Micro Fiche Scan

Name of device(s) tested:
    RQDX1,RQDX2
Test description:
    RQDX1/2 DUP EXERCISOR
MAINDEC Number or Package Identifier (after SEP 1977):
    CZRQDA0
Fiche Document Part Number:
    AH-FH55A-MC
Fiche preparation date unknown, using copyright year:
    1986
Image resolution:
    8-bit gray levels, max. quality for archiving

COPYRIGHT (C) 1986 by d|i|g|i|t|a|l

```
;    0001  0     module ZRQDM1 (
;    0002  0
;    0003  0
;    0004  0     %title 'RD/RX EXERCISER'
;    0005  0                 ident = 'V02.3',
;    0006  0                 addressing_mode (absolute),
;    0007  0                 environment (noeis)
;    0008  0                 ) =
;    0009  0
;    0010  1     begin
;    0011  1
;    0012  1
;C   0013  1     %(
;C   0014  1                                 IDENTIFICATION
;C   0015  1                                 --------------
;C   0016  1
;C   0017  1             PRODUCT CODE:       AC-FH54A-MC
;C   0018  1
;C   0019  1             PRODUCT NAME:       CZRQDA0 RQDX1/2 DUP EXER
;C   0020  1
;C   0021  1             PRODUCT DATE:       06-JAN-86
;C   0022  1
;C   0023  1             MAINTAINER:         DIAGNOSTIC ENGINEERING
;C   0024  1
;C   0025  1             AUTHOR:             DESRYN DUNCAN
;C   0026  1
;C   0027  1
;C   0028  1
;C   0029  1             Copyright (C) 1986
;C   0030  1
;C   0031  1             Digital Equipment Corporation, Maynard, Massachusetts  01754
;C   0032  1
;C   0033  1             This software is furnished  under a license for use only on a single
;C   0034  1             computer  system and  may be  copied only with  the inclusion of the
;C   0035  1             above copyright notice.  This software, or any other copies thereof,
;C   0036  1             may not be provided or  otherwise made available to any other person
;C   0037  1             except for use on such system and to one who agrees to these license
;C   0038  1             terms.  Title to and  ownership of the  software  shall at all times
;C   0039  1             remain in DEC.
;C   0040  1
;C   0041  1             the information in this document is subject to change without notice
;C   0042  1             and should  not be construed  as a commitment  by Digital  Equipment
;C   0043  1             Corporation.
;C   0044  1
;C   0045  1             DEC assumes  no  responsibility  for  the use or  reliability of its
;C   0046  1             software on equipment which is not supplied by DEC.
;C   0047  1
;C   0048  1
;C   0049  1             The following are trademarks of Digital Equipment Corporation:
;C   0050  1
;C   0051  1             DIGITAL         PDP             UNIBUS          MASSBUS
;C   0052  1             DEC             DECUS           DECTAPE
```

```
;  C 0053  1
;  C 0054  1
;  C 0055  1          REVISION HISTORY:
;  C 0056  1
;  C 0057  1
;  C 0058  1          REV 1.0   06-JAN-86   DUP WAS REMOVED FROM ORIGINAL EXERCISER AND PLACED  INTO THIS
;  C 0059  1                                EXERCISER. HOWEVER, BECAUSE OF THE WAY THE EXERCISER WAS WRITTEN,
;  C 0060  1                                SOME MSCP PARTS HAD TO BE INCLUDED IN THIS EXERCISER.
;  C 0061  1
```

```
;  C 0062  1
;  C 0063  1                    TABLE OF CONTENTS
;  C 0064  1
;  C 0065  1
;  C 0066  1
;  C 0067  1          1.0    GENERAL INFORMATION
;  C 0068  1          1.1    PROGRAM ABSTRACT
;  C 0069  1          1.2    SYSTEM REQUIREMENTS
;  C 0070  1          1.2.1  HARDWARE REQUIREMENTS
;  C 0071  1          1.2.2  SOFTWARE REQUIREMENTS
;  C 0072  1          1.3    RELATED DOCUMENTS AND STANDARDS
;  C 0073  1          1.4    DIAGNOSTIC HIERARCHY PREREQUISITES
;  C 0074  1          1.5    ASSUMPTIONS
;  C 0075  1          1.6    MEMORY MAP
;  C 0076  1
;  C 0077  1          2.0    OPERATING INSTRUCTIONS
;  C 0078  1          2.1    HARDWARE QUESTIONS
;  C 0079  1          2.2    SOFTWARE QUESTIONS
;  C 0080  1
;  C 0081  1          3.0    ERROR TYPES
;  C 0082  1          3.1    ERROR INFORMATION
;  C 0083  1          3.2    INITIALIZATION ERRORS
;  C 0084  1          3.3    EXERCISER ERRORS
;  C 0085  1          3.4    ERROR LOG MESSAGES
;  C 0086  1          3.5    MSCP ERRORS
;  C 0087  1          3.6    SAMPLE ERROR STATEMENT
;  C 0088  1
;  C 0089  1          4.0    PERFORMANCE AND PROGRESS REPORTS
;  C 0090  1
;  C 0091  1          5.0    TEST SUMMARY
;  C 0092  1          5.1    INITIALIZATION SUBTEST
;  C 0093  1          5.2    EXERCISER
;  C 0094  1          5.3    DROP UNIT SUMMARY
;  C 0095  1
;  C 0096  1          6.0    ERROR CODES
;  C 0097  1
;  C 0098  1          7.0    DATA PATTERNS
```

```
;  C 0099  1
;  C 0100  1
;  C 0101  1
;  C 0102  1
;  C 0103  1
;  C 0104  1
;  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
```

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

This program will functionally verify the DUP utility of the RQDX or RUX50 Controller/Disk Drive subsystems. It is designed to verify that the subsystem is functioning correctly and operating within design specifications. However, because of the way in which the program was originally written, the DUP testing is interleaved with MSCP testing and only executed after a particular number of MSCP I/O's have been done.As a result, there may be no new reads or writes done by DUP for 4 to 5 passes.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE REQUIREMENTS

LSI - 11/23 processor with 28K or more of memory, console device (eg. VT100) and RQDX or RUX50 controller board and attached RD51/52/53 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

F1


```
; C 0152  1
; C 0153  1
; C 0154  1
```

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 0155  1          1.6     MEMORY MAP
; C 0156  1                  ----------
; C 0157  1
; C 0158  1
; C 0159  1                  Memory layout on 28k machine - XXDP environment
; C 0160  1                  -----------------------------------------------
; C 0161  1
; C 0162  1                                                      ADDRESS
; C 0163  1
; C 0164  1                        +-----------------------+     160000
; C 0165  1                        !                       !
; C 0166  1                        !     XXDP+ MONITOR     !
; C 0167  1                        !                       !
; C 0168  1                        +-----------------------+
; C 0169  1                        !                       !
; C 0170  1                        !  XXDP+   TTY I/O AREA !
; C 0171  1                        !                       !
; C 0172  1                        +-----------------------+
; C 0173  1                        !                       !
; C 0174  1                        !                       !
; C 0175  1                        !         DRS           !
; C 0176  1                        !                       !
; C 0177  1                        !                       !
; C 0178  1                        +-----------------------+     121000
; C 0179  1                        !                       !
; C 0180  1                        !      EXERCISER        !
; C 0181  1                        !                       !
; C 0182  1                        +-----------------------+     2000
; C 0183  1                        !                       !
; C 0184  1                        !        STACKS         !
; C 0185  1                        !                       !
; C 0186  1                        +-----------------------+     400
; C 0187  1                        !                       !
; C 0188  1                        !      VECTOR AREA      !
; C 0189  1                        !                       !
; C 0190  1                        +-----------------------+     0
; C 0191  1
; C 0192  1
; C 0193  1                  In a machine  with more memory,  free space will occur
; C 0194  1                  between the exerciser and the DRS.
```

```
;C 0195  1
;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
```

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 RQDX 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 0244  1
;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
```

4. VECTOR ADDRESS (O) 154 ?

   Answer with the interrupt vector ofthe same RQDX 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-RQDX 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 RQDX or RUX50 controller board. 0 is the default answer.

7. ALSO RUN DUP EXERCISER (L) Y ?

   ANSWER Y TO HAVE TESTS PERFORMED SPECIFICALLY WITH THE DIAGNOSTIC BLOCKS. DUP TESTING IS INTERLEAVED WITH NORMAL EXERCISER TESTING.

8. WRITE ON DIAGNOSTIC AREA (L) Y ?

   ANSWERING Y TO THIS QUESTION ADDS WRITE TESTING IN THE DIAGNOSTIC BLOCK AREA. THIS CAN BE USED TO DETERMINE WHETHER A UNIT IS WRITING PROPERLY, WITHOUT USING THE CUSTOMER AREA.

9. 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.

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

    Enter in octal the less significant 16-bit word of the lowest

```
;  C 0297  1          LBN address in the test range. The value may be from 000000
;  C 0298  1          to 177777.
;  C 0299  1
;  C 0300  1
;  C 0301  1        11.  HIGHER OCTAL WORD OF BEGINNING LBN ADDRESS (0) 0?
;  C 0302  1
;  C 0303  1          Enter in octal the more significant 16-bit word of the
;  C 0304  1          lowest LBN address in the test range.
;  C 0305  1
;  C 0306  1
;  C 0307  1        12.  LOWER OCTAL WORD OF ENDING LBN ADDRESS (0) 150477?
;  C 0308  1
;  C 0309  1          Enter in octal the less significant 16-bit word of the
;  C 0310  1          highest LBN address in the test range. 150477 is the
;  C 0311  1          highest LBN address for an RD52.
;  C 0312  1
;  C 0313  1
;  C 0314  1        13.  HIGHER OCTAL WORD OF ENDING LBN ADDRESS (0) 0?
;  C 0315  1
;  C 0316  1          Enter in octal the more significant 16-bit word of the
;  C 0317  1          highest LBN address in the test range.
;  C 0318  1
;  C 0319  1
;  C 0320  1           Note:
;  C 0321  1                The four previous questions are usually software
;  C 0322  1                Parameter questions, but since three different disk
;  C 0323  1                drives exist on the subsystem, this becomes a  unit
;  C 0324  1                by unit question.  It is possible to specify an LBN
;  C 0325  1                which is too large since we are dealing with different
;  C 0326  1                drives. The program will check for block number bounds,
;  C 0327  1                and, if they are exceeded, will assign the maximum
;  C 0328  1                bounds for that drive.
;  C 0329  1
;  C 0330  1
;  C 0331  1
;  C 0332  1      14.  WRITE ON CUSTOMER DATA AREA ON THIS DISK UNIT (L) ?
;  C 0333  1
;  C 0334  1          Answering YES will destroy any customer data that is  on
;  C 0335  1          the  disk; therefore, the following warning message will
;  C 0336  1          appear, followed by a confirmation prompt:
;  C 0337  1
;  C 0338  1          ** WARNING - CUSTOMER DATA AREA WILL BE OVERWRITTEN! ...
;  C 0339  1           CONFIRM (L) ?
;  C 0340  1
;  C 0341  1          This question will default to NO if the operator has de-
;  C 0342  1          cided  to  bypass  the  hardware  questions.  Otherwise,
;  C 0343  1          there is no default.
```

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 ALLOW leading zeros ENTERED FOR numeric values.
   For example, for 14 minutes past midnight, you would enter 14, and for 30 minutes past 3 in the afternoon, enter 1530.

3. HARD 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 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
;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
```

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 RQDX 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 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
;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
```

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 compare the results.

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

15. RUNNING UNDER THE A.P.T. MONITOR (L) N ?

    THIS QUESTION SHOULD BE ANSWERED N (DEFAULT) IN THE FIELD. IT ENABLES THE PROGRAM TO KNOW THAT IT IS RUNNING UNDER A SPECIAL (AUTOMATED PRODUCT TEST) MONITOR.

```
;  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
;  C 0509  1
;  C 0510  1
;  C 0511  1
;  C 0512  1
;  C 0513  1
;  C 0514  1
;  C 0515  1
;  C 0516  1
;  C 0517  1
;  C 0518  1
;  C 0519  1
;  C 0520  1
;  C 0521  1
;  C 0522  1
;  C 0523  1
;  C 0524  1
;  C 0525  1
;  C 0526  1
;  C 0527  1
;  C 0528  1
;  C 0529  1
;  C 0530  1
;  C 0531  1
;  C 0532  1
;  C 0533  1
;  C 0534  1
;  C 0535  1
;  C 0536  1
;  C 0537  1
;  C 0538  1
;  C 0539  1
;  C 0540  1
```

THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED DISK UNITS.

16. 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.

17. 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.

18. 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.

19. 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).

20. NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM) (D) 16 ?
PATTERN VALUES (0) ?

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 0541  1
;  C 0542  1
;  C 0543  1
;  C 0544  1
;  C 0545  1
;  C 0546  1
;  C 0547  1
;  C 0548  1
;  C 0549  1
;  C 0550  1
;  C 0551  1
;  C 0552  1
;  C 0553  1
;  C 0554  1
;  C 0555  1
;  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
```

3.0 ERROR TYPES

This program has four types of error classifications; system fatal, drive fatal, hard and soft.

SYSTEM FATAL ERRORS

System fatal errors are used to indicate that an error was detected by the Diagnostic Supervisor in relation to loading/controlling the diagnostic process.

The content of each error is such that it should be self explanatory. However, the messages utilize some terms that are specific to the disk subsystem, and may require some getting use to.

DRIVE FATAL ERRORS

Drive fatal errors are a result of:

an error that is considered fatal to the drive, but testing will continue.

HARD ERRORS

Hard errors are a result of:

1. retries of a soft error or *
2. a non-recoverable error
3. a soft error if retries are not set.

* Note: Retries are executed in the controller

SOFT ERRORS

Soft errors are media related errors. All soft errors will be retried by the controller.

Note: Soft errors are retrieved from the controller via the error log capabilities of MSCP.

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

D2


```
;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
;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
```

3.3 EXERCISER ERRORS

If any errors originating from MSCP end message packets are reported, the MSCP exerciser diagnostic ZRQAH0 should be run to determine the exact nature of the error.

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)

```
;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
;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
```

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	SEK	DRV	HST	-- SFT ERS -- DAT	SEK	DRV	HST
X	XXXX	XXXX	XXXX	XXXXX	XXXXXX	X	X	X	X	X	X	X	X
.	..	..	..	...	....	.	.	.	.	.	.	.	.
.	..	..	..	...	....	.	.	.	.	.	.	.	.

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.

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.

H2


```
; 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
; 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
```

6.0 ERROR CODES

This section describes the 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.

I2

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 0881  1
;C 0882  1
;C 0883  1          41  Drive error.                          See #40.
;C 0884  1
;C 0885  1          42  Host write compare error.             Error detected when Host
;C 0886  1                                                    CPU compared the data
;C 0887  1                                                    written and read back. May
;C 0888  1                                                    be a problem with the Host
;C 0889  1                                                    or Controller RAM.
;C 0890  1
;C 0891  1          43 Message from internal diagnostics      See #40.
;C 0892  1
;C 0893  1          44  Duplicate unit number detected       ---
;C 0894  1              by the Controller.
;C 0895  1
;C 0896  1          45  Unknown end code received.           Problem with the Control-
;C 0897  1                                                    ler microcode or the port/
;C 0898  1                                                    DMA interface.
;C 0899  1
;C 0900  1
;C 0901  1
;C 0902  1
;C 0903  1          SOFT ERRORS
;C 0904  1          ----------
;C 0905  1
;C 0906  1          50  Controller error.                    See error-log packet for
;C 0907  1                                                    details as the exact cause
;C 0908  1                                                    may not be evident.
;C 0909  1
;C 0910  1          51  Host memory access error.            See #50.
;C 0911  1
;C 0912  1          52  Disk transfer error.                 See #50.
;C 0913  1
;C 0914  1          53  'Standard Disk Interconnect'        See #50.
;C 0915  1              error.
;C 0916  1
;C 0917  1          54  'Small Disk' error.                  See #50.
;C 0918  1
;C 0919  1
;C 0920  1
;C 0921  1
;C 0922  1          DUP ERRORS
;C 0923  1          ----------
;C 0924  1
;C 0925  1          60 Unable to load local controller DUP media.
;C 0926  1
;C 0927  1          61 (Not used)
;C 0928  1
;C 0929  1          62 Illegal unit number.
;C 0930  1
;C 0931  1          63 Illegal relative or physical block.
;C 0932  1
;C 0933  1          64 Device error.
```

K2


```
;  C 0934  1
;  C 0935  1                65 Zero length message.
;  C 0936  1
;  C 0937  1                66 Unknown DUP status code.
;  C 0938  1
;  C 0939  1                67 Invalid command.
```

```
;  C 0940  1
;  C 0941  1
;  C 0942  1
;  C 0943  1                DUP ERRORS  (CONTINUED)
;  C 0944  1                ----------
;  C 0945  1
;  C 0946  1
;  C 0947  1                68 No region available.
;  C 0948  1
;  C 0949  1                69 No region suitable.
;  C 0950  1
;  C 0951  1                70 Program not known.
;  C 0952  1
;  C 0953  1                71 Load failure.
;  C 0954  1
;  C 0955  1                72 Standalone.
;  C 0956  1
;  C 0957  1                73 Unknown DUP status code.
;  C 0958  1
;  C 0959  1
```

M2


```
;C 0960  1
;C 0961  1
;C 0962  1
;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
```

