0583  0     !
;  0584  0     C_ERR_FIELDS =
;  0585  0             set
;  0586  0             C_ERR_HRD       = [0, 0, 8, 0],            ! HARD ERRORRS
;  0587  0             C_ERR_SFT       = [0, 8, 8, 0]             ! SOFT ERRORS
;  0588  0             tes,
;  0589  0     !
;  0590  0     !****** DRIVER CONTROLLER TABLE (DCT) FIELDS
;  0591  0     !
;  0592  0     DCT_FIELDS =
```

```
;  0593  0          set
;  0594  0          WORD0          = [0, 0, 16, 0],          ! ALL FIELDS IN WORD 0
;  0595  0          CRING_CNT      = [0, 0, 8, 0],           ! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
;  0596  0          IG_INT         = [0, 14, 1, 0],          ! IGNORE INTERRUPT BIT
;  0597  0          STAT           = [0, 15, 1, 0],          ! ONLINE / OFFLINE STATUS
;  0598  0          SA_SAVE        = [1, 0, 16, 0],          ! SA REGISTER SAVE WORD
;  0599  0          RR_BEG         = [2, 0, 16, 0],          ! FIXED ADDRESSES OF START AND
;  0600  0          RR_END         = [3, 0, 16, 0],          !       END OF EACH RING
;  0601  0          CR_BEG         = [4, 0, 16, 0],          !              :
;  0602  0          CR_END         = [5, 0, 16, 0],          !              V
;  0603  0          RR_POLL        = [6, 0, 16, 0],          ! ADDR OF NEXT RRING SLOT TO BE POLLED
;  0604  0          CR_POLL        = [7, 0, 16, 0],          ! ADDR OF NEXT CRING SLOT TO BE POLLED
;  0605  0          CR_NEXT        = [8, 0, 16, 0]           ! ADDR OF NEXT AVAIL CRING SLOT
;  0606  0          tes,
;  0607  0      !
;  0608  0      !****** ERROR LOG PACKET SAVE AREA FIELDS
;  0609  0      !
;  0610  0      EP_FIELDS =
;  0611  0          set
;  0612  0          EL_CNTR        = [0, 0, 8, 0],           ! CONTROLLER NUMBER
;  0613  0          EL_CONTENTS    = [0, 8, 8, 0],           ! FLAG INDICATES IF PACKET CONTENTS ALREADY PRINTED
;  0614  0          EL_MSGLEN      = [1, 0, 16, 0],          ! PACKET LENGTH
;  0615  0          EL_CRN_LO      = [3, 0, 16, 0],          ! COMMAND REFERENCE NUMBER
;  0616  0          EL_CRN_HI      = [4, 0, 16, 0],          !
;  0617  0          EL_DK_NUM      = [5, 0, 16, 0],          ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0618  0          EL_FORMAT      = [7, 0, 8, 0],           ! FORMAT
;  0619  0          EL_CONTINUE    = [7, 14, 1, 0],          ! CONTINUE FLAG
;  0620  0          EL_SUCCESS     = [7, 15, 1, 0],          ! SUCCESS FLAG
;  0621  0          EL_CODE        = [8, 0, 5, 0],           ! ERROR CODE
;  0622  0          EL_SUBCODE     = [8, 5, 11, 0],          ! SUB CODE
;  0623  0          EL_RETRY       = [20, 8, 8, 0],          ! RETRY COUNT
;  0624  0          EL_BLOCK       = [23, 0, 16, 0],         ! BLOCK NUMBER
;  0625  0          EL_BLOCK_TYPE  = [24, 12, 4, 0]          ! TYPE OF BLOCK NUMBER INFO RETURNED
;  0626  0          tes,
;  0627  0      !
;  0628  0      !****** INFORMATION ABOUT LAST RESPONSE PACKET
;  0629  0      !
;  0630  0      LAST_PKT_FIELDS =
;  0631  0          set
;  0632  0          LAST_HRD_ERR   = [0, 0, 16, 0],          ! FLAG INDICATES IF HARD ERROR OCCURED
;  0633  0          LAST_CRN_LO    = [1, 0, 16, 0],          ! COMMAND REFERENCE NUMBER
;  0634  0          LAST_CRN_HI    = [2, 0, 16, 0]           !
;  0635  0          tes,
;  0636  0      !
;  0637  0      !****** RDRX REGISTER FIELDS
;  0638  0      !
;  0639  0      RC_REG =
;  0640  0          set
;  0641  0          RC_ALL         = [0, 16, 0]              ! DEFINE ALL BITS
;  0642  0          tes;
```

```
;  0643  0     !*****************************************************************************
;  0644  0     !                                                                            *
;  0645  0     !                          M A C R O S                                       *
;  0646  0     !                                                                            *
;  0647  0     !*****************************************************************************
;  0648  0     !
;  0649  0     macro
;  0650  0     !
;  0651  0     !****** CST FIELDS. MODEL FOR WDS 3-12, 13-21, 22-30, AND 31-39.       ZZZ
;  0652  0     !
;  0653  0           D_ALL           = 0, 16, 0%,                          ! ALL FIELDS
;  0654  0           D_DISK_NUM      = 0, 4, 0%,                           ! DISK ADDRESS
;  0655  0           D_TYPE          = 4, 1, 0%,                           !DISK TYPE - 1 BIT       ZZZ
;  0656  0           D_UNIT          = 8, 4, 0%,                           ! DISK UNIT NUMBER (DRS UNIT)
;  0657  0           D_FATAL         = 12, 1, 0%,                          ! FATAL ERROR BIT
;  0658  0           D_STAT          = 13, 1, 0%,                          ! DISK STATUS BIT
;  0659  0           D_PRES          = 14, 1, 0%,                          ! DISK PRESENT BIT
;  0660  0           D_PROT          = 15, 1, 0%,                          ! DISK PROTECTION BIT
;  0661  0           D_BEG0          = 0, 16, 0%,          !BEGIN TRACK LO          ZZZ
;  0662  0           D_BEG1          = 0, 16, 0%,          !BEGIN TRACK HI          ZZZ
;  0663  0           D_END0          = 0, 16, 0%,          !END TRACK LO            ZZZ
;  0664  0           D_END1          = 0, 16, 0%,          !END TRACK HI            ZZZ
;  0665  0           D_NAME_0        = 0, 8, 0%,           ! NAME (FIRST  CHARACTER)
;  0666  0           D_NAME_1        = 8, 8, 0%,           ! NAME (SECOND CHARACTER)
;  0667  0           D_NAME_2        = 0, 8, 0%,           ! NAME (THIRD  CHARACTER)
;  0668  0           D_NAME_3        = 8, 8, 0%,           ! NAME (FOURTH CHARACTER)
;  0669  0     !     D_NUL           = 0, 16, 0%,          !NUL AFTER NAME          ZZZ
;  0670  0           D_DBN           = 0, 8, 0%,           !RELATIVE DBN            ZZZ
;  0671  0           DUPWRITE        = 12, 1, 0%,          !DUP WRITE FLAG          ZZZ
;  0672  0           D_ACTIVE        = 13, 1, 0%,          !ACTIVE STATE            ZZZ
;  0673  0           DUPERROR        = 14, 1, 0%,          !DUP ERROR FLAG          ZZZ
;  0674  0           NODUPMEDIA      = 15, 1, 0%,          !NO DUP MEDIA            ZZZ
;  0675  0           D_COUNT         = 0, 16, 0%,          !MSCP FUNCTION COUNTER   ZZZ
;  0676  0
;  0677  0
;  0678  0     !************ BST FIELDS **************                                  ZZZ
;  0679  0     !                                                                      ZZZ
;  0680  0           HI_WRD          = 1, 0, 16, 0%,       !HI LBN                  ZZZ
;  0681  0           LO_WRD          = 0, 0, 16, 0%,       !LO LBN                  ZZZ
;  0682  0     !
;  0683  0     !****** BIT TEST
;  0684  0     !                (CAUTION: THE FIRST ARGUMENT IS THE ADDRESS AND NOT THE CONTENTS)
;  0685  0     !
; M0686  0     BIT_TST (ADDR, EXPECTED) =
; M0687  0         (if (.ADDR and EXPECTED) eql EXPECTED
; M0688  0         then
; M0689  0             TRUE
; M0690  0         else
;  0691  0             FALSE )%,
;  0692  0     !
;  0693  0     !****** RDRX WRITE
;  0694  0     !
; M0695  0     WRT_RDRX (0, FIELDNAM, IMAGE) =
```

F6