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

N2


```
; C 1012  1          Pattern 9       0000    000000      0 000 000 000 000 000
; C 1013  1                          0000    000000      0 000 000 000 000 000
; C 1014  1                          0000    000000      0 000 000 000 000 000
; C 1015  1                          FFFF    177777      1 111 111 111 111 111
; C 1016  1                          FFFF    177777      1 111 111 111 111 111
; C 1017  1                          FFFF    177777      1 111 111 111 111 111
; 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                          0000    000000      0 000 000 000 000 000
; C 1023  1                          FFFF    177777      1 111 111 111 111 111
; 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                          0000    000000      0 000 000 000 000 000
; C 1027  1                          FFFF    177777      1 111 111 111 111 111
; C 1028  1
; C 1029  1          Pattern 10      B6D9    133331      1 011 011 011 011 001
; C 1030  1
; C 1031  1          Pattern 11      5555    052525      0 101 010 101 010 101
; C 1032  1                          5555    052525      0 101 010 101 010 101
; C 1033  1                          5555    052525      0 101 010 101 010 101
; C 1034  1                          AAAA    125252      1 010 101 010 101 010
; C 1035  1                          AAAA    125252      1 010 101 010 101 010
; C 1036  1                          AAAA    125252      1 010 101 010 101 010
; 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                          5555    052525      0 101 010 101 010 101
; C 1042  1                          AAAA    125252      1 010 101 010 101 010
; 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                          5555    052525      0 101 010 101 010 101
; C 1046  1                          AAAA    125252      1 010 101 010 101 010
```

```
;C 1047  1          Pattern 12      2D2D    026455      0 010 110 100 101 101
;C 1048  1                          2D2D    026455      0 010 110 100 101 101
;C 1049  1                          2D2D    026455      0 010 110 100 101 101
;C 1050  1                          D2D2    151322      1 101 001 011 010 010
;C 1051  1                          D2D2    151322      1 101 001 011 010 010
;C 1052  1                          D2D2    151322      1 101 001 011 010 010
;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                          2D2D    026455      0 010 110 100 101 101
;C 1058  1                          2D2D    026455      0 010 110 100 101 101
;C 1059  1                          D2D2    151322      1 101 001 011 010 010
;C 1060  1                          2D2D    026455      0 010 110 100 101 101
;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                          D2D2    151322      1 101 001 011 010 010
;C 1064  1                          2D2D    026455      0 010 110 100 101 101
;C 1065  1                          D2D2    151322      1 101 001 011 010 010
;C 1066  1                          2D2D    026455      0 010 110 100 101 101
;C 1067  1
;C 1068  1          Pattern 13      6DB6    066666      0 110 110 110 110 110
;C 1069  1
;C 1070  1          Pattern 14      0001    000001      0 000 000 000 000 001
;C 1071  1                          0002    000002      0 000 000 000 000 010
;C 1072  1                          0004    000004      0 000 000 000 000 100
;C 1073  1                          0008    000010      0 000 000 000 001 000
;C 1074  1                          0010    000020      0 000 000 000 010 000
;C 1075  1                          0020    000040      0 000 000 000 100 000
;C 1076  1                          0040    000100      0 000 000 001 000 000
;C 1077  1                          0080    000200      0 000 000 010 000 000
;C 1078  1                          0100    000400      0 000 000 100 000 000
;C 1079  1                          0200    001000      0 000 001 000 000 000
;C 1080  1                          0400    002000      0 000 010 000 000 000
;C 1081  1                          0800    004000      0 000 100 000 000 000
;C 1082  1                          1000    010000      0 001 000 000 000 000
;C 1083  1                          2000    020000      0 010 000 000 000 000
;C 1084  1                          4000    040000      0 100 000 000 000 000
;C 1085  1                          8000    100000      1 000 000 000 000 000
```

```
; C 1086  1          Pattern 15      FFFE    177776          1 111 111 111 111 110
; C 1087  1                          FFFD    177775          1 111 111 111 111 101
; C 1088  1                          FFFB    177773          1 111 111 111 111 011
; C 1089  1                          FFF7    177767          1 111 111 111 110 111
; C 1090  1                          FFEF    177757          1 111 111 111 101 111
; C 1091  1                          FFDF    177737          1 111 111 111 011 111
; C 1092  1                          FFBF    177677          1 111 111 110 111 111
; C 1093  1                          FF7F    177577          1 111 111 101 111 111
; C 1094  1                          FEFF    177377          1 111 111 011 111 111
; C 1095  1                          FDFF    176777          1 111 110 111 111 111
; C 1096  1                          FBFF    175777          1 111 101 111 111 111
; C 1097  1                          F7FF    173777          1 111 011 111 111 111
; C 1098  1                          EFFF    167777          1 110 111 111 111 111
; C 1099  1                          DFFF    157777          1 101 111 111 111 111
; C 1100  1                          BFFF    137777          1 011 111 111 111 111
; C 1101  1                          7FFF    077777          0 111 111 111 111 111
; C 1102  1
; C 1103  1          Pattern 16      B6D9    133331          1 011 011 011 011 001
; C 1104  1                          B6D9    133331          1 011 011 011 011 001
; C 1105  1                          B6D9    133331          1 011 011 011 011 001
; C 1106  1                          DB6C    155554          1 101 101 101 101 100
; C 1107  1                          DB6C    155554          1 101 101 101 101 100
; C 1108  1                          DB6C    155554          1 101 101 101 101 100
; 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                          B6D9    133331          1 011 011 011 011 001
; C 1114  1                          DB6C    155554          1 101 101 101 101 100
; 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                          B6D9    133331          1 011 011 011 011 001
; C 1118  1                          DB6C    155554          1 101 101 101 101 100
```

```
;  C 1119  1          Pattern 17      (LBN)*   (LBN)          (LBN)
;  C 1120  1                          8D36     106466   1 000 110 100 110 110
;  C 1121  1                          8D36     106466   1 000 110 100 110 110
;  C 1122  1                          72C9     071311   0 111 001 011 001 001
;  C 1123  1                          72C9     071311   0 111 001 011 001 001
;  C 1124  1                          72C9     071311   0 111 001 011 001 001
;  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                          8D36     106466   1 000 110 100 110 110
;  C 1128  1                          8D36     106466   1 000 110 100 110 110
;  C 1129  1                          72C9     071311   0 111 001 011 001 001
;  C 1130  1                          72C9     071311   0 111 001 011 001 001
;  C 1131  1                          72C9     071311   0 111 001 011 001 001
;  C 1132  1                          72C9     071311   0 111 001 011 001 001
;  C 1133  1                          72C9     071311   0 111 001 011 001 001
;  C 1134  1                          8D36     106466   1 000 110 100 110 110
;  C 1135  1                          8D36     106466   1 000 110 100 110 110
;  C 1136  1                          8D36     106466   1 000 110 100 110 110
;  C 1137  1                          8D36     106466   1 000 110 100 110 110
;  C 1138  1                          8D36     106466   1 000 110 100 110 110
;  C 1139  1                          8D36     106466   1 000 110 100 110 110
;  C 1140  1
;  C 1141  1              * This word position contains the number of the locical
;  C 1142  1                block to be written.
;  C 1143  1
;  C 1144  1
;  C 1145  1
;  C 1146  1          Pattern 18      8D36     106466   1 000 110 100 110 110
;  C 1147  1                          (LBN)    (LBN)          (LBN)
;  C 1148  1                          72C9     071311   0 111 001 011 001 001
;  C 1149  1                          8D36     106466   1 000 110 100 110 110
;  C 1150  1                          8D36     106466   1 000 110 100 110 110
;  C 1151  1                          8D36     106466   1 000 110 100 110 110
;  C 1152  1                          72C9     071311   0 111 001 011 001 001
;  C 1153  1                          72C9     071311   0 111 001 011 001 001
;  C 1154  1                          72C9     071311   0 111 001 011 001 001
;  C 1155  1                          72C9     071311   0 111 001 011 001 001
;  C 1156  1                          8D36     106466   1 000 110 100 110 110
;  C 1157  1                          8D36     106466   1 000 110 100 110 110
;  C 1158  1                          8D36     106466   1 000 110 100 110 110
;  C 1159  1                          8D36     106466   1 000 110 100 110 110
;  C 1160  1                          8D36     106466   1 000 110 100 110 110
;  C 1161  1                          72C9     071311   0 111 001 011 001 001
;  C 1162  1                          72C9     071311   0 111 001 011 001 001
;  C 1163  1                          72C9     071311   0 111 001 011 001 001
;  C 1164  1                          72C9     071311   0 111 001 011 001 001
;  C 1165  1                          72C9     071311   0 111 001 011 001 001
;  C 1166  1                          72C9     071311   0 111 001 011 001 001
```

E3


```
;  C 1167  1          Pattern 19      (LBN)    (LBN)          (LBN)
;  C 1168  1                          B999     134631   1 011 100 110 011 001
;  C 1169  1                          B999     134631   1 011 100 110 011 001
;  C 1170  1                          4666     043146   0 100 011 001 100 110
;  C 1171  1                          4666     043146   0 100 011 001 100 110
;  C 1172  1                          4666     043146   0 100 011 001 100 110
;  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                          B999     134631   1 011 100 110 011 001
;  C 1176  1                          B999     134631   1 011 100 110 011 001
;  C 1177  1                          4666     043146   0 100 011 001 100 110
;  C 1178  1                          4666     043146   0 100 011 001 100 110
;  C 1179  1                          4666     043146   0 100 011 001 100 110
;  C 1180  1                          4666     043146   0 100 011 001 100 110
;  C 1181  1                          4666     043146   0 100 011 001 100 110
;  C 1182  1                          B999     134631   1 011 100 110 011 001
;  C 1183  1                          B999     134631   1 011 100 110 011 001
;  C 1184  1                          B999     134631   1 011 100 110 011 001
;  C 1185  1                          B999     134631   1 011 100 110 011 001
;  C 1186  1                          B999     134631   1 011 100 110 011 001
;  C 1187  1                          B999     134631   1 011 100 110 011 001
;  C 1188  1
;  C 1189  1          Pattern 20      B999     134631   1 011 100 110 011 001
;  C 1190  1                          (LBN)    (LBN)          (LBN)
;  C 1191  1                          4666     043146   0 100 011 001 100 110
;  C 1192  1                          B999     134631   1 011 100 110 011 001
;  C 1193  1                          B999     134631   1 011 100 110 011 001
;  C 1194  1                          B999     134631   1 011 100 110 011 001
;  C 1195  1                          4666     043146   0 100 011 001 100 110
;  C 1196  1                          4666     043146   0 100 011 001 100 110
;  C 1197  1                          4666     043146   0 100 011 001 100 110
;  C 1198  1                          4666     043146   0 100 011 001 100 110
;  C 1199  1                          B999     134631   1 011 100 110 011 001
;  C 1200  1                          B999     134631   1 011 100 110 011 001
;  C 1201  1                          B999     134631   1 011 100 110 011 001
;  C 1202  1                          B999     134631   1 011 100 110 011 001
;  C 1203  1                          B999     134631   1 011 100 110 011 001
;  C 1204  1                          4666     043146   0 100 011 001 100 110
;  C 1205  1                          4666     043146   0 100 011 001 100 110
;  C 1206  1                          4666     043146   0 100 011 001 100 110
;  C 1207  1                          4666     043146   0 100 011 001 100 110
;  C 1208  1                          4666     043146   0 100 011 001 100 110
;  C 1209  1                          4666     043146   0 100 011 001 100 110
;  C 1210  1
;  C 1211  1          Pattern 21      (LBN)    (LBN)          (LBN)
;  C 1212  1
;    1213  1       )%
```

F3


```
;    1214  1     %sbttl 'PROGRAM HEADER'
;    1215  1
;    1216  1     library 'ZRQDA0.L16';                   ! RDRX EXERCISER GLOBAL LIBRARY
;    1217  1
;    1218  1     !MMMrequire 'BLSMAC.REQ';               ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    1219  1     require 'HSAXA0.BLB';                   ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    2960  1
;    2961  1     literal
;    2962  1         DS$NBR_OF_TESTS = 1;                ! NUMBER OF TESTS IN THIS DIAGNOSTIC
;    2963  1
;    2964  1     EQUALS;
;    2965  1
;    2966  1     POINTER (ALL);
;    2967  1
;    2968  1     !+
;    2969  1     ! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;    2970  1     ! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;    2971  1     !-
;    2972  1
;    2973  1     !ZZZ HEADER (%ascii'ZRQA', %ascii'G', %ascii'0', 32000, 1, PRI00);      !ZZZ NEED POSITIVE NUMBER
;E)  2974  1     HEADER (%ascii'ZRQD', %ascii'A', %ascii'0', 32000, 1, PRI00,1);         !ZZZ FINAL 1 = NO TESTING ON TRAPS (SAVE TIM
;E T2975  1     !MMM HEADER (%ascii'ZRQA', %ascii'X', %ascii'8', 32000, 1, PRI00,1);          !MMM FINAL 1 = NO TESTING ON TRAPS (SAV
;    2976  1
```

```
;   2977  1     %sbttl 'DISPATCH TABLE'
;   2978  1
;   2979  1     !+
;   2980  1     ! THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;   2981  1     ! IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;   2982  1     !-
;   2983  1
;   2984  1     DISPATCH (DS$NBR_OF_TESTS);
```

H3


```
;  2985  1     %sbttl 'GLOBAL DATA SECTION'
;  2986  1
;  2987  1     !+
;  2988  1     ! THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
;  2989  1     ! IN MORE THAN ONE TEST.
;  2990  1     !-
;  2991  1
;  2992  1     psect
;  2993  1         global = $FFF$ (read, write, noexecute, global, concatenate);
;  2994  1
;  2995  1     global
;  2996  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;  2997  1                         ! RUN-TIME CONTROLLER STATUS TABLES
;  2998  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;  2999  1                         ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;  3000  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;  3001  1                         ! DRIVER CONTROLLER TABLES
;  3002  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;  3003  1                         ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;  3004  1         RDRX_ADDR : ref rdrx field (RC_REG),
;  3005  1                         ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;  3006  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;  3007  1                         ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;  3008  1
;  3009  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
;  3010  1                         !CONTAINS LO+ HI LBN FIELDS FOR SEQUENTIAL       !ZZZ
;  3011  1                         !I/O TRANSFER FOR EACH UNIT.                     !ZZZ
;  3012  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;  3013  1                         ! STATISTICS TABLES
;  3014  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;  3015  1                         ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;  3016  1
;  3017  1          DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),      !BUFFER FOR DUP  ZZZ
;  3018  1                         !INFO FROM RECEIVE + SEND CMDS                   ZZZ
;  3019  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED] INITIAL (BYTE (REP    !ZZZ
;  3020  1                 MAX_UNITS OF (1))),          !CURRENT TRACK DIRECTION    ZZZ
;  3021  1         RDM_CNT : WORD INITIAL (RDM_LEN),    !NO OF RANDOM NOS   \\KEEP  ZZZ
;  3022  1         RANDOM : VECTOR [RDM_LEN, WORD],     !RANDOM NO. TABLE   //TOGETHER  ZZZ
;  3023  1
;  3024  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;  3025  1                         ! STATISTICS TABLE FOR CONTROLLER ERRORS
;  3026  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;  3027  1                         ! MSCP PACKET POOL
;  3028  1         IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  3029  1                         ! ADDRESS OF AN MSCP PACKET (INTERUPT PROCESSING)
;  3030  1         PKT_USE : vector [PKT_CNT, byte, signed],
;  3031  1                         ! MSCP PACKET POOL ALLOCATION TABLE
;  3032  1         RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  3033  1                         ! RETURN PACKET POOL
;  3034  1         RP_USE : vector [RP_CNT, byte, signed],
;  3035  1                         ! RETURN PACKET POOL ALLOCATION TABLE
;  3036  1         RP_INDX : word,      ! CURRENT RETURN PACKET INDEX
;  3037  1         RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
```

I3


```
;  3038  1                                  ! CURRENT RETURN PACKET ADDRESS
;  3039  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;  3040  1                                  ! ERROR-LOG PACKET SAVE AREA
;  3041  1          BUFF_ADDR : vector [MAX_BUF_CNT],                ! TABLE OF I/O BUFFER DESCRIPTORS
;  3042  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],   ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;  3043  1          IODQ : vector [IODQ_LEN, byte],                  ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;  3044  1          IODQ_IN : word,                                  ! I/O DONE QUEUE IN POINTER
;  3045  1          IODQ_OUT : word,                                 ! I/O DONE QUEUE OUT POINTER
;  3046  1          ENTRY_REASON : byte,                             ! CURRENT OPERATOR COMMAND
;  3047  1          EOP_FLAG : byte,                                 ! END-OF-PASS FLAG
;  3048  1          DUP_FLAGS : WORD,                                !DUP FLAGS       ZZZ
;  3049  1          CCTLR : word,                                    ! NUMBER OF "CURRENT" CONTROLLER
;  3050  1          CDISK : word,                                    ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  3051  1          CUOFF : word,                                    ! CURRENT UNIT CST OFFSET
;  3052  1          CTLR_CNT : word,                                 ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;  3053  1          DUR : vector [MAX_UNITS, byte],                  ! DROP UNIT REASON
;  3054  1          QIO : vector [MAX_CTLR, byte],                   ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;  3055  1          FREE_MEM_ADDR,                                   ! START OF FREE MEMORY
;  3056  1          BYTS_PER_QIO : word,                             ! SIZE (BYTES) OF AN I/O BUFFER
;  3057  1          ST_CODE : word,                                  ! CURRENT STATUS CODE
;  3058  1          SB_CODE : word,                                  ! CURRENT SUB-CODE
;  3059  1          STEP : word,                                     ! CURRENT STEP IN HARD INIT
;  3060  1          OF_RC : signed word,                             ! OFFSET (0 OR 2) TO READ IP OR SA
;  3061  1          SA_REG : word,                                   ! STORAGE FOR SA REGISTER READS AND WRITES
;  3062  1          CMD_TIME : word,                                 ! COMMAND TIMEOUT VALUE (IN SECONDS)
;  3063  1          NEX : word,                                      ! NON-EXISTENT MEMORY TRAP INDICATOR
;  3064  1          CRN_LOW : word,                                  ! COMMAND REF NUMBER OF LAST COMMAND SENT
;  3065  1          CRN_HIGH : word,                                 ! COMMAND REF NUMBER (HI ORDER)
;  3066  1          TEMP1 : WORD,                                    !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;  3067  1          TEMP2 : WORD,                                    !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;  3068  1          CREDIT_BAL : word,                               ! CREDIT BALANCE
;  3069  1          NEXT_PRT_USE : byte,                             ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;  3070  1          HOURS : byte,                                    ! TIME OF DAY (HOURS)
;  3071  1          MINUTES : byte,                                  ! TIME OF DAY (MINUTES)
;  3072  1          CLK_TICKS : word,                                ! TIME OF DAY (LINE-CLOCK TICKS)
;  3073  1          FER0_LBN : word,                                 !LO LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;  3074  1          FER1_LBN : word,                                 !HI LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;  3075  1          CLK_PRESENT : byte,                              ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  3076  1          HOE_FLAG : byte,                                 ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;  3077  1
;  3078  1          TYPER : VECTOR [MAX_UNITS, WORD],   !READ I/O COUNTER                    ZZZ MMM
;  3079  1          TYPEW : VECTOR [MAX_UNITS, WORD],   !WRITE I/O COUNTER                   ZZZ MMM
;  3080  1          BAL_IN_PROGRESS : VECTOR [MAX_UNITS, WORD],  !FLAG SET TO BALANCE I/O TYPES      ZZZ MMM
;  3081  1          FORCE_WR : VECTOR [MAX_UNITS, WORD],          ! MMM
;  3082  1          CSR_MEM : WORD,                               ! MMM
;  3083  1          CSR_ADD : WORD INITIAL (%o'172100'),          ! MMM
;  3084  1          S_PATTERN : WORD,                   !PATTERN FOR DUP WRITES              ZZZ
;  3085  1          S_DUPPKT : WORD,                    !DBN BYTE COUNTER                    ZZZ
;  3086  1          P_INDEX : SIGNED WORD,              !CURRENT MESSAGE PACKET INDEX        ZZZ
;  3087  1          RD_COUNT : WORD INITIAL (0),        ! NUMBER OF WINCHESTER UNITS         ZZZ
;  3088  1          BRLEVEL : WORD,                     !BUS REQUEST LEVEL FROM OPERATOR     ZZZ
;  3089  1          D_FAIL : BYTE,                      !SIGNIFIES DUP TYPE ERROR            ZZZ
;  3090  1          FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
```

J3


```
;    3091  1          FER_LBN : word,                                          ! LBN OF THE "FORCED ERROR" BLOCK
;    3092  1          FER_BC : word,                                           ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    3093  1          INIT_OCCURED : byte initial (byte (FALSE)),              ! EXERCISER INITIALIZATION COMPLETE
;    3094  1          ADDR_VECT_OK : byte initial (byte (FALSE));              ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    3095  1
;    3096  1      ERRTBL;
```

K3


```
;    3097  1     %sbttl 'GLOBAL TEXT SECTION'
;    3098  1
;    3099  1     !+
;    3100  1     ! THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
;    3101  1     ! MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
;    3102  1     ! MORE THAN ONE TEST.
;    3103  1     !-
;    3104  1
;    3105  1     global bind
;    3106  1     !
;    3107  1     ! HARDWARE DIALOG
;    3108  1     !
;    3109  1         PTCH1 = uplit (%asciz'                                                  '),  !AREA TO PATCH IF NEEDED     ZZZ
;    3110  1         PTCH2 = uplit (%asciz'                                                  '),  !                            ZZZ
;    3111  1         PTCH3 = uplit (%asciz'                                                  '),  !                            ZZZ
;    3112  1         PTCH4 = uplit (%asciz'                                                  '),  !                            ZZZ
;    3113  1         PTCH5 = uplit (%asciz'                                                  '),  !                            ZZZ
;    3114  1         HWQ1 = uplit (%asciz'IP address'),
;    3115  1         HWQ2 = uplit (%asciz'Vector'),
;    3116  1         HWQ3 = uplit (%asciz'BR Level [usually 4-RQDX 5-RUX50]'),                  !ZZZ
;    3117  1         HWQ4 = uplit (%asciz'Drive number'),                                       !ZZZ
;    3118  1         HWQ5 = uplit (%asciz'Test entire customer area of this disk'),             !ZZZ
;    3119  1         HWQ6A = uplit (%asciz'Lower octal word of beginning LBN address'),         !ZZZ
;    3120  1         HWQ6B = uplit (%asciz'Higher octal word of beginning LBN address'),        !ZZZ
;    3121  1         HWQ7A = uplit (%asciz'Lower octal word of ending LBN address'),            !ZZZ
;    3122  1         HWQ7B = uplit (%asciz'Higher octal word of ending LBN address'),           !ZZZ
;    3123  1         HWQ8 = uplit (%asciz'Write on customer data area of this disk unit'),  !ZZZ
;    3124  1         HWQ9 = uplit (%asciz'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
;    3125  1         HWQ10 = uplit (%asciz'Also run DUP exerciser'),                            !ZZZ
;    3126  1         HWQ11 = uplit (%asciz'Write on diagnostic area'),                          !ZZZ
;    3127  1     !
;    3128  1     ! SOFTWARE DIALOG
;    3129  1     !
;    3130  1         SWQ1  = uplit (%asciz'Hard error limit'),                                                      !ZZZ
;    3131  1         SWQ2  = uplit (%asciz'Transfer limit in megabytes (0 for quick pass)'),                        !ZZZ
;    3132  1         SWQ4  = uplit (%asciz'Random seek mode'),
;    3133  1         SWQ7  = uplit (%asciz'Read-compares performed at the controller'),
;    3134  1         SWQ9  = uplit (%asciz'Write-compares performed at the controller'),
;    3135  1         SWQ10 = uplit (%asciz'Check all writes at host by reading'),
;    3136  1         SWQ11 = uplit (%asciz'User-defined data pattern'),
;    3137  1         SWQ12 = uplit (%asciz'Select pre-defined data pattern (0 for sequential selection)'),
;    3138  1         SWQ13 = uplit (%asciz'Number of words in data pattern (16 maximum)'),
;    3139  1         SWQ14 = uplit (%asciz'Pattern value (no leading zeros allowed)'),
;    3140  1         SWQ15 = uplit (%asciz'Clear statistical tables after printing'),
;    3141  1         SWQ17 = uplit (%asciz'Percentage of "Fixed Disk" operations out of total operations'),
;    3142  1         SWQ19 = uplit (%asciz'Units to be selected at random (No, implies sequential)'),
;    3143  1         SWQ20 = uplit (%asciz'Rewrite blocks when "Forced Error" detected on reads'),
;    3144  1         SWQ21 = uplit (%asciz'Halt on other hard errors (#s 31-34, 36-37, 39-45)'),                    !ZZZ
;    3145  1         SWQ22 = uplit (%asciz'Halt on soft errors (#s 50-54)'),                                        !ZZZ
;    3146  1         SWQ23 = uplit (%asciz'Halt on bad-block hard errors (#s 35, 38)'),                             !ZZZ
;    3147  1         SWQ24 = uplit (%asciz'Enter time as HHMM (example: 1305)'),                                    !ZZZ
;    3148  1     !MMM    SWQ25 = uplit (%asciz'Count each retry as a seperate soft error'),                             !ZZZ
;    3149  1         SWQ26 = uplit (%asciz'Running under the A.P.T. Monitor'),                                      !ZZZ
```

L3


```
;    3150  1          SWM1  = uplit (%asciz'The remaining questions only apply to unprotected disk units'),          !ZZZ
;    3151  1          SWQ27 = uplit (%asciz'Manufacturing Test'),                                                   !MMM
;    3152  1          SWQ28 = uplit (%asciz'Enable Host Memory (MSV11-P,L,J) Parity'),                              !MMM
;    3153  1
;    3154  1
;    3155  1          NULL  = uplit (%asciz''),
;    3156  1
;    3157  1      !++
;    3158  1      !  THE FOLLOWING DBMs ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
;    3159  1      !  RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
;    3160  1      !  FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
;    3161  1      !--
;    3162  1
;    3163  1          DBM5   = uplit (%asciz'%N%A** Drop unit %D2'),
;    3164  1          DBM12  = uplit (%asciz'%N%A** PROC_RETPKT: Conn ID %O6%A received'),
;    3165  1          DBM15  = uplit (%asciz'%N%A** Multi-drive test'),
;    3166  1          DBM18  = uplit (%asciz'%N%A** FATAL_ERROR: RETPKT not available'),
;    3167  1          DBM19  = uplit (%asciz'%N%A** FSET_OPAR: Can''t find disk %D3%A in CST %D1'),
;    3168  1          DBM20  = uplit (%asciz'%N%A** Bad conn ID %O6%A received from %O6'),
;    3169  1          DBM21  = uplit (%asciz'%N%A** Message type %O2%A received in MSCP packet'),
;    3170  1          DBM22  = uplit (%asciz'%N%A** SEQUEN: RETPKT not available'),
;    3171  1          DBM23  = uplit (%asciz'%N%A** Error in SET_CTLR_CHAR'),
;    3172  1          DBM25  = uplit (%asciz'%N%A** Ctlr timeout = %D3%A. seconds'),
;    3173  1          DBM26  = uplit (%asciz'%N%A** Error in UNIT_INIT'),
;    3174  1          DBM27  = uplit (%asciz'%N%A** UNIT_INIT: RETPKT has bad ENDCODE'),
;    3175  1          DBM28A = uplit (%asciz'%N%A** Unit size (Lo) = %D5%A.'),
;    3176  1          DBM28B = uplit (%asciz'%N%A** Unit size (Hi) = %D5%A.'),
;    3177  1          DBM29  = uplit (%asciz'%N%A** ACCESS: RETPKT has bad ENDCODE'),
;    3178  1          DBM32  = uplit (%asciz'%N%A** QIO_UNIT: CST %D1%A no unit selected'),
;    3179  1          DBM101 = uplit (%asciz'%N%A** Unit # is: %O6'),
;    3180  1          DBM104 = uplit (%asciz'%N%A** Removable disk is selected'),
;    3181  1          DBM105 = uplit (%asciz'%N%A** Fixed disk is selected'),
;    3182  1          DBM107 = uplit (%asciz'%N%A** Illegal function: %O6'),
;    3183  1          DBM108 = uplit (%asciz'%N%A** Command ref # %O6%A/%O6%A (Oct) not sent by Host'),
;    3184  1          DBM109 = uplit (%asciz'%N%A** Unknown Error Log format %O3%A received'),
;    3185  1      !   DBM110 = uplit (%asciz'%N%A** Error-Log save area full'),
;    3186  1          DBM111 = uplit (%asciz'%N%A** Op-code %O3%A, End-code %O3%A for ref # %O6%A/%O6%A (8)'),
;    3187  1          DBM112 = uplit (%asciz'%N%A** Cmd-bc %O6%A/%O6%A Rsp-bc %O6%A/%O6%A for %O6%A/%O6%A (8)'),
;    3188  1          DBM120 = uplit (%asciz'%N%A** Response already received for cmd %O6%A/%O6%A (8)'),
;    3189  1          DBM121 = uplit (%asciz'%N%A** Failure to send command after # %O6%A/%O6%A (8)'),
;    3190  1          DBM123 = uplit (%asciz'%N%A** Response has Error Log(s):'),                                   !MMM
;    3191  1          DBM125 = uplit (%asciz'%N%A** HOST MEMORY ERROR, CSR =  %O6%A'),                              !MMM
;    3192  1          DBM126 = uplit (%asciz'%N%A** HOST MEMORY ERROR, EXTENTED CSR =  %O6%A'),                     !MMM
;    3193  1          DBM127 = uplit (%asciz'%N%A** BAD BLOCK REPLACEMENT REQUEST FLAG SET'),                       !MMM
;    3194  1          DBM128 = uplit (%asciz'%N%A** ERROR DURING REPLACEMENT (BAD BLOCK) FLAG SET'),                !MMM
;    3195  1
;    3196  1
;    3197  1      !
;    3198  1      ! DROP UNIT MESSAGES
;    3199  1      !
;    3200  1          DU_MSG = uplit (%asciz'%N%AUNIT%D2%A DROPPED - '),
;    3201  1          DU_RSN = uplit (
;    3202  1                   uplit (%asciz'%AUSER COMMAND%N'),
```

```
;    3203  1                  uplit (%asciz'%CONFIGURATION ERROR%N'),
;    3204  1                  uplit (%asciz'%AINIT ERROR%N'),
;    3205  1                  uplit (%asciz'%ATRANSFER LIMIT REACHED%N'),
;    3206  1                  uplit (%asciz'%AERROR LIMIT REACHED%N'),
;    3207  1                  uplit (%asciz'%AUNRECOVERABLE DRIVE ERROR%N'),
;    3208  1                  uplit (%asciz'%AUNRECOVERABLE CONTROLLER ERROR%N'),
;    3209  1                  uplit (%asciz'%AFAILED TO COME ONLINE%N'),
;    3210  1                  uplit (%asciz'%AFAILED TO ACCESS EITHER FIRST OR LAST TRACK DURING INIT%N'),
;    3211  1                  uplit (%asciz'%ADISK WRITE PROTECTED%N'),
;    3212  1                  uplit (%asciz'%ACOMMAND TIME OUT%N')) : vector [11],
;    3213  1          !
;    3214  1          ! SYSTEM MESSAGES (PRINTF)
;    3215  1          !
;    3216  1             MSG_01 = uplit (%asciz'%N%APOWER DELAY - WAITING'),
;    3217  1             MSG_02 = uplit (%asciz'%N%AFUNCTIONAL TEST STARTED'),
;    3218  1             MSG_03 = uplit (%asciz'%N%N%AEXERCISER STARTED%N'),
;    3219  1          !
;    3220  1          ! REPORT MESSAGES (PRINTS)
;    3221  1          !
;    3222  1             RPT1  = uplit (%asciz'%N%N%AUNT DSK%S8%A# OF   # BYTES   # OF    # BYTES'),
;    3223  1             RPT2  = uplit (%asciz'%A  --HARD ERRORS-- --SOFT ERRORS--'),
;    3224  1             RPT3  = uplit (%asciz'%N%A #   #  TYPE  READS     READ   WRITES   WRITTEN'),
;    3225  1             RPT4  = uplit (%asciz'%A  SEK DAT DRV HST SEK DAT DRV HST'),
;    3226  1             RPT5  = uplit (%asciz'%N%A--- --- ----  -----  -------- ------  ---------'),
;    3227  1             RPT6  = uplit (%asciz'%A --- --- --- --- --- --- --- ---'),
;    3228  1             RPT7  = uplit (%asciz'%N%D2%D4%S2%T'),
;    3229  1             RPT8  = uplit (%asciz'%D4%Z3%D3%A,%Z3%A,%Z3'),
;    3230  1             RPT9  = uplit (%asciz'%D4%D4%D4%D4%D4%D4%D4%D4%D4'),
;    3231  1             RPT10 = uplit (%asciz'%N%A .   . CNTR     : .........   . .........'),
;    3232  1             RPT11 = uplit (%asciz'%A   .   .%D4%A   .   :   .%D4%A   .'),
;    3233  1             RPT12 = uplit (%asciz'%A   .   .%D4%A   .   :   .%D4%A   .'),
;    3234  1             RPT13 = UPLIT(%ASCIZ'%N%N%AUNIT  DISK          # OF   # BLKS       # OF    # BLKS '),
;    3235  1             RPT14 = UPLIT(%ASCIZ'%N%A  #    #    TYPE   READS     READ      WRITES  WRITTEN '),
;    3236  1             RPT15 = UPLIT(%ASCIZ'%N%A----  ---- ------- ------  ------    ------  ------ '),
;    3237  1             RPT16 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBN I/O  %D6%S3%D6%S5%D6%S3%D6'),
;    3238  1          !ZZZ    RPT17  = uplit (%asciz'%N%D2%D4%A  RD52'),
;    3239  1          !ZZZ    RPT18 = UPLIT(%ASCIZ'%N%S1%D2%S4%D2%A   DBNRD52  %D6%S3%D6%S5%D6%S3%D6'),
;    3240  1          !ZZZ    RPT19  = uplit (%asciz'%N%D2%D4%A  ????'),
;    3241  1
;    3242  1          !
;    3243  1          ! GENERAL ERROR MESSAGES
;    3244  1          !
;    3245  1          !       SYSTEM FATAL (ERRSF)
;    3246  1          !
;    3247  1             EGS_01 = uplit (%asciz'TOO MANY UNITS'),
;    3248  1             EGS_02 = uplit (%asciz'NOT ENOUGH FREE MEMORY FOR ALLOCATING READ/WRITE BUFFERS'),
;    3249  1          !
;    3250  1          !       DRIVE FATAL (ERRDF)
;    3251  1          !
;    3252  1             EGD_10 = uplit (%asciz'REGISTER EXISTENCE TEST FAILED'),
;    3253  1             EGD_11 = uplit (%asciz'VECTOR TEST FAILED'),
;    3254  1             EGD_12 = uplit (%asciz'BR LEVEL TEST FAILED'),
;    3255  1             EGD_13 = uplit (%asciz'INIT SEQUENCE FAILED'),
```

```
;    3256  1          EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
;    3257  1          EGD_15 = uplit (%asciz'ONLINE FAILED'),
;    3258  1          EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
;    3259  1          EGD_17 = uplit (%asciz'ACCESS FAILED'),
;    3260  1          EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
;    3261  1      !   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
;    3262  1          EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER'),
;    3263  1          EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND'),
;    3264  1          EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
;    3265  1          EGD_22 = uplit (%asciz'FAILED TO SEND ON-LINE COMMAND'),
;    3266  1          EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
;    3267  1          EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
;    3268  1      !
;    3269  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    3270  1      !
;    3271  1          EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
;    3272  1      !
;    3273  1      ! BASIC ERROR MESSAGES (PRINTB)
;    3274  1      !
;    3275  1      !       SYSTEM FATAL (ERRSF)
;    3276  1      !
;    3277  1          EBS_01 = uplit (%asciz'%AMORE THAN %D2%A UNITS SPECIFIED'),
;    3278  1      !
;    3279  1      !       DRIVE FATAL (ERRDF)
;    3280  1      !
;    3281  1          EBD_10 = uplit (%asciz'%A* NO RESPONSE AT ADDRESS %O6'),
;    3282  1          EBD_12 = uplit (%asciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
;    3283  1          EBD_13 = uplit (%asciz'%A* STEP %D1%A READ ERROR'),
;    3284  1          EBD_14 = uplit (%asciz'%A* BAD SA CODE FROM DRIVE %O6'),
;    3285  1          EBD_18 = uplit (%asciz'%A* DISK%D2%A WENT OFFLINE'),
;    3286  1          EBD_19 = uplit (%asciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
;    3287  1          EBD_24 = uplit (%asciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
;    3288  1      !
;    3289  1      !
;    3290  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    3291  1      !
;    3292  1      EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE'),                    !ZZZ
;    3293  1      EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),                   !ZZZ
;    3294  1      EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),                  !ZZZ
;    3295  1      EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),                   !ZZZ
;    3296  1      EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                             !ZZZ
;    3297  1      EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                          !ZZZ
;    3298  1      EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                         !ZZZ
;    3299  1      EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                          !ZZZ
;    3300  1      EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),                     !ZZZ
;    3301  1      EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                         !ZZZ
;    3302  1      EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                        !ZZZ
;    3303  1      EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),                    !ZZZ
;    3304  1      EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'),      !ZZZ
;    3305  1      !
;    3306  1          ERR_00  = uplit (%asciz'%A* DISK%D2'),
;    3307  1          ERR_COD = uplit (
;    3308  1                    uplit (%asciz'%AINVALID COMMAND'),
```

```
;    3309  1                    uplit (%asciz'%ACOMMAND ABORTED'),
;    3310  1                    uplit (%asciz'%AUNIT OFFLINE')
;    3311  1                    uplit (%asciz'%ATRANSITION TO AVAILABLE STATE'),
;    3312  1                    uplit (%asciz'%AMEDIA FORMAT ERROR'),
;    3313  1                    uplit (%asciz'%AWRITE-PROTECTED'),
;    3314  1                    uplit (%asciz'%ADEVICE COMPARE ERROR'),
;    3315  1                    uplit (%asciz'%ADATA ERROR')
;    3316  1                    uplit (%asciz'%AHOST BUFFER ACCESS ERROR'),
;    3317  1                    uplit (%asciz'%ACONTROLLER ERROR'),
;    3318  1                    uplit (%asciz'%ADRIVE ERROR'),
;    3319  1                    uplit (%asciz'%AMESSAGE FROM INTERNAL DIAGNOSTICS'),
;    3320  1                    uplit (%asciz'%AHOST COMPARE ERROR'),
;    3321  1                    uplit (%asciz'%ACOMMAND TIMEOUT'),
;    3322  1                    uplit (%asciz'%ABAD BLOCK REPLACEMENT COMPLETION')) : vector [15],                    !MMM
;    3323  1        !
;    3324  1        !       ERROR LOG MESSAGE (ERRSOFT)
;    3325  1        !
;    3326  1            ELG_00  = uplit (%asciz'%AERROR LOG MESSAGE RECEIVED:%N'),
;    3327  1            ELG_FMT = uplit (
;    3328  1                    uplit (%asciz'%A* CONTROLLER ERROR%N'),
;    3329  1                    uplit (%asciz'%A* HOST MEMORY ACCESS ERROR%N'),
;    3330  1                    uplit (%asciz'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
;    3331  1                    uplit (%asciz'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
;    3332  1                    uplit (%asciz'%A* DISK%D2%A - "SMALL DISK" ERROR%N'),
;    3333  1                    uplit (%asciz'%A* DISK%D2%A - "BAD BLOCK REPLACEMENT ATTEMPT"%N')) : vector [6],      !MMM
;    3334  1        !
;    3335  1        ! EXTENDED ERROR MESSAGES (PRINTX)
;    3336  1        !
;    3337  1            EX_SA  = uplit (%asciz'%N%A* SA: %O6'),
;    3338  1            EX_SC  = uplit (%asciz'%N%A* STATUS CODE: %O2'),
;    3339  1            EX_SBO = uplit (%asciz'%O4'),
;    3340  1            EX_SB  = uplit (%asciz'%N%A* SUB CODE: '),
;    3341  1            EX_CMD = uplit (%asciz'%N%A* COMMAND: '),
;    3342  1            EX_RD  = uplit (%asciz'%AREAD'),
;    3343  1            EX_WRT = uplit (%asciz'%AWRITE'),
;    3344  1            EX_CMP = uplit (%asciz'%A-COMPARE'),
;    3345  1            EX_ONL = uplit (%asciz'%AONLINE'),
;    3346  1            EX_ACC = uplit (%asciz'%AACCESS'),
;    3347  1            EX_OP  = uplit (%asciz'%O3'),
;    3348  1        !ZZZ        EX_BB  = uplit (%asciz'%N%A* BAD BLOCK (Host replaceable): %D5%A, (OCT %O6%A)')
;    3349  1        !ZZZ        EX_BB1 = uplit (%asciz'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A, (OCT %O6%A)'),
;    3350  1        !ZZZ        EX_BBU = uplit (%asciz'%N%A* BAD BLOCK REPORTED (Replaced): %D%A, (OCT %O6%A)'),
;    3351  1        !ZZZ        EX_LBN = uplit (%asciz'%N%A* LBN: %D5%A, (OCT %O6%A)'),
;    3352  1        !ZZZ        EX_PBN = uplit (%asciz'%N%A* PBN: %D5%A, (OCT %O6%A)'),
;    3353  1        !ZZZ        EX_LBR = uplit (%asciz'%N%A* LBN: (READ) %D5%A, (OCT %O6%A)'),
;    3354  1        !ZZZ        EX_LBW = uplit (%asciz'%N%A* LBN: (WRITE) %D5%A, (OCT %O6%A)'),
;    3355  1        !ZZZ        EX_RBN = uplit (%asciz'%N%A* REPLACEMENT BLOCK NO, %D5%A, (OCT %O6%A)'),
;    3356  1            EX_BB2 = uplit (%asciz'%N%A* BAD BLOCK: %O6%A %O6%A  (OCTAL)')                            !ZZZ
;    3357  1            EX_BB12 = uplit (%asciz'%N%A* 1ST BAD BLOCK: %O6%A %O6%A  (OCTAL)')                       !ZZZ
;    3358  1            EX_BBU2 = uplit (%asciz'%N%A* BAD BLOCK REPLACED: %O6%A %O6%A  (OCTAL)'),                 !ZZZ
;    3359  1            EX_LBN2 = uplit (%asciz'%N%A* LBN: %O6%A %O6%A  (OCTAL)'),                                !ZZZ
;    3360  1            EX_PBN2 = uplit (%asciz'%N%A* PBN: %O6%A %O6%A  (OCTAL)'),                                !ZZZ
;    3361  1            EX_LBR2 = uplit (%asciz'%N%A* LBN READ: %O6%A %O6%A  (OCTAL)'),                           !ZZZ
```