```
; M 0696  0          begin
; M 0697  0          local
; M 0698  0              RC_REG;
; M 0699  0          RC_REG <%fieldexpand (FIELDNAM)> = IMAGE;
; M 0700  0          (.RDRX_ADDR + (%upval * 0)) = .RC_REG;
;   0701  0          end%;
```

```
;    0702  0     !*****************************************************************************
;    0703  0     !                                                                            *
;    0704  0     !                          S T R U C T U R E S                               *
;    0705  0     !                                                                            *
;    0706  0     !*****************************************************************************
;    0707  0     !
;    0708  0     !****** NIBBLE (4-BIT) VECTOR STRUCTURE
;    0709  0     !
;    0710  0     !structure
;    0711  0     !    NIBVECTOR [I; N] =
;    0712  0     !        [(N + 1) / 2]
;    0713  0     !        (NIBVECTOR + I / 2) <(I ^ 2) and 4, 4>;
;    0714  0     !
;    0715  0     !****** RDRX ACCESS ALGORITHM
;    0716  0     !
;    0717  0     structure
;    0718  0         RDRX [O, P, S, E] =
;    0719  1             begin
;    0720  1             local
;    0721  1                 RC_REG;
;    0722  1             RC_REG = .(RDRX + %upval * 0) <0, %bpval, 0>;
;    0723  1             RC_REG
;    0724  1             end
;    0725  0             <P, S, E>;
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAE0.REQ/LIST=ZRQAE0.LIS/LIBRARY=ZRQAE0.L16/SOURCE=PAGE:53

; Run Time:          00:06.5
; Elapsed Time:      00:11.9
; Lines/CPU Min:      6671
; Lexemes/CPU-Min: 35429
; Memory Used:  76 pages
; Library Precompilation Complete
ZRQAE0.EXE   Memory allocation map  TKB M40.02      Page 1
                    9-JUL-84   08:32


Partition name : DUMMY
Identification : V01.8
Task  UIC      : [202,24]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 17728. words
Task address limits: 002000 107163
R-W disk blk limits: 000002 000107 000106 00070.

*** Root segment: ZRQAE0


R/W mem  limits: 002000 107163 105164 35444.
Disk blk limits: 000002 000107 000106 00070.
```

Memory allocation synopsis:

Section	Title	Ident	File

```
Section                                          Title  Ident  File
-------                                          -----  -----  -----
. BLK.:(RW,I,LCL,REL,CON)      002000 000000 00000.                  H6
$CODE$:(RO,I,LCL,REL,CON)      002000 075104 31300.  ZRQAM3 V01.8  ZRQAE0.OB2;8
                               [illegible]  ZRQAM1 V01.8  ZRQAE0.OB1;8
$OWN$ :(RW,D,LCL,REL,CON)      [illegible]  ZRQAM2 V01.8  ZRQAE0.OB1;8
                               [illegible]  ZRQAM3 V01.8  ZRQAE0.OB2;8
$PLIT$:(RO,D,LCL,REL,CON)      [illegible]  ZRQAM4 V01.8  ZRQAE0.OB1;8
                               [illegible]  ZRQAM5 V01.8  ZRQAE0.OB2;8
                               [illegible]
$XYZ$ :(RO,I,LCL,REL,CON)      107034 000130 00088.  ZRQAM4 V01.8  ZRQAE0.OB2;8
```

Global symbols:

ADDR.V	105107-R	BIT06	000100	BIT3	000010	BOE	000400	CNTR.E	023544-R	DATAGM	072560-R	DBM15	005016-R
ADR	000020	BIT07	000200	BIT4	000020	BST	077264-R	CREDIT	105060-R	DBM101	006040-R	DBM18	005046-R
ASTERI	024646-R	BIT08	000400	BIT5	000040	BUFF.A	104744-R	CRLF	024634-R	DBM104	006066-R	DBM19	005120-R
AZINT	067144-R	BIT09	001000	BIT6	000100	BUFF.O	104764-R	CRN.HI	105056-R	DBM105	006130-R	DBM20	005204-R
AZINTO	067126-R	BIT1	000002	BIT7	000200	BYTS.P	105034-R	CRN.LO	105054-R	DBM107	006166-R	DBM21	005260-R
BIT0	000001	BIT10	002000	BIT8	000400	CCTLR	105014-R	CST	077104-R	DBM108	006224-R	DBM22	005342-R
BIT00	000001	BIT11	004000	BIT9	001000	CDISK	105016-R	CST.AD	077232-R	DBM109	006314-R	DBM23	005406-R
BIT01	000002	BIT12	010000	BL$DIV	076704-R	CER.01	016510-R	CTLR.C	105022-R	DBM111	006374-R	DBM25	005444-R
BIT02	000004	BIT13	020000	BL$LAS	107034-R	CER.02	016554-R	CTLR.I	043556-R	DBM112	006474-R	DBM26	005512-R
BIT03	000010	BIT14	040000	BL$MOD	076716-R	CLK.PR	105070-R	CUOFF	105020-R	DBM12	004742-R	DBM27	005544-R
BIT04	000020	BIT15	100000	BL$MUL	076460-R	CLK.TI	105066-R	C.ERR.	100706-R	DBM120	006576-R	DBM28A	005616-R
BIT05	000040	BIT2	000004	BL$SHF	076730-R	CMD.TI	105050-R	DASH	024640-R	DBM121	006670-R	DBM28B	005656-R