C4


```
;    3362  1          EX_LBW2 = uplit (%asciz'%N%A* LBN WRITTEN: %O6%A %O6%A  (OCTAL)'),                    !ZZZ
;    3363  1          EX_RBN2 = uplit (%asciz'%N%A* RBN: %O6%A %O6%A  (OCTAL)'),                           !ZZZ
;    3364  1          EX_CBC  = uplit (%asciz'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
;    3365  1          EX_CBR  = uplit (%asciz'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
;    3366  1          EX_CBW  = uplit (%asciz'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
;    3367  1          EX_BC   = uplit (%asciz'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
;    3368  1          EX_BD   = uplit (%asciz'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
;    3369  1          EX_BDR  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
;    3370  1          EX_BDW  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
;    3371  1          EX_RP   = uplit (%asciz'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
;    3372  1          EX_WRD  = uplit (%asciz'%A %O6'),
;    3373  1          EX_TIM  = uplit (%asciz'%N%ATIME: %Z2%A:%Z2%A HOURS%N'),                              !MMM
;    3374  1    !      EX_DUP  = uplit (%asciz'%N%A* DUP FATAL TYPE MESSAGE, ERROR CODE: %O4%A (OCTAL)'),    !MMM
;    3375  1          EX_CB = uplit (%asciz'%N%A* BAD LBN: %O6%A %O6%A  (OCTAL)'),                                  !MMM
;    3376  1
;    3377  1
;    3378  1          XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                                             !ZZZ
;    3379  1          XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A, (OCT %O6%A)'),                                   !ZZZ
;    3380  1          XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                                          !ZZZ
;    3381  1          XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),                                 !ZZZ
;    3382  1          XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),                       !ZZZ
;    3383  1
;    3384  1    !
;    3385  1    ! CONFIGURATION ERROR MESSAGES (PRINTF)
;    3386  1    !
;    3387  1          CER_01 = uplit (%asciz'%N%ADUPLICATE UNIT:%D2%A AT IP: %O6'),
;    3388  1          CER_02 = uplit (%asciz'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
;    3389  1    !
;    3390  1    ! ERROR/EVENT SUB CODES (PRINTX)
;    3391  1    !
;    3392  1          SC_SDI = uplit (%asciz'%ASPIN-DOWN IGNORED'),
;    3393  1          SC_CON = uplit (%asciz'%ASTILL CONNECTED'),
;    3394  1          SC_DUP = uplit (%asciz'%ADUPLICATE UNIT NUMBER'),
;    3395  1          SC_ONL = uplit (%asciz'%AALREADY ONLINE'),
;    3396  1          SC_SON = uplit (%asciz'%ASTILL ONLINE'),
;    3397  1          SC_INR = uplit (%asciz'%AINCOMPLETE REPLACEMENT'),                                    !MMM
;    3398  1          SC_INV = uplit (%asciz'%AINVALID RCT'),                                               !MMM
;    3399  1          SC_UNK = uplit (%asciz'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
;    3400  1          SC_VOL = uplit (%asciz'%ANO VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
;    3401  1          SC_IOP = uplit (%asciz'%AUNIT INOPERATIVE (RD51/52 write fault)'),
;    3402  1          SC_DIS = uplit (%asciz'%AUNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS'),
;    3403  1          SC_FER = uplit (%asciz'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
;    3404  1          SC_FE2 = uplit (%asciz'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER'),
;    3405  1          SC_ISH = uplit (%asciz'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
;    3406  1          SC_IS2 = uplit (%asciz'%AHEADER COMPARE ERROR (Valid header not found)'),
;    3407  1          SC_DST = uplit (%asciz'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
;    3408  1          SC_DS2 = uplit (%asciz'%ADATA SYNC NOT FOUND (Data sync timeout)'),
;    3409  1    -  -  SC_ECC = uplit (%asciz'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
;    3410  1          SC_ECD = uplit (%asciz'%AUNCORRECTABLE ECC ERROR'),
;    3411  1          SC_RCT = uplit (%asciz'%ARCT CORRUPTED'),
;    3412  1          SC_FUL = uplit (%asciz'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
;    3413  1          SC_576 = uplit (%asciz'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
;    3414  1          SC_FCT = uplit (%asciz'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
```

```
;    3415  1          SC_EC1 = uplit (%asciz'%AONE SYMBOL ECC ERROR'),
;    3416  1          SC_EC2 = uplit (%asciz'%ATWO SYMBOL ECC ERROR'),
;    3417  1          SC_EC3 = uplit (%asciz'%ATHREE SYMBOL ECC ERROR'),
;    3418  1          SC_EC4 = uplit (%asciz'%AFOUR SYMBOL ECC ERROR'),
;    3419  1          SC_EC5 = uplit (%asciz'%AFIVE SYMBOL ECC ERROR'),
;    3420  1          SC_EC6 = uplit (%asciz'%ASIX SYMBOL ECC ERROR'),
;    3421  1          SC_EC7 = uplit (%asciz'%ASEVEN SYMBOL ECC ERROR'),
;    3422  1          SC_EC8 = uplit (%asciz'%AEIGHT SYMBOL ECC ERROR'),
;    3423  1          SC_EC9 = uplit (%asciz'%ACORRECTABLE ERROR IN ECC FIELD'),
;    3424  1          SC_SWP = uplit (%asciz'%AUNIT SOFTWARE WRITE PROTECTED'),
;    3425  1          SC_HWP = uplit (%asciz'%AUNIT HARDWARE WRITE PROTECTED'),
;    3426  1          SC_SAF = uplit (%asciz'%AUNIT DATA SAFETY WRITE PROTECTED'),     !MMM
;    3427  1          SC_ODA = uplit (%asciz'%AODD TRANSFER ADDRESS'),
;    3428  1          SC_ODB = uplit (%asciz'%AODD BYTE COUNT'),
;    3429  1          SC_NXM = uplit (%asciz'%ANON-EXISTENT HOST MEMORY'),
;    3430  1          SC_PAR = uplit (%asciz'%AHOST MEMORY PARITY ERROR'),
;    3431  1          SC_CTO = uplit (%asciz'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
;    3432  1          SC_SDS = uplit (%asciz'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
;    3433  1          SC_EDC = uplit (%asciz'%A"ERROR DETECTION CODE" ERROR'),
;    3434  1          SC_IDS = uplit (%asciz'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
;    3435  1          SC_SRT = uplit (%asciz'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete)'),
;    3436  1          SC_SRI = uplit (%asciz'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
;    3437  1          SC_POE = uplit (%asciz'%APOSITION ERROR (Mis-seek)'),
;    3438  1          SC_RDY = uplit (%asciz'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
;    3439  1          SC_CLK = uplit (%asciz'%ADRIVE CLOCK DROPOUT'),
;    3440  1          SC_RSP = uplit (%asciz'%ALOST RECEIVER READY BETWEEN SECTORS'),
;    3441  1          SC_SUR = uplit (%asciz'%ADRIVE DETECTED ERROR'),
;    3442  1          SC_PSP = uplit (%asciz'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR'),
;    3443  1          SC_REP = uplit (%asciz'%ABAD BLOCK SUCCESSFULLY REPLACED')                    !MMM
;    3444  1          SC_NBB = uplit (%asciz'%ABLOCK VERIFIED OK -- NOT A BAD BLOCK'),              !MMM
;    3445  1          SC_REF = uplit (%asciz'%AREPLACEMENT FAILURE'),                               !MMM
;    3446  1      !
;    3447  1      ! CONTROLLER GENERIC ERROR CODES
;    3448  1      !
;    3449  1          CNTR_ERR = uplit (
;    3450  1                     uplit (%asciz'%ACONTROLLER TIMEOUT'),
;    3451  1                     uplit (%asciz'%AENVELOPE/PACKET READ ERROR (Parity or timeout)'),
;    3452  1                     uplit (%asciz'%AENVELOPE/PACKET WRITE ERROR (Parity or timeout)'),
;    3453  1                     uplit (%asciz'%ACONTROLLER ROM AND RAM PARITY ERROR'),
;    3454  1                     uplit (%asciz'%ACONTROLLER RAM PARITY ERROR'),
;    3455  1                     uplit (%asciz'%ACONTROLLER ROM PARITY ERROR'),
;    3456  1                     uplit (%asciz'%ARING READ ERROR (Parity or timeout)'),
;    3457  1                     uplit (%asciz'%ARING WRITE ERROR (Parity or timeout)'),
;    3458  1                     uplit (%asciz'INTERRUPT MASTER FAILURE'),
;    3459  1                     uplit (%asciz'%AHOST ACCESS TIMEOUT (Higher level protocol dependent)'),
;    3460  1                     uplit (%asciz'%ACREDIT LIMIT EXCEEDED'),
;    3461  1                     uplit (%asciz'%AQ-BUS MASTER ERROR'),
;    3462  1                     uplit (%asciz'%ACONTROLLER FATAL ERROR'),
;    3463  1                     uplit (%asciz'%AINSTRUCTION LOOP TIMEOUT'),
;    3464  1                     uplit (%asciz'%AILLEGAL VIRTUAL CIRCUIT ID'),
;    3465  1                     uplit (%asciz'%AINTERRUPT VECTOR ILLEGAL'),
;    3466  1                     uplit (%asciz'%AMAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
;    3467  1                     uplit (%asciz'%AMAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
```

```
;  3468  1                    uplit (%asciz'%ACONTROLLER RAM ERROR (Non-parity)'),
;  3469  1                    uplit (%asciz'%AINIT SEQUENCE ERROR'),
;  3470  1                    uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
;  3471  1                    uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
;  3472  1                    uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
;  3473  1            !
;  3474  1            ! RD/RX CONTROLLER DEPENDENT ERRORS CODES
;  3475  1            !
;  3476  1                RDRX_ERR = uplit (
;  3477  1                           uplit (%asciz'%AT11 CPU FAILURE'),
;  3478  1                           uplit (%asciz'%ANON-PARITY RAM ERROR'),
;  3479  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
;  3480  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
;  3481  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
;  3482  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
;  3483  1                           uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) : vector [7],
;  3484  1            !
;  3485  1            ! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
;  3486  1            !
;  3487  1                DF_MSG  = uplit (%asciz'%N%AZRQD DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;  3488  1                HRD_MSG = uplit (%asciz'%N%AZRQD HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;  3489  1                SFT_MSG = uplit (%asciz'%N%AZRQD SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06%N'),
;  3490  1                HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),
;  3491  1
;  3492  1
;  3493  1            !
;  3494  1            ! MISCELLANEOUS
;  3495  1            !
;  3496  1                SPACE4   = uplit (%asciz'%S4'),
;  3497  1                CRLF     = uplit (%asciz'%N'),
;  3498  1                DASH     = uplit (%asciz'%A -'),
;  3499  1                ASTERISK = uplit (%asciz'%A* ');
```