```
DBM29  005716-R  EGD.24 012110-R  EX.CBW 015554-R  GP$31  025466-R  LOE     040000    L$SPC  002056-R  RDRX.A 077260-R
DBM32  005764-R  EGH.30 012154-R  EX.CMD 014566-R  GP$32  025500-R  LOT     000010    L$SPCP 002020-R  RDRX.E 024264-R
DBM5   004714-R  EGS.01 011140-R  EX.CMP 014626-R  GP$4   025074-R  L$ACP   002110-R  L$SPTP 002024-R  REG.EX 044204-R
DCT    077234-R  EGS.02 011160-R  EX.LBN 015146-R  GP$5   025106-R  L$APT   002036-R  L$STA  002030-R  RETPKT 102436-R
DCT.AD 077256-R  EH.0   012632-R  EX.LBR 015242-R  GP$6   025116-R  L$AU    032316-R  L$SW   024706-R  ROUND. 065642-R
DFPTBL 024656-R  EH.1   012670-R  EX.LBW 015310-R  GP$7   025126-R  L$AUT   002070-R  L$SWLE 024704-R  RPS.RE 066624-R
DF.MSG 024302-R  EH.10  013244-R  EX.ONL 014642-R  GP$8   025136-R  L$AUTO  031354-R  L$TEST 002114-R  RPT1   007646-R
DIO.RE 062434-R  EH.12  013274-R  EX.OP  014666-R  GP$9   025150-R  L$CCP   002106-R  L$TIML 002014-R  RPT10  010356-R
DISK.R 070446-R  EH.13  013330-R  EX.PBN 015204-R  HARD.E 063460-R  L$CLEA  031572-R  L$UNIT 002012-R  RPT11  010444-R
DRIVER 043224-R  EH.2   012726-R  EX.RBN 015356-R  HARD.I 045144-R  L$CO    002032-R  MD.INI 052072-R  RPT12  010512-R
DROP.C 033770-R  EH.3   012766-R  EX.RD  014606-R  HOE    100000    L$DEPO  002011-R  MINUTE 105064-R  RPT13  010560-R
DRV.CT 034076-R  EH.4   013024-R  EX.RP  016140-R  HOE.FL 105071-R  L$DESC  025016-R  MODULA 034630-R  RPT14  010660-R
DR.ERR 050622-R  EH.5   013050-R  EX.SA  014472-R  HOST.W 066206-R  L$DESP  002076-R  MSCP.P 100710-R  RPT15  010756-R
DR.RET 067012-R  EH.6   013100-R  EX.SB  014544-R  HOURS  105063-R  L$DEVP  002060-R  MSG.01 007526-R  RPT16  011056-R
DUP    055600-R  EH.7   013130-R  EX.SBO 014540-R  HRD.MS 024400-R  L$DISP  002124-R  MSG.02 007560-R  RPT2   007732-R
DUPCOM 060254-R  EH.8   013160-R  EX.SC  014510-R  HRD.SU 024576-R  L$DLY   002116-R  MSG.03 007614-R  RPT3   007776-R
DUPIDL 060422-R  EH.9   013214-R  EX.TIM 016236-R  HWPTE0 024672-R  L$DTP   002040-R  MULTI. 051416-R  RPT4   010062-R
DUPPKT 077636-R  ELG.FM 014460-R  EX.WRD 016226-R  HWPTE1 024674-R  L$DTYP  002034-R  NAME.H 024700-R  RPT5   010126-R
DUPRED 057212-R  ELG.00 014126-R  EX.WRT 014616-R  HWPTS0 024666-R  L$DU    032224-R  NAME.L 024676-R  RPT6   010214-R
DUPROU 024722-R  ELOG.P 103212-R  FATAL. 067636-R  HWPTS1 024670-R  L$DUT   002072-R  NEX    105052-R  RPT7   010260-R
DUPWRT 056134-R  EMS.BL 036524-R  FER.BC 105104-R  HWPT.B 024662-R  L$DVTY  024776-R  NEXT.P 105062-R  RPT8   010276-R
DUP.CO 062766-R  EMS.CM 040554-R  FER.LB 105102-R  HWPT.D 024664-R  L$EF    002052-R  NEX.TR 032326-R  RPT9   010324-R
DUP.FL 105012-R  EMS.DB 036326-R  FILL.B 062006-R  HWPT.I 024656-R  L$ENVI  002044-R  NULL   004712-R  RP.ADD 103210-R
DUP.RS 067344-R  EMS.EL 037432-R  FORCED 105100-R  HWPT.V 024660-R  L$ERRT  002126-R  OF.RC  105044-R  RP.IND 103206-R
DUR    105024-R  EMS.ER 041102-R  FREE.M 105032-R  HWQ1   002136-R  L$ETP   002102-R  OUT.IO 033644-R  RP.USE 103176-R
DU.MSG 006760-R  EMS.RP 037374-R  FSET.U 063334-R  HWQ10  002734-R  L$EXP1  002046-R  OVF.CH 065566-R  SA.REG 105046-R
DU.RSN 007500-R  EMS.01 041306-R  GET.IO 033464-R  HWQ11  002764-R  L$EXP4  002064-R  PKT.US 102422-R  SB.COD 105040-R
D$PCNT 002122-R  EMS.10 041344-R  GET.PK 032614-R  HWQ2   002152-R  L$EXP5  002066-R  PNT    001000    SCAN.E 071712-R
EBD.10 012242-R  EMS.12 041406-R  GET.RA 053204-R  HWQ3   002162-R  L$HARD  025042-R  POLL.C 070070-R  SC.CLK 021642-R
EBD.12 012302-R  EMS.13 041444-R  GET.RE 033342-R  HWQ4   002226-R  L$HIME  002120-R  POLL.R 070170-R  SC.CON 016654-R
EBD.13 012350-R  EMS.14 041506-R  GP$DIS 025404-R  HWQ5   002244-R  L$HPCP  002016-R  PRI    002000    SC.CTO 021104-R
EBD.14 012402-R  EMS.18 041550-R  GP$1   025042-R  HWQ6A  002314-R  L$HPTP  002022-R  PRI00  000000    SC.DIS 017202-R
EBD.18 012442-R  EMS.21 041612-R  GP$10  025162-R  HWQ6B  002366-R  L$HRDL  025040-R  PRI01  000040    SC.DST 017612-R
EBD.19 012476-R  EMS.22 041630-R  GP$11  025176-R  HWQ7A  002442-R  L$HW    024656-R  PRI02  000100    SC.DS2 017666-R
EBD.24 012556-R  EMS.24 041646-R  GP$12  025212-R  HWQ7B  002512-R  L$HWLE  024654-R  PRI03  000140    SC.DUP 016676-R
EBS.01 012200-R  EMS.30 041710-R  GP$13  025222-R  HWQ8   002562-R  L$ICP   002104-R  PRI04  000200    SC.ECC 017740-R
EF.CON 000036    ENTRY. 105010-R  GP$14  025236-R  HWQ9   002634-R  L$INIT  031342-R  PRI05  000240    SC.ECD 020022-R
EF.NEW 000035    EOP.FL 105011-R  GP$15  025250-R  IBE    010000    L$LADP  002026-R  PRI06  000300    SC.EC1 020272-R
EF.PWR 000034    ERRBLK 002134-R  GP$16  025262-R  IDU    000040    L$LAST  107040-R  PRI07  000340    SC.EC2 020322-R
EF.RES 000037    ERRMSG 002132-R  GP$17  025274-R  IER    020000    L$LOAD  002100-R  PROC.R 062240-R  SC.EC3 020352-R
EF.STA 000040    ERRNBR 002130-R  GP$18  025306-R  INIT.I 052232-R  L$LUN   002074-R  PUTA.B 033604-R  SC.EC4 020404-R
EGD.10 011252-R  ERRTYP 002126-R  GP$19  025314-R  INIT.O 105106-R  L$MREV  002050-R  PUT.IO 033540-R  SC.EC5 020434-R
EGD.11 011312-R  ERR.CO 014072-R  GP$2   025052-R  INIT.T 043066-R  L$NAME  002000-R  PUT.PK 033202-R  SC.EC6 020464-R
EGD.12 011336-R  ERR.00 013402-R  GP$20  025322-R  INI.CT 044060-R  L$NDHR  025232-R  PUT.RE 033450-R  SC.EC7 020514-R
EGD.13 011364-R  EVL    000004    GP$21  025330-R  INI.RR 046014-R  L$NDHW  024702-R  P.INDE 105076-R  SC.EC8 020546-R
EGD.14 011412-R  EX.ACC 014654-R  GP$22  025336-R  INT.GE 044742-R  L$NDSF  025516-R  QIO    105030-R  SC.EC9 020600-R
EGD.15 011442-R  EX.BB  014672-R  GP$23  025344-R  IN.IOD 033702-R  L$NDSW  024766-R  QIO.FU 054720-R  SC.EDC 021234-R
EGD.16 011460-R  EX.BBU 015056-R  GP$24  025352-R  IODQ   104774-R  L$PRIO  002042-R  QIO.GE 052604-R  SC.FCT 020224-R
EGD.17 011510-R  EX.BB1 014762-R  GP$25  025366-R  IODQ.I 105004-R  L$PROT  024770-R  QIO.LB 060646-R  SC.FER 017274-R
EGD.18 011526-R  EX.BC  015626-R  GP$26  025376-R  IODQ.O 105006-R  L$PRT   002112-R  QIO.OK 052436-R  SC.FE2 017362-R
EGD.19 011546-R  EX.BD  015702-R  GP$27  025410-R  IO.RET 063064-R  L$REPP  002062-R  QIO.OU 052532-R  SC.FUL 020074-R
EGD.20 011606-R  EX.BDR 015760-R  GP$28  025424-R  IPKT.A 102420-R  L$REV   002010-R  QIO.SI 061576-R  SC.HWP 020702-R
EGD.21 011674-R  EX.BDW 016050-R  GP$29  025434-R  IRDRX. 077262-R  L$RPT   026574-R  QIO.UN 053314-R  SC.IDS 021274-R
EGD.22 012006-R  EX.CBC 015436-R  GP$3   025062-R  ISR    000100    L$SFTL  025234-R  RANDOM 100646-R  SC.IOP 017130-R
EGD.23 012046-R  EX.CBR 015502-R  GP$30  025450-R  IXE    004000    L$SOFT  025236-R  RDM.CN 100644-R  SC.ISH 017452-R
```

J6

SEQ 0486

```
ZRQAE0.EXE   Memory allocation map  TKB M40.02      Page 3
ZRQAE0                9-JUL-84   08:32


SC.IS2 017532-R  SC.SDS 021156-R  SET.CT 046116-R  SWP.FL 024712-R  SWQ15  003616-R  SWQ9   003216-R  VEC.BR 044416-R
SC.NXM 021014-R  SC.SON 016750-R  SET.UP 032514-R  SWP.RA 024716-R  SWQ17  003666-R  S.DUPP 105074-R  WAIT   034612-R
SC.ODA 020742-R  SC.SRI 021436-R  SFPTBL 024706-R  SWP.TI 024720-R  SWQ19  003764-R  S.PATT 105072-R  XX13   016276-R
SC.ODB 020772-R  SC.SRT 021344-R  SFT.MS 024476-R  SWP.UC 024724-R  SWQ2   003040-R  TALLY  077304-R  XX23   016320-R
SC.ONL 016726-R  SC.SUR 021736-R  SOFT.E 073504-R  SWP.UD 024726-R  SWQ20  004054-R  TIME   032336-R  XX32   016354-R
SC.PAR 021050-R  SC.SWP 020642-R  SPACE4 024630-R  SWP.XF 024710-R  SWQ21  004152-R  TRK.SG 100640-R  XX33   016402-R
SC.POE 021524-R  SC.UNK 016770-R  STEP   105042-R  SWQ1   003016-R  SWQ22  004254-R  T$FREE 107160-R  XX34   016440-R
SC.PSP 021766-R  SC.VOL 017050-R  ST.COD 105036-R  SWQ10  003272-R  SWQ23  004332-R  T$PTHV 000004    $END.L 107162-R
SC.RCT 020054-R  SC.576 020150-R  SWEEP  066472-R  SWQ11  003336-R  SWQ24  004424-R  T.ADDR 077634-R  $SAVE2 076776-R
SC.RDY 021560-R  SEND   034154-R  SWM1   004622-R  SWQ12  003370-R  SWQ25  004522-R  T1     043052-R  $SAVE3 077012-R
SC.RSP 021670-R  SEQUEN 070546-R  SWP.DP 024714-R  SWQ13  003466-R  SWQ4   003122-R  UAM    000200    $SAVE4 077030-R
SC.SDI 016630-R  SET.CP 032420-R  SWP.ER 024706-R  SWQ14  003544-R  SWQ7   003144-R  UPD.IO 065246-R  $SAVE5 077050-R


*** Task builder statistics:

    Total work file references: 153058.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 5616. words (21. pages)
    Size of work file: 4864. words (19. pages)