F4


```
;    3500  1     %sbttl 'DEFAULT HARDWARE P-TABLE'
;    3501  1
;    3502  1     !+
;    3503  1     ! THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
;    3504  1     ! THE TEST-DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
;    3505  1     ! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P-TABLES,
;    3506  1     ! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
;    3507  1     !-
;    3508  1
;    3509  1     BGNHW (DFPTBL);
;    3510  1
;    3511  1     global
;    3512  1         HWPT_IP_ADDR : word initial (INIT_IP_ADDR),          ! IP ADDRESS
;    3513  1         HWPT_VECTOR : word initial (INIT_INTR_VECT),         ! VECTOR ADDRESS
;    3514  1         HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),        ! BR LEVEL
;    3515  1         HWPT_DISK : WORD INITIAL (%o'000340'),               !PROTECT, WHOLE DISK, NO DUP   ZZZ
;    3516  1                                                              ! DK 0          ZZZ
;    3517  1         HWPTS0_LBN : word initial (0),                       ! STARTING TRACK LO ZZZ
;    3518  1         HWPTS1_LBN : word initial (0),                       ! STARTING TRACK HI ZZZ
;    3519  1         HWPTE0_LBN : word initial (%o'177777'),              ! ENDING TRACK LO   ZZZ
;    3520  1         HWPTE1_LBN : word initial (0),                       ! ENDING TRACK HI   ZZZ
;    3521  1         NAME_LO    : WORD INITIAL (%o'020040'),              !DISK TYPE          ZZZ
;    3522  1         NAME_HI    : WORD INITIAL (%o'020040');              !DISK TYPE          ZZZ
;    3523  1
;    3524  1     ENDHW;
```

```
;    3525  1     %sbttl 'SOFTWARE P-TABLE'
;    3526  1
;    3527  1     !+
;    3528  1     ! THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;    3529  1     ! PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;    3530  1     ! SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;    3531  1     ! AT RUN TIME.
;    3532  1     !-
;    3533  1
;    3534  1     BGNSW (SFPTBL);
;    3535  1
;    3536  1     global
;    3537  1         SWP_ERROR : word initial (32),                                  ! HARD ERROR LIMIT FOR DROPPING UNIT
;    3538  1         SWP_XFER : WORD INITIAL (0),                                    !XFER LIMIT. DEFAULT = QUICK PASS       !ZZZ
;    3539  1         SWP_FLAGS : word initial (SWF_RDM or SWF_CRC or SWF_HWC or SWF_FER  ! FLAGS (SEE DOCUMENTATION)     !ZZZ
;    3540  1                                   or SWF_HRD or SWF_BLK),               !                                 !ZZZ
;    3541  1         SWP_DPAT : word initial (0),                                    ! DATA PATTERN NUMBER
;    3542  1         SWP_RAT : word initial (99),                                    ! RD51/52 OPERATION RATIO
;    3543  1         SWP_TIME : word initial (0),                                    ! START TIME (HHMM)
;    3544  1     !MMM    DUPROUND : WORD INITIAL (11),              !NO OF I/Os PER DBN TEST ZZZ
;    3545  1
;    3546  1     !       THE NEXT TWO LOCATIONS SHOULD BE TOGETHER
;    3547  1
;    3548  1         SWP_UCNT : word initial (MAX_UDP_CNT),                          ! USER DATA PATTERN COUNT
;    3549  1         SWP_UDPAT : vector [MAX_UDP_CNT, word],                         ! USER DATA PATTERN
;    3550  1
;    3551  1         DUPROUND : WORD INITIAL (11),              !NO OF I/Os PER DBN TEST ZZZ
;    3552  1         MAN_TST : word initial (0),                                     ! Reduce Seek Duty cycle for Manufacturing MMM
;    3553  1         TST_PAR : word initial (1);                                     ! Enable Host Memory Parity MMM
;    3554  1
;    3555  1     ENDSW;
```

H4


```
;    3556  1     %sbttl 'PROTECTION TABLE'
;    3557  1
;    3558  1     !+
;    3559  1     ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;    3560  1     ! TO PROTECT THE LOAD MEDIA.
;    3561  1     !-
;    3562  1
;    3563  1     BGNPROT (0, -1, 6);
;    3564  1
;    3565  1     !1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;    3566  1     !2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;    3567  1     !3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;    3568  1
;    3569  1     ENDPROT;
;    3570  1     end
;    3571  1
;    3572  0     eludom


                                        .TITLE  ZRQDM1 RD/RX EXERCISER
                                        .IDENT  /V02.3/
                                        .ENABL  AMA

000000                                  .PSECT  $CODE$,  RO
000000  132     122     121     L$NAME::.ASCII  /ZRQ/
000003  104                             .ASCII  /D/
000004  000                             .BYTE   0
000005  000                             .BYTE   0
000006  000                             .BYTE   0
000007  000                             .BYTE   0
000010                          L$REV::
000010  101                             .ASCII  /A/
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$SOFT
000022  024046'                 L$HPTP::.WORD   L$HW
000024  024076'                 L$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$MREV::
000050  004                             .BYTE   4
000051  000                             .BYTE   0
000052  000000                  L$EF::  .WORD   0
```

I4


```
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$AUTO
000112  024164'                 L$PRT:: .WORD   L$PROT
000114  000001                  L$TEST::.WORD   1
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  040     040     040     P.AAA:  .ASCII  /   /
000141  040     040     040             .ASCII  /   /
000144  040     040     040             .ASCII  /   /
000147  040     040     040             .ASCII  /   /
000152  040     040     040             .ASCII  /   /
000155  040     040     040             .ASCII  /   /
000160  040     040     040             .ASCII  /   /
000163  040     040     040             .ASCII  /   /
000166  040     040     040             .ASCII  /   /
000171  040     040     040             .ASCII  /   /
000174  040     040     040             .ASCII  /   /
000177  040     040     040             .ASCII  /   /
000202  040     040     040             .ASCII  /   /
000205  040     040     000             .ASCII  /   /<00>
000210  040     040     040     P.AAB:  .ASCII  /   /
000213  040     040     040             .ASCII  /   /
000216  040     040     040             .ASCII  /   /
000221  040     040     040             .ASCII  /   /
000224  040     040     040             .ASCII  /   /
000227  040     040     040             .ASCII  /   /
000232  040     040     040             .ASCII  /   /
000235  040     040     040             .ASCII  /   /
000240  040     040     040             .ASCII  /   /
000243  040     040     040             .ASCII  /   /
000246  040     040     040             .ASCII  /   /
000251  040     040     040             .ASCII  /   /
000254  040     040     040             .ASCII  /   /
```

J4


```
000257  040  040  000          .ASCII  /  /<00>
000262  040  040  040  P.AAC:   .ASCII  /  /
000265  040  040  040          .ASCII  /  /
000270  040  040  040          .ASCII  /  /
000273  040  040  040          .ASCII  /  /
000276  040  040  040          .ASCII  /  /
000301  040  040  040          .ASCII  /  /
000304  040  040  040          .ASCII  /  /
000307  040  040  040          .ASCII  /  /
000312  040  040  040          .ASCII  /  /
000315  040  040  040          .ASCII  /  /
000320  040  040  040          .ASCII  /  /
000323  040  040  040          .ASCII  /  /
000326  040  040  040          .ASCII  /  /
000331  040  040  000          .ASCII  /  /<00>
000334  040  040  040  P.AAD:   .ASCII  /  /
000337  040  040  040          .ASCII  /  /
000342  040  040  040          .ASCII  /  /
000345  040  040  040          .ASCII  /  /
000350  040  040  040          .ASCII  /  /
000353  040  040  040          .ASCII  /  /
000356  040  040  040          .ASCII  /  /
000361  040  040  040          .ASCII  /  /
000364  040  040  040          .ASCII  /  /
000367  040  040  040          .ASCII  /  /
000372  040  040  040          .ASCII  /  /
000375  040  040  040          .ASCII  /  /
000400  040  040  040          .ASCII  /  /
000403  040  040  000          .ASCII  /  /<00>
000406  040  040  040  P.AAE:   .ASCII  /  /
000411  040  040  040          .ASCII  /  /
000414  040  040  040          .ASCII  /  /
000417  040  040  040          .ASCII  /  /
000422  040  040  040          .ASCII  /  /
000425  040  040  040          .ASCII  /  /
000430  040  040  040          .ASCII  /  /
000433  040  040  040          .ASCII  /  /
000436  040  040  040          .ASCII  /  /
000441  040  040  040          .ASCII  /  /
000444  040  040  040          .ASCII  /  /
000447  040  040  040          .ASCII  /  /
000452  040  040  040          .ASCII  /  /
000455  040  040  000          .ASCII  /  /<00>
000460  111  120  040  P.AAF:   .ASCII  /IP /
000463  141  144  144          .ASCII  /add/
000466  162  145  163          .ASCII  /res/
000471  163  000  000          .ASCII  /s/<00><00>
000474  126  145  143  P.AAG:   .ASCII  /Vec/
000477  164  157  162          .ASCII  /tor/
000502  000  000               .ASCII  <00><00>
000504  102  122  040  P.AAH:   .ASCII  /BR /
000507  114  145  166          .ASCII  /Lev/
000512  145  154  040          .ASCII  /el /
```

K4


```
000515  133  165  163                    .ASCII  /[us/
000520  165  141  154                    .ASCII  /ual/
000523  154  171  040                    .ASCII  /ly /
000526  064  055  122                    .ASCII  /4-R/
000531  121  104  130                    .ASCII  /QDX/
000534  040  065  055                    .ASCII  / 5-/
000537  122  125  130                    .ASCII  /RUX/
000542  065  060  135                    .ASCII  /50]/
000545  000                             .ASCII  <00>
000546  104  162  151          P.AAI:    .ASCII  /Dri/
000551  166  145  040                    .ASCII  /ve /
000554  156  165  155                    .ASCII  /num/
000557  142  145  162                    .ASCII  /ber/
000562  000  000                        .ASCII  <00><00>
000564  124  145  163          P.AAJ:    .ASCII  /Tes/
000567  164  040  145                    .ASCII  /t e/
000572  156  164  151                    .ASCII  /nti/
000575  162  145  040                    .ASCII  /re /
000600  143  165  163                    .ASCII  /cus/
000603  164  157  155                    .ASCII  /tom/
000606  145  162  040                    .ASCII  /er /
000611  141  162  145                    .ASCII  /are/
000614  141  040  157                    .ASCII  /a o/
000617  146  040  164                    .ASCII  /f t/
000622  150  151  163                    .ASCII  /his/
000625  040  144  151                    .ASCII  / di/
000630  163  153  000                    .ASCII  /sk/<00>
000633  000                             .ASCII  <00>
000634  114  157  167          P.AAK:    .ASCII  /Low/
000637  145  162  040                    .ASCII  /er /
000642  157  143  164                    .ASCII  /oct/
000645  141  154  040                    .ASCII  /al /
000650  167  157  162                    .ASCII  /wor/
000653  144  040  157                    .ASCII  /d o/
000656  146  040  142                    .ASCII  /f b/
000661  145  147  151                    .ASCII  /egi/
000664  156  156  151                    .ASCII  /nni/
000667  156  147  040                    .ASCII  /ng /
000672  114  102  116                    .ASCII  /LBN/
000675  040  141  144                    .ASCII  / ad/
000700  144  162  145                    .ASCII  /dre/
000703  163  163  000                    .ASCII  /ss/<00>
000706  110  151  147          P.AAL:    .ASCII  /Hig/
000711  150  145  162                    .ASCII  /her/
000714  040  157  143                    .ASCII  / cc/
000717  164  141  154                    .ASCII  /tal/
000722  040  167  157                    .ASCII  / wo/
000725  162  144  040                    .ASCII  /rd /
000730  157  146  040                    .ASCII  /of /
000733  142  145  147                    .ASCII  /beg/
000736  151  156  156                    .ASCII  /inn/
000741  151  156  147                    .ASCII  /ing/
000744  040  114  102                    .ASCII  / LB/
```