    Elapsed time:00:00:36
```

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
ADDR.V	105107-R	# ZRQAM1	ZRQAM2	ZRQAM3
ADR	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
ASTERI	024646-R	# ZRQAM1	ZRQAM2	
AZINT	067144-R	# ZRQAM3		
AZINTO	067126-R	# ZRQAM3		
BIT0	000001	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT00	000001	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT01	000002	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT02	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT03	000010	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT04	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT05	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT06	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT07	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT08	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT09	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT1	000002	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT10	002000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT11	004000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT12	010000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT13	020000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT14	040000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT15	100000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT2	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT3	000010	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT4	000020	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT5	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT6	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT7	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT8	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BIT9	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
BL$DIV	076704-R	# B16MUL	ZRQAM2	ZRQAM3
BL$LAS	107034-R	# ZRQAM4		
BL$MOD	076716-R	# B16MUL	ZRQAM2	ZRQAM3
BL$MUL	076460-R	# B16MUL	ZRQAM2	ZRQAM3
BL$SHF	076730-R	# B16MUL	ZRQAM3	
BOE	000400	# ZRQAM1	# ZRQAM2	# ZRQAM3
BST	077264-R	# ZRQAM1	ZRQAM2	ZRQAM3
BUFF.A	104744-R	# ZRQAM1	ZRQAM2	ZRQAM3
BUFF.O	104764-R	# ZRQAM1	ZRQAM2	ZRQAM3
BYTS.P	105034-R	# ZRQAM1	ZRQAM2	ZRQAM3
CCTLR	105014-R	# ZRQAM1	ZRQAM2	ZRQAM3
CDISK	105016-R	# ZRQAM1	ZRQAM2	ZRQAM3
CER.01	016510-R	# ZRQAM1	ZRQAM2	
CER.02	016554-R	# ZRQAM1	ZRQAM2	
CLK.PR	105070-R	# ZRQAM1	ZRQAM2	ZRQAM3
CLK.TI	105066-R	# ZRQAM1	ZRQAM2	ZRQAM3
CMD.TI	105050-R	# ZRQAM1	ZRQAM2	ZRQAM3
CNTR.E	023544-R	# ZRQAM1	ZRQAM2	
CREDIT	105060-R	# ZRQAM1	ZRQAM2	ZRQAM3
CRLF	024634-R	# ZRQAM1	ZRQAM2	ZRQAM3
CRN.HI	105056-R	# ZRQAM1	ZRQAM2	ZRQAM3

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
CRN.LO	105054-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST	077104-R	# ZRQAM1	ZRQAM2	ZRQAM3
CST.AD	077232-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.C	105022-R	# ZRQAM1	ZRQAM2	ZRQAM3
CTLR.I	043556-R	# ZRQAM3		
CUOFF	105020-R	# ZRQAM1	ZRQAM2	ZRQAM3
C.ERR.	100706-R	# ZRQAM1	ZRQAM2	ZRQAM3
DASH	024640-R	# ZRQAM1	ZRQAM2	
DATAGM	072560-R	# ZRQAM3		
DBM101	006040-R	# ZRQAM1		
DBM104	006066-R	# ZRQAM1		
DBM105	006130-R	# ZRQAM1		
DBM107	006166-R	# ZRQAM1	ZRQAM2	
DBM108	006224-R	# ZRQAM1	ZRQAM3	
DBM109	006314-R	# ZRQAM1	ZRQAM3	
DBM111	006374-R	# ZRQAM1	ZRQAM3	
DBM112	006474-R	# ZRQAM1	ZRQAM3	
DBM12	004742-R	# ZRQAM1	ZRQAM3	
DBM120	006576-R	# ZRQAM1	ZRQAM3	
DBM121	006670-R	# ZRQAM1	ZRQAM3	
DBM15	005016-R	# ZRQAM1		
DBM18	005046-R	# ZRQAM1	ZRQAM3	
DBM19	005120-R	# ZRQAM1	ZRQAM3	
DBM20	005204-R	# ZRQAM1	ZRQAM3	
DBM21	005260-R	# ZRQAM1	ZRQAM3	
DBM22	005342-R	# ZRQAM1	ZRQAM3	
DBM23	005406-R	# ZRQAM1	ZRQAM3	
DBM25	005444-R	# ZRQAM1	ZRQAM3	
DBM26	005512-R	# ZRQAM1	ZRQAM3	
DBM27	005544-R	# ZRQAM1	ZRQAM3	
DBM28A	005616-R	# ZRQAM1		
DBM28B	005656-R	# ZRQAM1		
DBM29	005716-R	# ZRQAM1	ZRQAM3	
DBM32	005764-R	# ZRQAM1		
DBM5	004714-R	# ZRQAM1	ZRQAM2	
DCT	077234-R	# ZRQAM1	ZRQAM2	ZRQAM3
DCT.AD	077256-R	# ZRQAM1	ZRQAM2	ZRQAM3
DFPTBL	024656-R	# ZRQAM1		
DF.MSG	024302-R	# ZRQAM1	ZRQAM3	
DIO.RE	062434-R	# ZRQAM3		
DISK.R	070446-R	# ZRQAM3		
DRIVER	043224-R	# ZRQAM3		
DROP.C	033770-R	# ZRQAM2	ZRQAM3	
DRV.CT	034076-R	# ZRQAM2	ZRQAM3	
DR.ERR	050622-R	# ZRQAM3		
DR.RET	067012-R	# ZRQAM3		
DUP	055600-R	# ZRQAM3		
DUPCOM	060254-R	# ZRQAM3		
DUPIDL	060422-R	# ZRQAM3		
DUPPKT	077636-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUPRED	057212-R	# ZRQAM3		
DUPROU	024722-R	# ZRQAM1	ZRQAM3	

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
DUPWRT	056134-R	# ZRQAM3		
DUP.CO	062766-R	# ZRQAM3		
DUP.FL	105012-R	# ZRQAM1	ZRQAM2	ZRQAM3
DUP.RS	067344-R	# ZRQAM3		
DUR	105024-R	# ZRQAM1	ZRQAM2	ZRQAM3
DU.MSG	006760-R	# ZRQAM1	ZRQAM2	
DU.RSN	007500-R	# ZRQAM1	ZRQAM2	
D$PCNT	002122-R	# ZRQAM1		
EBD.10	012242-R	# ZRQAM1	ZRQAM2	
EBD.12	012302-R	# ZRQAM1	ZRQAM2	
EBD.13	012350-R	# ZRQAM1	ZRQAM2	
EBD.14	012402-R	# ZRQAM1	ZRQAM2	
EBD.18	012442-R	# ZRQAM1	ZRQAM2	
EBD.19	012476-R	# ZRQAM1	ZRQAM2	
EBD.24	012556-R	# ZRQAM1	ZRQAM2	
EBS.01	012200-R	# ZRQAM1	ZRQAM2	
EF.CON	000036	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.NEW	000035	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.PWR	000034	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.RES	000037	# ZRQAM1	# ZRQAM2	# ZRQAM3
EF.STA	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
EGD.10	011252-R	# ZRQAM1	ZRQAM3	
EGD.11	011312-R	# ZRQAM1	ZRQAM3	
EGD.12	011336-R	# ZRQAM1	ZRQAM3	
EGD.13	011364-R	# ZRQAM1	ZRQAM3	
EGD.14	011412-R	# ZRQAM1	ZRQAM3	
EGD.15	011442-R	# ZRQAM1	ZRQAM3	
EGD.16	011460-R	# ZRQAM1	ZRQAM3	
EGD.17	011510-R	# ZRQAM1	ZRQAM3	
EGD.18	011526-R	# ZRQAM1	ZRQAM3	
EGD.19	011546-R	# ZRQAM1	ZRQAM3	
EGD.20	011606-R	# ZRQAM1	ZRQAM3	
EGD.21	011674-R	# ZRQAM1	ZRQAM3	
EGD.22	012006-R	# ZRQAM1	ZRQAM3	
EGD.23	012046-R	# ZRQAM1	ZRQAM3	
EGD.24	012110-R	# ZRQAM1	ZRQAM3	
EGH.30	012154-R	# ZRQAM1	ZRQAM3	
EGS.01	011140-R	# ZRQAM1	ZRQAM2	
EGS.02	011160-R	# ZRQAM1	ZRQAM3	
EH.0	012632-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.1	012670-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.10	013244-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.12	013274-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.13	013330-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.2	012726-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.3	012766-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.4	013024-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.5	013050-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.6	013100-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.7	013130-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.8	013160-R	# ZRQAM1	ZRQAM2	ZRQAM3