L4


```
000747  116  040  141                    .ASCII  /N a/
000752  144  144  162                    .ASCII  /ddr/
000755  145  163  163                    .ASCII  /ess/
000760  000  000                         .ASCII  <00><00>
000762  114  157  167          P.AAM:    .ASCII  /Low/
000765  145  162  040                    .ASCII  /er /
000770  157  143  164                    .ASCII  /oct/
000773  141  154  040                    .ASCII  /al /
000776  167  157  162                    .ASCII  /wor/
001001  144  040  157                    .ASCII  /d o/
001004  146  040  145                    .ASCII  /f e/
001007  156  144  151                    .ASCII  /ndi/
001012  156  147  040                    .ASCII  /ng /
001015  114  102  116                    .ASCII  /LBN/
001020  040  141  144                    .ASCII  / ad/
001023  144  162  145                    .ASCII  /dre/
001026  163  163  000                    .ASCII  /ss/<00>
001031  000                             .ASCII  <00>
001032  110  151  147          P.AAN:    .ASCII  /Hig/
001035  150  145  162                    .ASCII  /her/
001040  040  157  143                    .ASCII  / oc/
001043  164  141  154                    .ASCII  /tal/
001046  040  167  157                    .ASCII  / wo/
001051  162  144  040                    .ASCII  /rd /
001054  157  146  040                    .ASCII  /of /
001057  145  156  144                    .ASCII  /end/
001062  151  156  147                    .ASCII  /ing/
001065  040  114  102                    .ASCII  / LB/
001070  116  040  141                    .ASCII  /N a/
001073  144  144  162                    .ASCII  /ddr/
001076  145  163  163                    .ASCII  /ess/
001101  000                             .ASCII  <00>
001102  127  162  151          P.AAO:    .ASCII  /Wri/
001105  164  145  040                    .ASCII  /te /
001110  157  156  040                    .ASCII  /on /
001113  143  165  163                    .ASCII  /cus/
001116  164  157  155                    .ASCII  /tom/
001121  145  162  040                    .ASCII  /er /
001124  144  141  164                    .ASCII  /dat/
001127  141  040  141                    .ASCII  /a a/
001132  162  145  141                    .ASCII  /rea/
001135  040  157  146                    .ASCII  / of/
001140  040  164  150                    .ASCII  / th/
001143  151  163  040                    .ASCII  /is /
001146  144  151  163                    .ASCII  /dis/
001151  153  040  165                    .ASCII  /k u/
001154  156  151  164                    .ASCII  /nit/
001157  000                             .ASCII  <00>
001160  052  052  040          P.AAP:    .ASCII  /** /
001163  127  101  122                    .ASCII  /WAR/
001166  116  111  116                    .ASCII  /NIN/
001171  107  040  055                    .ASCII  /G -/
001174  040  103  125                    .ASCII  / CU/
```

```
001177  123  124  117                .ASCII  /STO/
001202  115  105  122                .ASCII  /MER/
001205  040  104  101                .ASCII  / DA/
001210  124  101  040                .ASCII  /TA /
001213  101  122  105                .ASCII  /ARE/
001216  101  040  115                .ASCII  /A M/
001221  101  131  040                .ASCII  /AY /
001224  102  105  040                .ASCII  /BE /
001227  117  126  105                .ASCII  /OVE/
001232  122  127  122                .ASCII  /RWR/
001235  111  124  124                .ASCII  /ITT/
001240  105  116  041                .ASCII  /EN!/
001243  040  056  056                .ASCII  / ../
001246  056  040  103                .ASCII  /. C/
001251  117  116  106                .ASCII  /ONF/
001254  111  122  115                .ASCII  /IRM/
001257  000                        .ASCII  <00>
001260  101  154  163  P.AAQ:        .ASCII  /Als/
001263  157  040  162                .ASCII  /o r/
001266  165  156  040                .ASCII  /un /
001271  104  125  120                .ASCII  /DUP/
001274  040  145  170                .ASCII  / ex/
001277  145  162  143                .ASCII  /erc/
001302  151  163  145                .ASCII  /ise/
001305  162  000  000                .ASCII  /r/<00><00>
001310  127  162  151  P.AAR:        .ASCII  /Wri/
001313  164  145  040                .ASCII  /te /
001316  157  156  040                .ASCII  /on /
001321  144  151  141                .ASCII  /dia/
001324  147  156  157                .ASCII  /gno/
001327  163  164  151                .ASCII  /sti/
001332  143  040  141                .ASCII  /c a/
001335  162  145  141                .ASCII  /rea/
001340  000  000                    .ASCII  <00><00>
001342  110  141  162  P.AAS:        .ASCII  /Her/
001345  144  040  145                .ASCII  /d e/
001350  162  162  157                .ASCII  /rro/
001353  162  040  154                .ASCII  /r l/
001356  151  155  151                .ASCII  /imi/
001361  164  000  000                .ASCII  /t/<00><00>
001364  124  162  141  P.AAT:        .ASCII  /Tra/
001367  156  163  146                .ASCII  /nsf/
001372  145  162  040                .ASCII  /er /
001375  154  151  155                .ASCII  /lim/
001400  151  164  040                .ASCII  /it /
001403  151  156  040                .ASCII  /in /
001406  155  145  147                .ASCII  /meg/
001411  141  142  171                .ASCII  /aby/
001414  164  145  163                .ASCII  /tes/
001417  040  050  060                .ASCII  / (0/
001422  040  146  157                .ASCII  / fo/
001425  162  040  161                .ASCII  /r q/
001430  165  151  143                .ASCII  /uic/
```

```
001433  153  040  160                    .ASCII  /k p/
001436  141  163  163                    .ASCII  /ass/
001441  051  000  000                    .ASCII  /)/<00><00>
001444  122  141  156          P.AAU:    .ASCII  /Ran/
001447  144  157  155                    .ASCII  /dom/
001452  040  163  145                    .ASCII  / se/
001455  145  153  040                    .ASCII  /ek /
001460  155  157  144                    .ASCII  /mod/
001463  145  000  000                    .ASCII  /e/<00><00>
001466  122  145  141          P.AAV:    .ASCII  /Rea/
001471  144  055  143                    .ASCII  /d-c/
001474  157  155  160                    .ASCII  /omp/
001477  141  162  145                    .ASCII  /are/
001502  163  040  160                    .ASCII  /s p/
001505  145  162  146                    .ASCII  /erf/
001510  157  162  155                    .ASCII  /orm/
001513  145  144  040                    .ASCII  /ed /
001516  141  164  040                    .ASCII  /at /
001521  164  150  145                    .ASCII  /the/
001524  040  143  157                    .ASCII  / co/
001527  156  164  162                    .ASCII  /ntr/
001532  157  154  154                    .ASCII  /oll/
001535  145  162  000                    .ASCII  /er/<00>
001540  127  162  151          P.AAW:    .ASCII  /Wri/
001543  164  145  055                    .ASCII  /te-/
001546  143  157  155                    .ASCII  /com/
001551  160  141  162                    .ASCII  /par/
001554  145  163  040                    .ASCII  /es /
001557  160  145  162                    .ASCII  /per/
001562  146  157  162                    .ASCII  /for/
001565  155  145  144                    .ASCII  /med/
001570  040  141  164                    .ASCII  / at/
001573  040  164  150                    .ASCII  / th/
001576  145  040  143                    .ASCII  /e c/
001601  157  156  164                    .ASCII  /ont/
001604  162  157  154                    .ASCII  /rol/
001607  154  145  162                    .ASCII  /ler/
001612  000  000                         .ASCII  <00><00>
001614  103  150  145          P.AAX:    .ASCII  /Che/
001617  143  153  040                    .ASCII  /ck /
001622  141  154  154                    .ASCII  /all/
001625  040  167  162                    .ASCII  / wr/
001630  151  164  145                    .ASCII  /ite/
001633  163  040  141                    .ASCII  /s a/
001636  164  040  150                    .ASCII  /t h/
001641  157  163  164                    .ASCII  /ost/
001644  040  142  171                    .ASCII  / by/
001647  040  162  145                    .ASCII  / re/
001652  141  144  151                    .ASCII  /adi/
001655  156  147  000                    .ASCII  /ng/<00>
001660  125  163  145          P.AAY:    .ASCII  /Use/
001663  162  055  144                    .ASCII  /r-d/
001666  145  146  151                    .ASCII  /efi/
```

```
001671  156  145  144          .ASCII  /ned/
001674  040  144  141          .ASCII  / da/
001677  164  141  040          .ASCII  /ta /
001702  160  141  164          .ASCII  /pat/
001705  164  145  162          .ASCII  /ter/
001710  156  000              .ASCII  /n/<00>
001712  123  145  154  P.AAZ:  .ASCII  /Sel/
001715  145  143  164          .ASCII  /ect/
001720  040  160  162          .ASCII  / pr/
001723  145  055  144          .ASCII  /e-d/
001726  145  146  151          .ASCII  /efi/
001731  156  145  144          .ASCII  /ned/
001734  040  144  141          .ASCII  / da/
001737  164  141  040          .ASCII  /ta /
001742  160  141  164          .ASCII  /pat/
001745  164  145  162          .ASCII  /ter/
001750  156  040  050          .ASCII  /n (/
001753  060  040  146          .ASCII  /0 f/
001756  157  162  040          .ASCII  /or /
001761  163  145  161          .ASCII  /seq/
001764  165  145  156          .ASCII  /uen/
001767  164  151  141          .ASCII  /tia/
001772  154  040  163          .ASCII  /l s/
001775  145  154  145          .ASCII  /ele/
002000  143  164  151          .ASCII  /cti/
002003  157  156  051          .ASCII  /on)/
002006  000  000              .ASCII  <00><00>
002010  116  165  155  P.ABA:  .ASCII  /Num/
002013  142  145  162          .ASCII  /ber/
002016  040  157  146          .ASCII  / of/
002021  040  167  157          .ASCII  / wo/
002024  162  144  163          .ASCII  /rds/
002027  040  151  156          .ASCII  / in/
002032  040  144  141          .ASCII  / da/
002035  164  141  040          .ASCII  /ta /
002040  160  141  164          .ASCII  /pat/
002043  164  145  162          .ASCII  /ter/
002046  156  040  050          .ASCII  /n (/
002051  061  066  040          .ASCII  /16 /
002054  155  141  170          .ASCII  /max/
002057  151  155  165          .ASCII  /imu/
002062  155  051  000          .ASCII  /m)/<00>
002065  000                  .ASCII  <00>
002066  120  141  164  P.ABB:  .ASCII  /Pat/
002071  164  145  162          .ASCII  /ter/
002074  156  040  166          .ASCII  /n v/
002077  141  154  165          .ASCII  /alu/
002102  145  040  050          .ASCII  /e (/
002105  156  157  040          .ASCII  /no /
002110  154  145  141          .ASCII  /lea/
002113  144  151  156          .ASCII  /din/
002116  147  040  172          .ASCII  /g z/
002121  145  162  157          .ASCII  /ero/
```

```
002124  163  040  141                    .ASCII  /s a/
002127  154  154  157                    .ASCII  /llo/
002132  167  145  144                    .ASCII  /wed/
002135  051  000  000                    .ASCII  /)/<00><00>
002140  103  154  145            P.ABC:  .ASCII  /Cle/
002143  141  162  040                    .ASCII  /ar /
002146  163  164  141                    .ASCII  /sta/
002151  164  151  163                    .ASCII  /tis/
002154  164  151  143                    .ASCII  /tic/
002157  141  154  040                    .ASCII  /al /
002162  164  141  142                    .ASCII  /tab/
002165  154  145  163                    .ASCII  /les/
002170  040  141  146                    .ASCII  / af/
002173  164  145  162                    .ASCII  /ter/
002176  040  160  162                    .ASCII  / pr/
002201  151  156  164                    .ASCII  /int/
002204  151  156  147                    .ASCII  /ing/
002207  000                            .ASCII  <00>
002210  120  145  162            P.ABD:  .ASCII  /Per/
002213  143  145  156                    .ASCII  /cen/
002216  164  141  147                    .ASCII  /tag/
002221  145  040  157                    .ASCII  /e o/
002224  146  040  042                    .ASCII  /f "/
002227  106  151  170                    .ASCII  /Fix/
002232  145  144  040                    .ASCII  /ed /
002235  104  151  163                    .ASCII  /Dis/
002240  153  042  040                    .ASCII  /k" /
002243  157  160  145                    .ASCII  /ope/
002246  162  141  164                    .ASCII  /rat/
002251  151  157  156                    .ASCII  /ion/
002254  163  040  157                    .ASCII  /s o/
002257  165  164  040                    .ASCII  /ut /
002262  157  146  040                    .ASCII  /of /
002265  164  157  164                    .ASCII  /tot/
002270  141  154  040                    .ASCII  /al /
002273  157  160  145                    .ASCII  /ope/
002276  162  141  164                    .ASCII  /rat/
002301  151  157  156                    .ASCII  /ion/
002304  163  000                        .ASCII  /s/<00>
002306  125  156  151            P.ABE:  .ASCII  /Uni/
002311  164  163  040                    .ASCII  /ts /
002314  164  157  040                    .ASCII  /to /
002317  142  145  040                    .ASCII  /be /
002322  163  145  154                    .ASCII  /sel/
002325  145  143  164                    .ASCII  /ect/
002330  145  144  040                    .ASCII  /ed /
002333  141  164  040                    .ASCII  /at /
002336  162  141  156                    .ASCII  /ran/
002341  144  157  155                    .ASCII  /dom/
002344  040  050  116                    .ASCII  / (N/
002347  157  054  040                    .ASCII  /o, /
002352  151  155  160                    .ASCII  /imp/
002355  154  151  145                    .ASCII  /lie/
```

```
002360  163  040  163          .ASCII  /s s/
002363  145  161  165          .ASCII  /equ/
002366  145  156  164          .ASCII  /ent/
002371  151  141  154          .ASCII  /ial/
002374  051  000               .ASCII  /)/<00>
002376  122  145  167  P.ABF:   .ASCII  /Rew/
002401  162  151  164          .ASCII  /rit/
002404  145  040  142          .ASCII  /e b/
002407  154  157  143          .ASCII  /loc/
002412  153  163  040          .ASCII  /ks /
002415  167  150  145          .ASCII  /whe/
002420  156  040  042          .ASCII  /n "/
002423  106  157  162          .ASCII  /For/
002426  143  145  144          .ASCII  /ced/
002431  040  105  162          .ASCII  / Er/
002434  162  157  162          .ASCII  /ror/
002437  042  040  144          .ASCII  /" d/
002442  145  164  145          .ASCII  /ete/
002445  143  164  145          .ASCII  /cte/
002450  144  040  157          .ASCII  /d o/
002453  156  040  162          .ASCII  /n r/
002456  145  141  144          .ASCII  /ead/
002461  163  000  000          .ASCII  /s/<00><00>
002464  110  141  154  P.ABG:   .ASCII  /Hal/
002467  164  040  157          .ASCII  /t o/
002472  156  040  157          .ASCII  /n o/
002475  164  150  145          .ASCII  /the/
002500  162  040  150          .ASCII  /r h/
002503  141  162  144          .ASCII  /ard/
002506  040  145  162          .ASCII  / er/
002511  162  157  162          .ASCII  /ror/
002514  163  040  050          .ASCII  /s (/
002517  043  163  040          .ASCII  /#s /
002522  063  061  055          .ASCII  /31-/
002525  063  064  054          .ASCII  /34,/
002530  040  063  066          .ASCII  / 36/
002533  055  063  067          .ASCII  /-37/
002536  054  040  063          .ASCII  /, 3/
002541  071  055  064          .ASCII  /9-4/
002544  065  051  000          .ASCII  /5)/<00>
002547  000                    .ASCII  <00>
002550  110  141  154  P.ABH:   .ASCII  /Hal/
002553  164  040  157          .ASCII  /t o/
002556  156  040  163          .ASCII  /n s/
002561  157  146  164          .ASCII  /oft/
002564  040  145  162          .ASCII  / er/
002567  162  157  162          .ASCII  /ror/
002572  163  040  050          .ASCII  /s (/
002575  043  163  040          .ASCII  /#s /
002600  065  060  055          .ASCII  /50-/
002603  065  064  051          .ASCII  /54)/
002606  000  000               .ASCII  <00><00>
002610  110  141  154  P.ABI:   .ASCII  /Hal/
```

```
002613  164  040  157                .ASCII  /t o/
002616  156  040  142                .ASCII  /n b/
002621  141  144  055                .ASCII  /ad-/
002624  142  154  157                .ASCII  /blo/
002627  143  153  040                .ASCII  /ck /
002632  150  141  162                .ASCII  /har/
002635  144  040  145                .ASCII  /d e/
002640  162  162  157                .ASCII  /rro/
002643  162  163  040                .ASCII  /rs /
002646  050  043  163                .ASCII  /(#s/
002651  040  063  065                .ASCII  / 35/
002654  054  040  063                .ASCII  /, 3/
002657  070  051  000                .ASCII  /8)/<00>
002662  105  156  164  P.ABJ:        .ASCII  /Ent/
002665  145  162  040                .ASCII  /er /
002670  164  151  155                .ASCII  /tim/
002673  145  040  141                .ASCII  /e a/
002676  163  040  110                .ASCII  /s H/
002701  110  115  115                .ASCII  /HMM/
002704  040  050  145                .ASCII  / (e/
002707  170  141  155                .ASCII  /xam/
002712  160  154  145                .ASCII  /ple/
002715  072  040  061                .ASCII  /: 1/
002720  063  060  065                .ASCII  /305/
002723  051  000  000                .ASCII  /)/<00><00>
002726  122  165  156  P.ABK:        .ASCII  /Run/
002731  156  151  156                .ASCII  /nin/
002734  147  040  165                .ASCII  /g u/
002737  156  144  145                .ASCII  /nde/
002742  162  040  164                .ASCII  /r t/
002745  150  145  040                .ASCII  /he /
002750  101  056  120                .ASCII  /A.P/
002753  056  124  056                .ASCII  /.T./
002756  040  115  157                .ASCII  / Mo/
002761  156  151  164                .ASCII  /nit/
002764  157  162  000                .ASCII  /or/<00>
002767  000                         .ASCII  <00>
002770  124  150  145  P.ABL:        .ASCII  /The/
002773  040  162  145                .ASCII  / re/
002776  155  141  151                .ASCII  /mai/
003001  156  151  156                .ASCII  /nin/
003004  147  040  161                .ASCII  /g q/
003007  165  145  163                .ASCII  /ues/
003012  164  151  157                .ASCII  /tio/
003015  156  163  040                .ASCII  /ns /
003020  157  156  154                .ASCII  /onl/
003023  171  040  141                .ASCII  /y a/
003026  160  160  154                .ASCII  /ppl/
003031  171  040  164                .ASCII  /y t/
003034  157  040  165                .ASCII  /o u/
003037  156  160  162                .ASCII  /npr/
003042  157  164  145                .ASCII  /ote/
003045  143  164  145                .ASCII  /cte/
```