EH.9	013214-R	# ZRQAM1	ZRQAM2	ZRQAM3

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
ELG.FM	014460-R	# ZRQAM1	ZRQAM2	
ELG.OO	014126-R	# ZRQAM1	ZRQAM2	
ELOG.P	103212-R	# ZRQAM1	ZRQAM2	ZRQAM3
EMS.BL	036524-R	# ZRQAM2		
EMS.CM	040554-R	# ZRQAM2	ZRQAM3	
EMS.DB	036326-R	# ZRQAM2		
EMS.EL	037432-R	# ZRQAM2	ZRQAM3	
EMS.ER	041102-R	# ZRQAM2	ZRQAM3	
EMS.RP	037374-R	# ZRQAM2	ZRQAM3	
EMS.01	041306-R	# ZRQAM2		
EMS.10	041344-R	# ZRQAM2	ZRQAM3	
EMS.12	041406-R	# ZRQAM2	ZRQAM3	
EMS.13	041444-R	# ZRQAM2	ZRQAM3	
EMS.14	041506-R	# ZRQAM2	ZRQAM3	
EMS.18	041550-R	# ZRQAM2	ZRQAM3	
EMS.21	041612-R	# ZRQAM2	ZRQAM3	
EMS.22	041630-R	# ZRQAM2	ZRQAM3	
EMS.24	041646-R	# ZRQAM2	ZRQAM3	
EMS.30	041710-R	# ZRQAM2	ZRQAM3	
ENTRY.	105010-R	# ZRQAM1	ZRQAM2	ZRQAM3
EOP.FL	105011-R	# ZRQAM1	ZRQAM2	ZRQAM3
ERRBLK	002134-R	# ZRQAM1		
ERRMSG	002132-R	# ZRQAM1		
ERRNBR	002130-R	# ZRQAM1		
ERRTYP	002126-R	# ZRQAM1		
ERR.CO	014072-R	# ZRQAM1	ZRQAM2	
ERR.OO	013402-R	# ZRQAM1	ZRQAM2	
EVL	000004	# ZRQAM1	# ZRQAM2	# ZRQAM3
EX.ACC	014654-R	# ZRQAM1	ZRQAM2	
EX.BB	014672-R	# ZRQAM1	ZRQAM2	
EX.BBU	015056-R	# ZRQAM1	ZRQAM2	
EX.BB1	014762-R	# ZRQAM1	ZRQAM2	
EX.BC	015626-R	# ZRQAM1	ZRQAM2	
EX.BD	015702-R	# ZRQAM1	ZRQAM2	
EX.BDR	015760-R	# ZRQAM1	ZRQAM2	
EX.BDW	016050-R	# ZRQAM1	ZRQAM2	
EX.CBC	015436-R	# ZRQAM1	ZRQAM2	
EX.CBR	015502-R	# ZRQAM1	ZRQAM2	
EX.CBW	015554-R	# ZRQAM1	ZRQAM2	
EX.CMD	014566-R	# ZRQAM1	ZRQAM2	
EX.CMP	014626-R	# ZRQAM1	ZRQAM2	
EX.LBN	015146-R	# ZRQAM1	ZRQAM2	
EX.LBR	015242-R	# ZRQAM1	ZRQAM2	
EX.LBW	015310-R	# ZRQAM1	ZRQAM2	
EX.ONL	014642-R	# ZRQAM1	ZRQAM2	
EX.OP	014666-R	# ZRQAM1	ZRQAM2	
EX.PBN	015204-R	# ZRQAM1	ZRQAM2	
EX.RBN	015356-R	# ZRQAM1	ZRQAM2	
EX.RD	014606-R	# ZRQAM1	ZRQAM2	
EX.RP	016140-R	# ZRQAM1	ZRQAM2	
EX.SA	014472-R	# ZRQAM1	ZRQAM2	
EX.SB	014544-R	# ZRQAM1	ZRQAM2	

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
EX.SBO	014540-R	# ZRQAM1	ZRQAM2	
EX.SC	014510-R	# ZRQAM1	ZRQAM2	
EX.TIM	016236-R	# ZRQAM1	ZRQAM2	
EX.WRD	016226-R	# ZRQAM1	ZRQAM2	
EX.WRT	014616-R	# ZRQAM1	ZRQAM2	
FATAL.	067636-R	# ZRQAM3		
FER.BC	105104-R	# ZRQAM1	ZRQAM2	ZRQAM3
FER.LB	105102-R	# ZRQAM1	ZRQAM2	ZRQAM3
FILL.B	062006-R	# ZRQAM3		
FORCED	105100-R	# ZRQAM1	ZRQAM2	ZRQAM3
FREE.M	105032-R	# ZRQAM1	ZRQAM2	ZRQAM3
FSET.U	063334-R	# ZRQAM3		
GET.IO	033464-R	# ZRQAM2	ZRQAM3	
GET.PK	032614-R	# ZRQAM2	ZRQAM3	
GET.RA	053204-R	# ZRQAM3		
GET.RE	033342-R	# ZRQAM2	ZRQAM3	
GP$DIS	025404-R	# ZRQAM2		
GP$1	025042-R	# ZRQAM2		
GP$10	025162-R	# ZRQAM2		
GP$11	025176-R	# ZRQAM2		
GP$12	025212-R	# ZRQAM2		
GP$13	025222-R	# ZRQAM2		
GP$14	025236-R	# ZRQAM2		
GP$15	025250-R	# ZRQAM2		
GP$16	025262-R	# ZRQAM2		
GP$17	025274-R	# ZRQAM2		
GP$18	025306-R	# ZRQAM2		
GP$19	025314-R	# ZRQAM2		
GP$2	025052-R	# ZRQAM2		
GP$20	025322-R	# ZRQAM2		
GP$21	025330-R	# ZRQAM2		
GP$22	025336-R	# ZRQAM2		
GP$23	025344-R	# ZRQAM2		
GP$24	025352-R	# ZRQAM2		
GP$25	025366-R	# ZRQAM2		
GP$26	025376-R	# ZRQAM2		
GP$27	025410-R	# ZRQAM2		
GP$28	025424-R	# ZRQAM2		
GP$29	025434-R	# ZRQAM2		
GP$3	025062-R	# ZRQAM2		
GP$30	025450-R	# ZRQAM2		
GP$31	025466-R	# ZRQAM2		
GP$32	025500-R	# ZRQAM2		
GP$4	025074-R	# ZRQAM2		
GP$5	025106-R	# ZRQAM2		
GP$6	025116-R	# ZRQAM2		
GP$7	025126-R	# ZRQAM2		
GP$8	025136-R	# ZRQAM2		
GP$9	025150-R	# ZRQAM2		
HARD.E	063460-R	# ZRQAM3		
HARD.I	045144-R	# ZRQAM3		
HOE	100000	# ZRQAM1	# ZRQAM2	# ZRQAM3

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
HOE.FL	105071-R	# ZRQAM1	ZRQAM2	ZRQAM3
HOST.W	066206-R	# ZRQAM3		
HOURS	105063-R	# ZRQAM1	ZRQAM2	ZRQAM3
HRD.MS	024400-R	# ZRQAM1	ZRQAM3	
HRD.SU	024576-R	# ZRQAM1	ZRQAM3	
HWPTE0	024672-R	# ZRQAM1		
HWPTE1	024674-R	# ZRQAM1		
HWPTS0	024666-R	# ZRQAM1		
HWPTS1	024670-R	# ZRQAM1		
HWPT.B	024662-R	# ZRQAM1		
HWPT.D	024664-R	# ZRQAM1		
HWPT.I	024656-R	# ZRQAM1		
HWPT.V	024660-R	# ZRQAM1		
HWQ1	002136-R	# ZRQAM1	ZRQAM2	
HWQ10	002734-R	# ZRQAM1	ZRQAM2	
HWQ11	002764-R	# ZRQAM1	ZRQAM2	
HWQ2	002152-R	# ZRQAM1	ZRQAM2	
HWQ3	002162-R	# ZRQAM1	ZRQAM2	
HWQ4	002226-R	# ZRQAM1	ZRQAM2	
HWQ5	002244-R	# ZRQAM1	ZRQAM2	
HWQ6A	002314-R	# ZRQAM1	ZRQAM2	
HWQ6B	002366-R	# ZRQAM1	ZRQAM2	
HWQ7A	002442-R	# ZRQAM1	ZRQAM2	
HWQ7B	002512-R	# ZRQAM1	ZRQAM2	
HWQ8	002562-R	# ZRQAM1	ZRQAM2	
HWQ9	002634-R	# ZRQAM1	ZRQAM2	
IBE	010000	# ZRQAM1	# ZRQAM2	# ZRQAM3
IDU	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
IER	020000	# ZRQAM1	# ZRQAM2	# ZRQAM3
INIT.I	052232-R	# ZRQAM3		
INIT.O	105106-R	# ZRQAM1	ZRQAM2	ZRQAM3
INIT.T	043066-R	# ZRQAM3		
INI.CT	044060-R	# ZRQAM3		
INI.RR	046014-R	# ZRQAM3		
INT.GE	044742-R	# ZRQAM3		
IN.IOD	033702-R	# ZRQAM2	ZRQAM3	
IODQ	104774-R	# ZRQAM1	ZRQAM2	ZRQAM3
IODQ.I	105004-R	# ZRQAM1	ZRQAM2	ZRQAM3
IODQ.O	105006-R	# ZRQAM1	ZRQAM2	ZRQAM3
IO.RET	063064-R	# ZRQAM3		
IPKT.A	102420-R	# ZRQAM1	ZRQAM2	ZRQAM3
IRDRX.	077262-R	# ZRQAM1	ZRQAM2	ZRQAM3
ISR	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
IXE	004000	# ZRQAM1	# ZRQAM2	# ZRQAM3
LOE	040000	# ZRQAM1	# ZRQAM2	# ZRQAM3
LOT	000010	# ZRQAM1	# ZRQAM2	# ZRQAM3
L$ACP	002110-R	# ZRQAM1		
L$APT	002036-R	# ZRQAM1		
L$AU	032316-R	ZRQAM1	# ZRQAM2	
L$AUT	002070-R	# ZRQAM1		
L$AUTO	031354-R	ZRQAM1	# ZRQAM2	
L$CCP	002106-R	# ZRQAM1		

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
L$CLEA	031572-R	ZRQAM1	# ZRQAM2	
L$CO	002032-R	# ZRQAM1		
L$DEPO	002011-R	# ZRQAM1		
L$DESC	025016-R	ZRQAM1	# ZRQAM2	
L$DESP	002076-R	# ZRQAM1		
L$DEVP	002060-R	# ZRQAM1		
L$DISP	002124-R	# ZRQAM1		
L$DLY	002116-R	# ZRQAM1	ZRQAM2	ZRQAM3
L$DTP	002040-R	# ZRQAM1		
L$DTYP	002034-R	# ZRQAM1		
L$DU	032224-R	ZRQAM1	# ZRQAM2	
L$DUT	002072-R	# ZRQAM1		
L$DVTY	024776-R	ZRQAM1	# ZRQAM2	
L$EF	002052-R	# ZRQAM1		
L$ENVI	002044-R	# ZRQAM1		
L$ERRT	002126-R	# ZRQAM1		
L$ETP	002102-R	# ZRQAM1		
L$EXP1	002046-R	# ZRQAM1		
L$EXP4	002064-R	# ZRQAM1		
L$EXP5	002066-R	# ZRQAM1		
L$HARD	025042-R	ZRQAM1	# ZRQAM2	
L$HIME	002120-R	# ZRQAM1	ZRQAM2	
L$HPCP	002016-R	# ZRQAM1		
L$HPTP	002022-R	# ZRQAM1		
L$HRDL	025040-R	# ZRQAM2		
L$HW	024656-R	# ZRQAM1		
L$HWLE	024654-R	# ZRQAM1		
L$ICP	002104-R	# ZRQAM1		
L$INIT	031342-R	ZRQAM1	# ZRQAM2	
L$LADP	002026-R	# ZRQAM1		
L$LAST	107040-R	ZRQAM1	# ZRQAM4	
L$LOAD	002100-R	# ZRQAM1		
L$LUN	002074-R	# ZRQAM1	ZRQAM2	ZRQAM3
L$MREV	002050-R	# ZRQAM1		
L$NAME	002000-R	# ZRQAM1		
L$NDHR	025232-R	# ZRQAM2		
L$NDHW	024702-R	# ZRQAM1		
L$NDSF	025516-R	# ZRQAM2		
L$NDSW	024766-R	# ZRQAM1		
L$PRIO	002042-R	# ZRQAM1		
L$PROT	024770-R	# ZRQAM1		
L$PRT	002112-R	# ZRQAM1		
L$REPP	002062-R	# ZRQAM1		
L$REV	002010-R	# ZRQAM1		
L$RPT	026574-R	ZRQAM1	# ZRQAM2	
L$SFTL	025234-R	# ZRQAM2		
L$SOFT	025236-R	ZRQAM1	# ZRQAM2	
L$SPC	002056-R	# ZRQAM1		
L$SPCP	002020-R	# ZRQAM1		
L$SPTP	002024-R	# ZRQAM1		
L$STA	002030-R	# ZRQAM1		
L$SW	024706-R	# ZRQAM1		

GLOBAL CROSS REFERENCE

SYMBOL	VALUE	REFERENCES...		
L$SWLE	024704-R	# ZRQAM1		
L$TEST	002114-R	# ZRQAM1		
L$T1ML	002014-R	# ZRQAM1		
L$UNIT	002012-R	# ZRQAM1	ZRQAM2	ZRQAM3
MD.INI	052072-R	# ZRQAM3		
MINUTE	105064-R	# ZRQAM1	ZRQAM2	ZRQAM3
MODULA	034630-R	# ZRQAM2	ZRQAM3	
MSCP.P	100710-R	# ZRQAM1	ZRQAM2	ZRQAM3
MSG.01	007526-R	# ZRQAM1	ZRQAM2	
MSG.02	007560-R	# ZRQAM1	ZRQAM3	
MSG.03	007614-R	# ZRQAM1	ZRQAM3	
MULTI.	051416-R	# ZRQAM3		
NAME.H	024700-R	# ZRQAM1		
NAME.L	024676-R	# ZRQAM1		
NEX	105052-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEXT.P	105062-R	# ZRQAM1	ZRQAM2	ZRQAM3
NEX.TR	032326-R	# ZRQAM2	ZRQAM3	
NULL	004712-R	# ZRQAM1	ZRQAM2	
OF.RC	105044-R	# ZRQAM1	ZRQAM2	ZRQAM3
OUT.IO	033644-R	# ZRQAM2	ZRQAM3	
OVF.CH	065566-R	# ZRQAM3		
PKT.US	102422-R	# ZRQAM1	ZRQAM2	ZRQAM3
PNT	001000	# ZRQAM1	# ZRQAM2	# ZRQAM3
POLL.C	070070-R	# ZRQAM3		
POLL.R	070170-R	# ZRQAM3		
PRI	002000	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI00	000000	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI01	000040	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI02	000100	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI03	000140	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI04	000200	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI05	000240	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI06	000300	# ZRQAM1	# ZRQAM2	# ZRQAM3
PRI07	000340	# ZRQAM1	# ZRQAM2	# ZRQAM3
PROC.R	062240-R	# ZRQAM3		
PUTA.B	033604-R	# ZRQAM2	ZRQAM3	
PUT.IO	033540-R	# ZRQAM2	ZRQAM3	
PUT.PK	033202-R	# ZRQAM2	ZRQAM3	
PUT.RE	033450-R	# ZRQAM2	ZRQAM3	
P.INDE	105076-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO	105030-R	# ZRQAM1	ZRQAM2	ZRQAM3
QIO.FU	054720-R	# ZRQAM3		
QIO.GE	052604-R	# ZRQAM3		
QIO.LB	060646-R	# ZRQAM3		
QIO.OK	052436-R	# ZRQAM3		
QIO.OU	052532-R	# ZRQAM3		
QIO.SI	061576-R	# ZRQAM3		
QIO.UN	053314-R	# ZRQAM3		
RANDOM	100646-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDM.CN	100644-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDRX.A	077260-R	# ZRQAM1	ZRQAM2	ZRQAM3
RDRX.E	024264-R	# ZRQAM1	ZRQAM2	

F7

GLOBAL CROSS REFERENCE CREF V02

SYMBOL	VALUE	REFERENCES...		
REG.EX	044204-R	# ZRQAM3		
RETPKT	102436-R	# ZRQAM1	ZRQAM2	ZRQAM3
ROUND.	065642-R	# ZRQAM3		
RPS.RE	066624-R	# ZRQAM3		
RPT1	007646-R	# ZRQAM1	ZRQAM2	
RPT10	010356-R	# ZRQAM1	ZRQAM2	
RPT11	010444-R	# ZRQAM1	ZRQAM2	
RPT12	010512-R	# ZRQAM1	ZRQAM2	
RPT13	010560-R	# ZRQAM1	ZRQAM2	
RPT14	010660-R	# ZRQAM1	ZRQAM2	
RPT15	010756-R	# ZRQAM1	ZRQAM2	
RPT16	011056-R	# ZRQAM1	ZRQAM2	
RPT2	007732-R	# ZRQAM1	ZRQAM2	
RPT3	007776-R	# ZRQAM1	ZRQAM2	
RPT4	010062-R	# ZRQAM1	ZRQAM2	
RPT5	010126-R	# ZRQAM1	ZRQAM2	
RPT6	010214-R	# ZRQAM1	ZRQAM2	
RPT7	010260-R	# ZRQAM1	ZRQAM2	
RPT8	010276-R	# ZRQAM1	ZRQAM2	
RPT9	010324-R	# ZRQAM1	ZRQAM2	
RP.ADD	103210-R	# ZRQAM1	ZRQAM2	ZRQAM3
RP.IND	103206-R	# ZRQAM1	ZRQAM2	ZRQAM3
RP.USE	103176-R	# ZRQAM1	ZRQAM2	ZRQAM3
SA.REG	105046-R	# ZRQAM1	ZRQAM2	ZRQAM3
SB.COD	105040-R	# ZRQAM1	ZRQAM2	ZRQAM3
SCAN.E	071712-R	# ZRQAM3		
SC.CLK	021642-R	# ZRQAM1	ZRQAM2	
SC.CON	016654-R	# ZRQAM1	ZRQAM2	
SC.CTO	021104-R	# ZRQAM1	ZRQAM2	
SC.DIS	017202-R	# ZRQAM1	ZRQAM2	
SC.DST	017612-R	# ZRQAM1	ZRQAM2	
SC.DS2	017666-R	# ZRQAM1	ZRQAM2	
SC.DUP	016676-R	# ZRQAM1	ZRQAM2	
SC.ECC	017740-R	# ZRQAM1	ZRQAM2	
SC.ECD	020022-R	# ZRQAM1	ZRQAM2	
SC.EC1	020272-R	# ZRQAM1	ZRQAM2	
SC.EC2	020322-R	# ZRQAM1	ZRQAM2	
SC.EC3	020352-R	# ZRQAM1	ZRQAM2	
SC.EC4	020404-R	# ZRQAM1	ZRQAM2	
SC.EC5	020434-R	# ZRQAM1	ZRQAM2	
SC.EC6	020464-R	# ZRQAM1	ZRQAM2	
SC.EC7	020514-R	# ZRQAM1	ZRQAM2	
SC.EC8	020546-R	# ZRQAM1	ZRQAM2	
SC.EC9	020600-R	# ZRQAM1	ZRQAM2	
SC.EDC	021234-R	# ZRQAM1	ZRQAM2	
SC.FCT	020224-R	# ZRQAM1	ZRQAM2	
SC.FER	017274-R	# ZRQAM1	ZRQAM2	
SC.FE2	017362-R	# ZRQAM1	ZRQAM2	
SC.FUL	020074-R	# ZRQAM1	ZRQAM2	
SC.HWP	020702-R	# ZRQAM1	ZRQAM2	
SC.IDS	021274-R	# ZRQAM1	ZRQAM2	
SC.IOP	017130-R	# ZRQAM1	ZRQAM2	

SYMBOL VALUE REFERENCES...

SC.ISH	017452-R	#	ZRQAM1	ZRQAM2	
SC.IS2	017532-R	#	ZRQAM1	ZRQAM2	
SC.NXM	021014-R	#	ZRQAM1	ZRQAM2	
SC.ODA	020742-R	#	ZRQAM1	ZRQAM2	
SC.ODB	020772-R	#	ZRQAM1	ZRQAM2	
SC.ONL	016726-R	#	ZRQAM1	ZRQAM2	
SC.PAR	021050-R	#	ZRQAM1	ZRQAM2	
SC.POE	021524-R	#	ZRQAM1	ZRQAM2	
SC.PSP	021766-R	#	ZRQAM1	ZRQAM2	
SC.RCT	020054-R	#	ZRQAM1	ZRQAM2	
SC.RDY	021560-R	#	ZRQAM1	ZRQAM2	
SC.RSP	021670-R	#	ZRQAM1	ZRQAM2	
SC.SDI	016630-R	#	ZRQAM1	ZRQAM2	
SC.SDS	021156-R	#	ZRQAM1	ZRQAM2	
SC.SON	016750-R	#	ZRQAM1	ZRQAM2	
SC.SRI	021436-R	#	ZRQAM1	ZRQAM2	
SC.SRT	021344-R	#	ZRQAM1	ZRQAM2	
SC.SUR	021736-R	#	ZRQAM1	ZRQAM2	
SC.SWP	020642-R	#	ZRQAM1	ZRQAM2	
SC.UNK	016770-R	#	ZRQAM1	ZRQAM2	
SC.VOL	017050-R	#	ZRQAM1	ZRQAM2	
SC.576	020150-R	#	ZRQAM1	ZRQAM2	
SEND	034154-R	#	ZRQAM2	ZRQAM3	
SEQUEN	070546-R	#	ZRQAM3		
SET.CP	032420-R	#	ZRQAM2	ZRQAM3	
SET.CT	046116-R	#	ZRQAM3		
SET.UP	032514-R	#	ZRQAM2	ZRQAM3	
SFPTBL	024706-R	#	ZRQAM1		
SFT.MS	024476-R	#	ZRQAM1	ZRQAM3	
SOFT.E	073504-R	#	ZRQAM3		
SPACE4	024630-R	#	ZRQAM1	ZRQAM2	
STEP	105042-R	#	ZRQAM1	ZRQAM2	ZRQAM3
ST.COD	105036-R	#	ZRQAM1	ZRQAM2	ZRQAM3
SWEEP	066472-R	#	ZRQAM3		
SWM1	004622-R	#	ZRQAM1	ZRQAM2	
SWP.DP	024714-R	#	ZRQAM1	ZRQAM3	
SWP.ER	024706-R	#	ZRQAM1	ZRQAM3	
SWP.FL	024712-R	#	ZRQAM1	ZRQAM2	ZRQAM3
SWP.RA	024716-R	#	ZRQAM1	ZRQAM3	
SWP.TI	024720-R	#	ZRQAM1	ZRQAM3	
SWP.UC	024724-R	#	ZRQAM1	ZRQAM3	
SWP.UD	024726-R	#	ZRQAM1	ZRQAM3	
SWP.XF	024710-R	#	ZRQAM1	ZRQAM3	
SWQ1	003016-R	#	ZRQAM1	ZRQAM2	
SWQ10	003272-R	#	ZRQAM1	ZRQAM2	
SWQ11	003336-R	#	ZRQAM1	ZRQAM2	
SWQ12	003370-R	#	ZRQAM1	ZRQAM2	
SWQ13	003466-R	#	ZRQAM1	ZRQAM2	
SWQ14	003544-R	#	ZRQAM1	ZRQAM2	
SWQ15	003616-R	#	ZRQAM1	ZRQAM2	
SWQ17	003666-R	#	ZRQAM1	ZRQAM2	
SWQ19	003764-R	#	ZRQAM1	ZRQAM2	

GLOBAL CROSS REFERENCE CREF V02