F5


```
003050  144  040  144                .ASCII  /d d/
003053  151  163  153                .ASCII  /isk/
003056  040  165  156                .ASCII  / un/
003061  151  164  163                .ASCII  /its/
003064  000  000                     .ASCII  <00><00>
003066  115  141  156          P.ABM: .ASCII  /Man/
003071  165  146  141                .ASCII  /ufa/
003074  143  164  165                .ASCII  /ctu/
003077  162  151  156                .ASCII  /rin/
003102  147  040  124                .ASCII  /g T/
003105  145  163  164                .ASCII  /est/
003110  000  000                     .ASCII  <00><00>
003112  105  156  141          P.ABN: .ASCII  /Ena/
003115  142  154  145                .ASCII  /ble/
003120  040  110  157                .ASCII  / Ho/
003123  163  164  040                .ASCII  /st /
003126  115  145  155                .ASCII  /Mem/
003131  157  162  171                .ASCII  /ory/
003134  040  050  115                .ASCII  / (M/
003137  123  126  061                .ASCII  /SV1/
003142  061  055  120                .ASCII  /1-P/
003145  054  114  054                .ASCII  /,L,/
003150  112  051  040                .ASCII  /J) /
003153  120  141  162                .ASCII  /Par/
003156  151  164  171                .ASCII  /ity/
003161  000                         .ASCII  <00>
003162  000  000               P.ABO: .ASCII  <00><00>
003164  045  116  045          P.ABP: .ASCII  /%N%/
003167  101  052  052                .ASCII  /A**/
003172  040  104  162                .ASCII  / Dr/
003175  157  160  040                .ASCII  /op /
003200  165  156  151                .ASCII  /uni/
003203  164  040  045                .ASCII  /t %/
003206  104  062  000                .ASCII  /D2/<00>
003211  000                         .ASCII  <00>
003212  045  116  045          P.ABQ: .ASCII  /%N%/
003215  101  052  052                .ASCII  /A**/
003220  040  120  122                .ASCII  / PR/
003223  117  103  137                .ASCII  /OC_/
003226  122  105  124                .ASCII  /RET/
003231  120  113  124                .ASCII  /PKT/
003234  072  040  103                .ASCII  /: C/
003237  157  156  156                .ASCII  /onn/
003242  040  111  104                .ASCII  / ID/
003245  040  045  117                .ASCII  / %O/
003250  066  045  101                .ASCII  /6%A/
003253  040  162  145                .ASCII  / re/
003256  143  145  151                .ASCII  /cei/
003261  166  145  144                .ASCII  /ved/
003264  000  000                     .ASCII  <00><00>
003266  045  116  045          P.ABR: .ASCII  /%N%/
003271  101  052  052                .ASCII  /A**/
003274  040  115  165                .ASCII  / Mu/
```

```
003277  154  164  151                    .ASCII  /lti/
003302  055  144  162                    .ASCII  /-dr/
003305  151  166  145                    .ASCII  /ive/
003310  040  164  145                    .ASCII  / te/
003313  163  164  000                    .ASCII  /st/<00>
003316  045  116  045            P.ABS:  .ASCII  /%N%/
003321  101  052  052                    .ASCII  /A**/
003324  040  106  101                    .ASCII  / FA/
003327  124  101  114                    .ASCII  /TAL/
003332  137  105  122                    .ASCII  / ER/
003335  122  117  122                    .ASCII  /ROR/
003340  072  040  122                    .ASCII  /: R/
003343  105  124  120                    .ASCII  /ETP/
003346  113  124  040                    .ASCII  /KT /
003351  156  157  164                    .ASCII  /not/
003354  040  141  166                    .ASCII  / av/
003357  141  151  154                    .ASCII  /ail/
003362  141  142  154                    .ASCII  /abl/
003365  145  000  000                    .ASCII  /e/<00><00>
003370  045  116  045            P.ABT:  .ASCII  /%N%/
003373  101  052  052                    .ASCII  /A**/
003376  040  106  123                    .ASCII  / FS/
003401  105  124  137                    .ASCII  /ET_/
003404  125  120  101                    .ASCII  /UPA/
003407  122  072  040                    .ASCII  /R: /
003412  103  141  156                    .ASCII  /Can/
003415  047  164  040                    .ASCII  /'t /
003420  146  151  156                    .ASCII  /fin/
003423  144  040  144                    .ASCII  /d d/
003426  151  163  153                    .ASCII  /isk/
003431  040  045  104                    .ASCII  / %D/
003434  063  045  101                    .ASCII  /3%A/
003437  040  151  156                    .ASCII  / in/
003442  040  103  123                    .ASCII  / CS/
003445  124  040  045                    .ASCII  /T %/
003450  104  061  000                    .ASCII  /D1/<00>
003453  000                            .ASCII  <00>
003454  045  116  045            P.ABU:  .ASCII  /%N%/
003457  101  052  052                    .ASCII  /A**/
003462  040  102  141                    .ASCII  / Ba/
003465  144  040  143                    .ASCII  /d c/
003470  157  156  156                    .ASCII  /onn/
003473  040  111  104                    .ASCII  / ID/
003476  040  045  117                    .ASCII  / %O/
003501  066  045  101                    .ASCII  /6%A/
003504  040  162  145                    .ASCII  / re/
003507  143  145  151                    .ASCII  /cei/
003512  166  145  144                    .ASCII  /ved/
003515  040  146  162                    .ASCII  / fr/
003520  157  155  040                    .ASCII  /om /
003523  045  117  066                    .ASCII  /%O6/
003526  000  000                        .ASCII  <00><00>
003530  045  116  045            P.ABV:  .ASCII  /%N%/
```

```
003533  101  052  052          .ASCII  /A**/
003536  040  115  145          .ASCII  / Me/
003541  163  163  141          .ASCII  /ssa/
003544  147  145  040          .ASCII  /ge /
003547  164  171  160          .ASCII  /typ/
003552  145  040  045          .ASCII  /e %/
003555  117  062  045          .ASCII  /02%/
003560  101  040  162          .ASCII  /A r/
003563  145  143  145          .ASCII  /ece/
003566  151  166  145          .ASCII  /ive/
003571  144  040  151          .ASCII  /d i/
003574  156  040  115          .ASCII  /n M/
003577  123  103  120          .ASCII  /SCP/
003602  040  160  141          .ASCII  / pa/
003605  143  153  145          .ASCII  /cke/
003610  164  000               .ASCII  /t/<00>
003612  045  116  045  P.ABW:   .ASCII  /%N%/
003615  101  052  052          .ASCII  /A**/
003620  040  123  105          .ASCII  / SE/
003623  121  125  105          .ASCII  /QUE/
003626  116  072  040          .ASCII  /N: /
003631  122  105  124          .ASCII  /RET/
003634  120  113  124          .ASCII  /PKT/
003637  040  156  157          .ASCII  / no/
003642  164  040  141          .ASCII  /t a/
003645  166  141  151          .ASCII  /vai/
003650  154  141  142          .ASCII  /lab/
003653  154  145  000          .ASCII  /le/<00>
003656  045  116  045  P.ABX:   .ASCII  /%N%/
003661  101  052  052          .ASCII  /A**/
003664  040  105  162          .ASCII  / Er/
003667  162  157  162          .ASCII  /ror/
003672  040  151  156          .ASCII  / in/
003675  040  123  105          .ASCII  / SE/
003700  124  137  103          .ASCII  /T_C/
003703  124  114  122          .ASCII  /TLR/
003706  137  103  110          .ASCII  /_CH/
003711  101  122  000          .ASCII  /AR/<00>
003714  045  116  045  P.ABY:   .ASCII  /%N%/
003717  101  052  052          .ASCII  /A**/
003722  040  103  164          .ASCII  / Ct/
003725  154  162  040          .ASCII  /lr /
003730  164  151  155          .ASCII  /tim/
003733  145  157  165          .ASCII  /eou/
003736  164  040  075          .ASCII  /t =/
003741  040  045  104          .ASCII  / %D/
003744  063  045  101          .ASCII  /3%A/
003747  056  040  163          .ASCII  /. s/
003752  145  143  157          .ASCII  /eco/
003755  156  144  163          .ASCII  /nds/
003760  000  000               .ASCII  <00><00>
003762  045  116  045  P.ABZ:   .ASCII  /%N%/
003765  101  052  052          .ASCII  /A**/
```

```
003770  040  105  162                    .ASCII  / Er/
003773  162  157  162                    .ASCII  /ror/
003776  040  151  156                    .ASCII  / in/
004001  040  125  116                    .ASCII  / UN/
004004  111  124  137                    .ASCII  /IT_/
004007  111  116  111                    .ASCII  /INI/
004012  124  000                         .ASCII  /T/<00>
004014  045  116  045          P.ACA:    .ASCII  /%N%/
004017  101  052  052                    .ASCII  /A**/
004022  040  125  116                    .ASCII  / UN/
004025  111  124  137                    .ASCII  /IT_/
004030  111  116  111                    .ASCII  /INI/
004033  124  072  040                    .ASCII  /T: /
004036  122  105  124                    .ASCII  /RET/
004041  120  113  124                    .ASCII  /PKT/
004044  040  150  141                    .ASCII  / ha/
004047  163  040  142                    .ASCII  /s b/
004052  141  144  040                    .ASCII  /ad /
004055  105  116  104                    .ASCII  /END/
004060  103  117  104                    .ASCII  /COD/
004063  105  000  000                    .ASCII  /E/<00><00>
004066  045  116  045          P.ACB:    .ASCII  /%N%/
004071  101  052  052                    .ASCII  /A**/
004074  040  125  156                    .ASCII  / Un/
004077  151  164  040                    .ASCII  /it /
004102  163  151  172                    .ASCII  /siz/
004105  145  040  050                    .ASCII  /e (/
004110  114  157  051                    .ASCII  /Lo)/
004113  040  075  040                    .ASCII  / = /
004116  045  104  065                    .ASCII  /%D5/
004121  045  101  056                    .ASCII  /%A./
004124  000  000                         .ASCII  <00><00>
004126  045  116  045          P.ACC:    .ASCII  /%N%/
004131  101  052  052                    .ASCII  /A**/
004134  040  125  156                    .ASCII  / Un/
004137  151  164  040                    .ASCII  /it /
004142  163  151  172                    .ASCII  /siz/
004145  145  040  050                    .ASCII  /e (/
004150  110  151  051                    .ASCII  /Hi)/
004153  040  075  040                    .ASCII  / = /
004156  045  104  065                    .ASCII  /%D5/
004161  045  101  056                    .ASCII  /%A./
004164  000  000                         .ASCII  <00><00>
004166  045  116  045          P.ACD:    .ASCII  /%N%/
004171  101  052  052                    .ASCII  /A**/
004174  040  101  103                    .ASCII  / AC/
004177  103  105  123                    .ASCII  /CES/
004202  123  072  040                    .ASCII  /S: /
004205  122  105  124                    .ASCII  /RET/
004210  120  113  124                    .ASCII  /PKT/
004213  040  150  141                    .ASCII  / ha/
004216  163  040  142                    .ASCII  /s b/
004221  141  144  040                    .ASCII  /ad /
```

```
004224  105  116  104                    .ASCII  /END/
004227  103  117  104                    .ASCII  /COD/
004232  105  000                         .ASCII  /E/<00>
004234  045  116  045          P.ACE:    .ASCII  /%N%/
004237  101  052  052                    .ASCII  /A**/
004242  040  121  111                    .ASCII  / QI/
004245  117  137  125                    .ASCII  /O_U/
004250  116  111  124                    .ASCII  /NIT/
004253  072  040  103                    .ASCII  /: C/
004256  123  124  040                    .ASCII  /ST /
004261  045  104  061                    .ASCII  /%D1/
004264  045  101  040                    .ASCII  /%A /
004267  156  157  040                    .ASCII  /no /
004272  165  156  151                    .ASCII  /uni/
004275  164  040  163                    .ASCII  /t s/
004300  145  154  145                    .ASCII  /ele/
004303  143  164  145                    .ASCII  /cte/
004306  144  000                         .ASCII  /d/<00>
004310  045  116  045          P.ACF:    .ASCII  /%N%/
004313  101  052  052                    .ASCII  /A**/
004316  040  125  156                    .ASCII  / Un/
004321  151  164  040                    .ASCII  /it /
004324  043  040  151                    .ASCII  /# i/
004327  163  072  040                    .ASCII  /s: /
004332  045  117  066                    .ASCII  /%O6/
004335  000                             .ASCII  <00>
004336  045  116  045          P.ACG:    .ASCII  /%N%/
004341  101  052  052                    .ASCII  /A**/
004344  040  122  145                    .ASCII  / Re/
004347  155  157  166                    .ASCII  /mov/
004352  141  142  154                    .ASCII  /abl/
004355  145  040  144                    .ASCII  /e d/
004360  151  163  153                    .ASCII  /isk/
004363  040  151  163                    .ASCII  / is/
004366  040  163  145                    .ASCII  / se/
004371  154  145  143                    .ASCII  /lec/
004374  164  145  144                    .ASCII  /ted/
004377  000                             .ASCII  <00>
004400  045  116  045          P.ACH:    .ASCII  /%N%/
004403  101  052  052                    .ASCII  /A**/
004406  040  106  151                    .ASCII  / Fi/
004411  170  145  144                    .ASCII  /xed/
004414  040  144  151                    .ASCII  / di/
004417  163  153  040                    .ASCII  /sk /
004422  151  163  040                    .ASCII  /is /
004425  163  145  154                    .ASCII  /sel/
004430  145  143  164                    .ASCII  /ect/
004433  145  144  000                    .ASCII  /ed/<00>
004436  045  116  045          P.ACI:    .ASCII  /%N%/
004441  101  052  052                    .ASCII  /A**/
004444  040  111  154                    .ASCII  / Il/
004447  154  145  147                    .ASCII  /leg/
004452  141  154  040                    .ASCII  /al /
```

```
004455  146  165  156                    .ASCII  /fun/
004460  143  164  151                    .ASCII  /cti/
004463  157  156  072                    .ASCII  /on:/
004466  040  045  117                    .ASCII  / %O/
004471  066  000  000                    .ASCII  /6/<00><00>
004474  045  116  045          P.ACJ:    .ASCII  /%N%/
004477  101  052  052                    .ASCII  /A**/
004502  040  103  157                    .ASCII  / Co/
004505  155  155  141                    .ASCII  /mma/
004510  156  144  040                    .ASCII  /nd /
004513  162  145  146                    .ASCII  /ref/
004516  040  043  040                    .ASCII  / # /
004521  045  117  066                    .ASCII  /%O6/
004524  045  101  057                    .ASCII  /%A/<57>
004527  045  117  066                    .ASCII  /%O6/
004532  045  101  040                    .ASCII  /%A /
004535  050  117  143                    .ASCII  /(Oc/
004540  164  051  040                    .ASCII  /t) /
004543  156  157  164                    .ASCII  /not/
004546  040  163  145                    .ASCII  / se/
004551  156  164  040                    .ASCII  /nt /
004554  142  171  040                    .ASCII  /by /
004557  110  157  163                    .ASCII  /Hos/
004562  164  000                         .ASCII  /t/<00>
004564  045  116  045          P.ACK:    .ASCII  /%N%/
004567  101  052  052                    .ASCII  /A**/
004572  040  125  156                    .ASCII  / Un/
004575  153  156  157                    .ASCII  /kno/
004600  167  156  040                    .ASCII  /wn /
004603  105  162  162                    .ASCII  /Err/
004606  157  162  040                    .ASCII  /or /
004611  114  157  147                    .ASCII  /Log/
004614  040  146  157                    .ASCII  / fo/
004617  162  155  141                    .ASCII  /rma/
004622  164  040  045                    .ASCII  /t %/
004625  117  063  045                    .ASCII  /03%/
004630  101  040  162                    .ASCII  /A r/
004633  145  143  145                    .ASCII  /ece/
004636  151  166  145                    .ASCII  /ive/
004641  144  000  000                    .ASCII  /d/<00><00>
004644  045  116  045          P.ACL:    .ASCII  /%N%/
004647  101  052  052                    .ASCII  /A**/
004652  040  117  160                    .ASCII  / Op/
004655  055  143  157                    .ASCII  /-co/
004660  144  145  040                    .ASCII  /de /
004663  045  117  063                    .ASCII  /%03/
004666  045  101  054                    .ASCII  /%A,/
004671  040  105  156                    .ASCII  / En/
004674  144  055  143                    .ASCII  /d-c/
004677  157  144  145                    .ASCII  /ode/
004702  040  045  117                    .ASCII  / %O/
004705  063  045  101                    .ASCII  /3%A/
004710  040  146  157                    .ASCII  / fo/
```

```
004713  162     040     162             .ASCII  /r r/
004716  145     146     040             .ASCII  /ef /
004721  043     040     045             .ASCII  /# %/
004724  117     066     045             .ASCII  /06%/
004727  101     057     045             .ASCII  /A/<57>/%/
004732  117     066     045             .ASCII  /06%/
004735  101     040     050             .ASCII  /A (/
004740  070     051     000             .ASCII  /8)/<00>
004743  000                             .ASCII  <00>
004744  045     116     045     P.ACM:  .ASCII  /%N%/
004747  101     052     052             .ASCII  /A**/
004752  040     103     155             .ASCII  / Cm/
004755  144     055     142             .ASCII  /d-b/
004760  143     040     045             .ASCII  /c %/
004763  117     066     045             .ASCII  /06%/
004766  101     057     045             .ASCII  /A/<57>/%/
004771  117     066     045             .ASCII  /06%/
004774  101     040     122             .ASCII  /A R/
004777  163     160     055             .ASCII  /sp-/
005002  142     143     040             .ASCII  /bc /
005005  045     117     066             .ASCII  /%06/
005010  045     101     057             .ASCII  /%A/<57>
005013  045     117     066             .ASCII  /%06/
005016  045     101     040             .ASCII  /%A /
005021  146     157     162             .ASCII  /for/
005024  040     045     117             .ASCII  / %0/
005027  066     045     101             .ASCII  /6%A/
005032  057     045     117             .ASCII  <57>/%0/
005035  066     045     101             .ASCII  /6%A/
005040  040     050     070             .ASCII  / (8/
005043  051     000     000             .ASCII  /)/<00><00>
005046  045     116     045     P.ACN:  .ASCII  /%N%/
005051  101     052     052             .ASCII  /A**/
005054  040     122     145             .ASCII  / Re/
005057  163     160     157             .ASCII  /spo/
005062  156     163     145             .ASCII  /nse/
005065  040     141     154             .ASCII  / al/
005070  162     145     141             .ASCII  /rea/
005073  144     171     040             .ASCII  /dy /
005076  162     145     143             .ASCII  /rec/
005101  145     151     166             .ASCII  /eiv/
005104  145     144     040             .ASCII  /ed /
005107  146     157     162             .ASCII  /for/
005112  040     143     155             .ASCII  / cm/
005115  144     040     045             .ASCII  /d %/
005120  117     066     045             .ASCII  /06%/
005123  101     057     045             .ASCII  /A/<57>/%/
005126  117     066     045             .ASCII  /06%/
005131  101     040     050             .ASCII  /A (/
005134  070     051     000             .ASCII  /8)/<00>
005137  000                             .ASCII  <00>
005140  045     116     045     P.ACO:  .ASCII  /%N%/
005143  101     052     052             .ASCII  /A**/
```