```
SYMBOL  VALUE      REFERENCES...

SWQ2    003040-R  # ZRQAM1    ZRQAM2
SWQ20   004054-R  # ZRQAM1    ZRQAM2
SWQ21   004152-R  # ZRQAM1    ZRQAM2
SWQ22   004254-R  # ZRQAM1    ZRQAM2
SWQ23   004332-R  # ZRQAM1    ZRQAM2
SWQ24   004424-R  # ZRQAM1    ZRQAM2
SWQ25   004522-R  # ZRQAM1    ZRQAM2
SWQ4    003122-R  # ZRQAM1    ZRQAM2
SWQ7    003144-R  # ZRQAM1    ZRQAM2
SWQ9    003216-R  # ZRQAM1    ZRQAM2
S.DUPP  105074-R  # ZRQAM1    ZRQAM2    ZRQAM3
S.PATT  105072-R  # ZRQAM1    ZRQAM2    ZRQAM3
TALLY   077304-R  # ZRQAM1    ZRQAM2    ZRQAM3
TIME    032336-R  # ZRQAM2    ZRQAM3
TRK.SG  100640-R  # ZRQAM1    ZRQAM2    ZRQAM3
T$FREE  107160-R  # ZRQAM4
T$PTHV  000004      ZRQAM1  # ZRQAM4
T.ADDR  077634-R  # ZRQAM1    ZRQAM2    ZRQAM3
T1      043052-R    ZRQAM1  # ZRQAM3
UAM     000200    # ZRQAM1  # ZRQAM2  # ZRQAM3
UPD.IO  065246-R  # ZRQAM3
VEC.BR  044416-R  # ZRQAM3
WAIT    034612-R  # ZRQAM2    ZRQAM3
XX13    016276-R  # ZRQAM1    ZRQAM2
XX23    016320-R  # ZRQAM1    ZRQAM2
XX32    016354-R  # ZRQAM1    ZRQAM2
XX33    016402-R  # ZRQAM1    ZRQAM2
XX34    016440-R  # ZRQAM1    ZRQAM2
$END.L  107162-R  # ZRQAM4
$SAVE2  076776-R    B16MUL  # B16SAV    ZRQAM2    ZRQAM3
$SAVE3  077012-R  # B16SAV    ZRQAM2    ZRQAM3
$SAVE4  077030-R  # B16SAV    ZRQAM2    ZRQAM3
$SAVE5  077050-R    B16MUL  # B16SAV    ZRQAM2    ZRQAM3
$       VERIFY_CODE = f$verify(0)
$       on control_y then                 $ goto END
$       set verify
$       inquire ANSWER "*       Compile ZRQAE0 library? [Y/N]"
$       if ANSWER .nes. "Y" then          $ goto L1
$       bliss/pdp11 ZRQAE0.REQ/list=ZRQAE0.LIS/library=ZRQAE0.L16/source=page:53
$L1:    inquire ANSWER "*       Compile Module 1? [Y/N]"
$       if ANSWER .nes. "Y" then          $ goto L2
$       bliss/pdp11 ZRQAE0.BL1/list=ZRQAE0.LS1/object=ZRQAE0.OB1/source=page:53
$L2:    inquire ANSWER "*       Compile Module 2? [Y/N]"
$       if ANSWER .nes. "Y" then          $ goto L3
$       bliss/pdp11 ZRQAE0.BL2/list=ZRQAE0.LS2/object=ZRQAE0.OB2/source=page:53
$L3:    inquire ANSWER "*       Task-Build? [Y/N]"
$       if ANSWER .nes. "Y" then          $ goto L4
$       mcr tkb
ZRQAE0.EXE/nohd/nomm,ZRQAE0/CR/-sp=ZRQAE0.OB1,ZRQAE0.OB2,NOEIS.OLB/lb
/
par=DUMMY:2000:176000
stack=0
```

```
//
$!
$L4:
$!      TKBBIN
$!      INPUT FILENAME?    ZRQAEO.EXE
$!
$!      ZRQAEO.EXE -> ZRQAEO.BIN
$       inquire ANSWER "*       Purge Directory? [Y/N]"
$       if ANSWER .nes. "Y" then        $ goto END
$       pur
$END:   set noverify
$       if VERIFY_CODE then             $ set verify
$       exit
.
```