M5


```
005146  040  106  141            .ASCII  / Fa/
005151  151  154  165            .ASCII  /ilu/
005154  162  145  040            .ASCII  /re /
005157  164  157  040            .ASCII  /to /
005162  163  145  156            .ASCII  /sen/
005165  144  040  143            .ASCII  /d c/
005170  157  155  155            .ASCII  /omm/
005173  141  156  144            .ASCII  /and/
005176  040  141  146            .ASCII  / af/
005201  164  145  162            .ASCII  /ter/
005204  040  043  040            .ASCII  / # /
005207  045  117  066            .ASCII  /%O6/
005212  045  101  057            .ASCII  /%A/<57>
005215  045  117  066            .ASCII  /%O6/
005220  045  101  040            .ASCII  /%A /
005223  050  070  051            .ASCII  /(8)/
005226  000  000                 .ASCII  <00><00>
005230  045  116  045  P.ACP:     .ASCII  /%N%/
005233  101  052  052            .ASCII  /A**/
005236  040  122  145            .ASCII  / Re/
005241  163  160  157            .ASCII  /spo/
005244  156  163  145            .ASCII  /nse/
005247  040  150  141            .ASCII  / ha/
005252  163  040  105            .ASCII  /s E/
005255  162  162  157            .ASCII  /rro/
005260  162  040  114            .ASCII  /r L/
005263  157  147  050            .ASCII  /og(/
005266  163  051  072            .ASCII  /s):/
005271  000                      .ASCII  <00>
005272  045  116  045  P.ACQ:     .ASCII  /%N%/
005275  101  052  052            .ASCII  /A**/
005300  040  110  117            .ASCII  / HO/
005303  123  124  040            .ASCII  /ST /
005306  115  105  115            .ASCII  /MEM/
005311  117  122  131            .ASCII  /ORY/
005314  040  105  122            .ASCII  / ER/
005317  122  117  122            .ASCII  /ROR/
005322  054  040  103            .ASCII  /, C/
005325  123  122  040            .ASCII  /SR /
005330  075  040  040            .ASCII  /=  /
005333  045  117  066            .ASCII  /%O6/
005336  045  101  000            .ASCII  /%A/<00>
005341  000                      .ASCII  <00>
005342  045  116  045  P.ACR:     .ASCII  /%N%/
005345  101  052  052            .ASCII  /A**/
005350  040  110  117            .ASCII  / HO/
005353  123  124  040            .ASCII  /ST /
005356  115  105  115            .ASCII  /MEM/
005361  117  122  131            .ASCII  /ORY/
005364  040  105  122            .ASCII  / ER/
005367  122  117  122            .ASCII  /ROR/
005372  054  040  105            .ASCII  /, E/
005375  130  124  105            .ASCII  /XTE/
```

```
005400  116  124  105                .ASCII  /NTE/
005403  104  040  103                .ASCII  /D C/
005406  123  122  040                .ASCII  /SR /
005411  075  040  040                .ASCII  /=  /
005414  045  117  066                .ASCII  /%OG/
005417  045  101  000                .ASCII  /%A/<00>
005422  045  116  045       P.ACS.   .ASCII  /%N%/
005425  101  052  052                .ASCII  /A**/
005430  040  102  101                .ASCII  / BA/
005433  104  040  102                .ASCII  /D B/
005436  114  117  103                .ASCII  /LOC/
005441  113  040  122                .ASCII  /K R/
005444  105  120  114                .ASCII  /EPL/
005447  101  103  105                .ASCII  /ACE/
005452  115  105  116                .ASCII  /MEN/
005455  124  040  122                .ASCII  /T R/
005460  105  121  125                .ASCII  /EQU/
005463  105  123  124                .ASCII  /EST/
005466  040  106  114                .ASCII  / FL/
005471  101  107  040                .ASCII  /AG /
005474  123  105  124                .ASCII  /SET/
005477  000                         .ASCII  <00>
005500  045  116  045       P.ACT:   .ASCII  /%N%/
005503  101  052  052                .ASCII  /A**/
005506  040  105  122                .ASCII  / ER/
005511  122  117  122                .ASCII  /ROR/
005514  040  104  125                .ASCII  / DU/
005517  122  111  116                .ASCII  /RIN/
005522  107  040  122                .ASCII  /G R/
005525  105  120  114                .ASCII  /EPL/
005530  101  103  105                .ASCII  /ACE/
005533  115  105  116                .ASCII  /MEN/
005536  124  040  050                .ASCII  /T (/
005541  102  101  104                .ASCII  /BAD/
005544  040  102  114                .ASCII  / BL/
005547  117  103  113                .ASCII  /OCK/
005552  051  040  106                .ASCII  /) F/
005555  114  101  107                .ASCII  /LAG/
005560  040  123  105                .ASCII  / SE/
005563  124  000  000                .ASCII  /T/<00><00>
005566  045  116  045       P.ACU:   .ASCII  /%N%/
005571  101  125  116                .ASCII  /AUN/
005574  111  124  045                .ASCII  /IT%/
005577  104  062  045                .ASCII  /D2%/
005602  101  040  104                .ASCII  /A D/
005605  122  117  120                .ASCII  /ROP/
005610  120  105  104                .ASCII  /PED/
005613  040  055  040                .ASCII  / - /
005616  000  000                      .ASCII  <00><00>
005620  045  101  125       P.ACW:   .ASCII  /%AU/
005623  123  105  122                .ASCII  /SER/
005626  040  103  117                .ASCII  / CO/
005631  115  115  101                .ASCII  /MMA/
```

B6


```
005634  116  104  045                .ASCII  /ND%/
005637  116  000  000                .ASCII  /N/<00><00>
005642  045  101  103  P.ACX:        .ASCII  /%AC/
005645  117  116  106                .ASCII  /ONF/
005650  111  107  125                .ASCII  /IGU/
005653  122  101  124                .ASCII  /RAT/
005656  111  117  116                .ASCII  /ION/
005661  040  105  122                .ASCII  / ER/
005664  122  117  122                .ASCII  /ROR/
005667  045  116  000                .ASCII  /%N/<00>
005672  045  101  111  P.ACY:        .ASCII  /%AI/
005675  116  111  124                .ASCII  /NIT/
005700  040  105  122                .ASCII  / ER/
005703  122  117  122                .ASCII  /ROR/
005706  045  116  000                .ASCII  /%N/<00>
005711  000                         .ASCII  <00>
005712  045  101  124  P.ACZ:        .ASCII  /%AT/
005715  122  101  116                .ASCII  /RAN/
005720  123  106  105                .ASCII  /SFE/
005723  122  040  114                .ASCII  /R L/
005726  111  115  111                .ASCII  /IMI/
005731  124  040  122                .ASCII  /T R/
005734  105  101  103                .ASCII  /EAC/
005737  110  105  104                .ASCII  /HED/
005742  045  116  000                .ASCII  /%N/<00>
005745  000                         .ASCII  <00>
005746  045  101  105  P.ADA:        .ASCII  /%AE/
005751  122  122  117                .ASCII  /RRO/
005754  122  040  114                .ASCII  /R L/
005757  111  115  111                .ASCII  /IMI/
005762  124  040  122                .ASCII  /T R/
005765  105  101  103                .ASCII  /EAC/
005770  110  105  104                .ASCII  /HED/
005773  045  116  000                .ASCII  /%N/<00>
005776  045  101  125  P.ADB:        .ASCII  /%AU/
006001  116  122  105                .ASCII  /NRE/
006004  103  117  126                .ASCII  /COV/
006007  105  122  101                .ASCII  /ERA/
006012  102  114  105                .ASCII  /BLE/
006015  040  104  122                .ASCII  / DR/
006020  111  126  105                .ASCII  /IVE/
006023  040  105  122                .ASCII  / ER/
006026  122  117  122                .ASCII  /ROR/
006031  045  116  000                .ASCII  /%N/<00>
006034  045  101  125  P.ADC:        .ASCII  /%AU/
006037  116  122  105                .ASCII  /NRE/
006042  103  117  126                .ASCII  /COV/
006045  105  122  101                .ASCII  /ERA/
006050  102  114  105                .ASCII  /BLE/
006053  040  103  117                .ASCII  / CO/
006056  116  124  122                .ASCII  /NTR/
006061  117  114  114                .ASCII  /OLL/
006064  105  122  040                .ASCII  /ER /
```

C6


```
006067  105  122  122                .ASCII  /ERR/
006072  117  122  045                .ASCII  /OR%/
006075  116  000  000                .ASCII  /N/<00><00>
006100  045  101  106  P.ADD:        .ASCII  /%AF/
006103  101  111  114                .ASCII  /AIL/
006106  105  104  040                .ASCII  /ED /
006111  124  117  040                .ASCII  /TO /
006114  103  117  115                .ASCII  /COM/
006117  105  040  117                .ASCII  /E O/
006122  116  114  111                .ASCII  /NLI/
006125  116  105  045                .ASCII  /NE%/
006130  116  000                     .ASCII  /N/<00>
006132  045  101  106  P.ADE:        .ASCII  /%AF/
006135  101  111  114                .ASCII  /AIL/
006140  105  104  040                .ASCII  /ED /
006143  124  117  040                .ASCII  /TO /
006146  101  103  103                .ASCII  /ACC/
006151  105  123  123                .ASCII  /ESS/
006154  040  105  111                .ASCII  / EI/
006157  124  110  105                .ASCII  /THE/
006162  122  040  106                .ASCII  /R F/
006165  111  122  123                .ASCII  /IRS/
006170  124  040  117                .ASCII  /T O/
006173  122  040  114                .ASCII  /R L/
006176  101  123  124                .ASCII  /AST/
006201  040  124  122                .ASCII  / TR/
006204  101  103  113                .ASCII  /ACK/
006207  040  104  125                .ASCII  / DU/
006212  122  111  116                .ASCII  /RIN/
006215  107  040  111                .ASCII  /G I/
006220  116  111  124                .ASCII  /NIT/
006223  045  116  000                .ASCII  /%N/<00>
006226  045  101  104  P.ADF:        .ASCII  /%AD/
006231  111  123  113                .ASCII  /ISK/
006234  040  127  122                .ASCII  / WR/
006237  111  124  105                .ASCII  /ITE/
006242  040  120  122                .ASCII  / PR/
006245  117  124  105                .ASCII  /OTE/
006250  103  124  105                .ASCII  /CTE/
006253  104  045  116                .ASCII  /D%N/
006256  000  000                     .ASCII  <00><00>
006260  045  101  103  P.ADG:        .ASCII  /%AC/
006263  117  115  115                .ASCII  /OMM/
006266  101  116  104                .ASCII  /AND/
006271  040  124  111                .ASCII  / TI/
006274  115  105  040                .ASCII  /ME /
006277  117  125  124                .ASCII  /OUT/
006302  045  116  000                .ASCII  /%N/<00>
006305  000                         .ASCII  <00>
006306  005620'      P.ACV:          .WORD   P.ACW
006310  005642'                      .WORD   P.ACX
006312  005672'                      .WORD   P.ACY
006314  005712'                      .WORD   P.ACZ
```

```
006316  005746'                                 .WORD   P.ADA
006320  005776'                                 .WORD   P.ADB
006322  006034'                                 .WORD   P.ADC
006324  006100'                                 .WORD   P.ADD
006326  006132'                                 .WORD   P.ADE
006330  006226'                                 .WORD   P.ADF
006332  006260'                                 .WORD   P.ADG
006334      045     116     045         P.ADH:  .ASCII  /%N%/
006337      101     120     117                 .ASCII  /APO/
006342      127     105     122                 .ASCII  /WER/
006345      040     104     105                 .ASCII  / DE/
006350      114     101     131                 .ASCII  /LAY/
006353      040     055     040                 .ASCII  / - /
006356      127     101     111                 .ASCII  /WAI/
006361      124     111     116                 .ASCII  /TIN/
006364      107     000                         .ASCII  /G/<00>
006366      045     116     045         P.ADI:  .ASCII  /%N%/
006371      101     106     125                 .ASCII  /AFU/
006374      116     103     124                 .ASCII  /NCT/
006377      111     117     116                 .ASCII  /ION/
006402      101     114     040                 .ASCII  /AL /
006405      124     105     123                 .ASCII  /TES/
006410      124     040     123                 .ASCII  /T S/
006413      124     101     122                 .ASCII  /TAR/
006416      124     105     104                 .ASCII  /TED/
006421      000                                 .ASCII  <00>
006422      045     116     045         P.ADJ:  .ASCII  /%N%/
006425      116     045     101                 .ASCII  /N%A/
006430      105     130     105                 .ASCII  /EXE/
006433      122     103     111                 .ASCII  /RCI/
006436      123     105     122                 .ASCII  /SER/
006441      040     123     124                 .ASCII  / ST/
006444      101     122     124                 .ASCII  /ART/
006447      105     104     045                 .ASCII  /ED%/
006452      116     000                         .ASCII  /N/<00>
006454      045     116     045         P.ADK:  .ASCII  /%N%/
006457      116     045     101                 .ASCII  /N%A/
006462      125     116     124                 .ASCII  /UNT/
006465      040     104     123                 .ASCII  / DS/
006470      113     045     123                 .ASCII  /K%S/
006473      070     045     101                 .ASCII  /8%A/
006476      043     040     117                 .ASCII  /# O/
006501      106     040     040                 .ASCII  /F  /
006504      040     043     040                 .ASCII  / # /
006507      102     131     124                 .ASCII  /BYT/
006512      105     123     040                 .ASCII  /ES /
006515      040     040     043                 .ASCII  /  #/
006520      040     11.     106                 .ASCII  / OF/
006523      040     040     040                 .ASCII  /   /
006526      040     043     040                 .ASCII  / # /
006531      102     131     124                 .ASCII  /BYT/
006534      105     123     000                 .ASCII  /ES/<00>
006537      000                                 .ASCII  <00>
```

```
006540  045  101  040  P.ADL:  .ASCII  /%A /
006543  040  055  055          .ASCII  / --/
006546  110  101  122          .ASCII  /HAR/
006551  104  040  105          .ASCII  /D E/
006554  122  122  117          .ASCII  /RRO/
006557  122  123  055          .ASCII  /RS-/
006562  055  040  055          .ASCII  /- -/
006565  055  123  117          .ASCII  /-SO/
006570  106  124  040          .ASCII  /FT /
006573  105  122  122          .ASCII  /ERR/
006576  117  122  123          .ASCII  /ORS/
006601  055  055  000          .ASCII  /--/<00>
006604  045  116  045  P.ADM:  .ASCII  /%N%/
006607  101  040  043          .ASCII  /A #/
006612  040  040  040          .ASCII  /   /
006615  043  040  040          .ASCII  /#  /
006620  124  131  120          .ASCII  /TYP/
006623  105  040  040          .ASCII  /E  /
006626  122  105  101          .ASCII  /REA/
006631  104  123  040          .ASCII  /DS /
006634  040  040  040          .ASCII  /   /
006637  040  122  105          .ASCII  / RE/
006642  101  104  040          .ASCII  /AD /
006645  040  040  127          .ASCII  /  W/
006650  122  111  124          .ASCII  /RIT/
006653  105  123  040          .ASCII  /ES /
006656  040  040  127          .ASCII  /  W/
006661  122  111  124          .ASCII  /RIT/
006664  124  105  116          .ASCII  /TEN/
006667  000                   .ASCII  <00>
006670  045  101  040  P.ADN:  .ASCII  /%A /
006673  040  123  105          .ASCII  / SE/
006676  113  040  104          .ASCII  /K D/
006701  101  124  040          .ASCII  /AT /
006704  104  122  126          .ASCII  /DRV/
006707  040  110  123          .ASCII  / HS/
006712  124  040  123          .ASCII  /T S/
006715  105  113  040          .ASCII  /EK /
006720  104  101  124          .ASCII  /DAT/
006723  040  104  122          .ASCII  / DR/
006726  126  040  110          .ASCII  /V H/
006731  123  124  000          .ASCII  /ST/<00>
006734  045  116  045  P.ADO:  .ASCII  /%N%/
006737  101  055  055          .ASCII  /A--/
006742  055  040  055          .ASCII  /- -/
006745  055  055  040          .ASCII  /-- /
006750  055  055  055          .ASCII  /---/
006753  055  040  040          .ASCII  /-  /
006756  055  055  055          .ASCII  /---/
006761  055  055  040          .ASCII  /-- /
006764  040  055  055          .ASCII  / --/
006767  055  055  055          .ASCII  /---/
006772  055  055  055          .ASCII  /---/
```

F6


```
006775  055  040  055                .ASCII  /- -/
007000  055  055  055                .ASCII  /---/
007003  055  055  040                .ASCII  /-- /
007006  040  055  055                .ASCII  / --/
007011  055  055  055                .ASCII  /---/
007014  055  055  055                .ASCII  /---/
007017  055  000  000                .ASCII  /-/<00><00>
007022  045  101  040     P.ADP:     .ASCII  /%A /
007025  055  055  055                .ASCII  /---/
007030  040  055  055                .ASCII  / --/
007033  055  040  055                .ASCII  /- -/
007036  055  055  040                .ASCII  /-- /
007041  055  055  055                .ASCII  /---/
007044  040  055  055                .ASCII  / --/
007047  055  040  055                .ASCII  /- -/
007052  055  055  040                .ASCII  /-- /
007055  055  055  055                .ASCII  /---/
007060  040  055  055                .ASCII  / --/
007063  055  000  000                .ASCII  /-/<00><00>
007066  045  116  045     P.ADQ:     .ASCII  /%N%/
007071  104  062  045                .ASCII  /D2%/
007074  104  064  045                .ASCII  /D4%/
007077  123  062  045                .ASCII  /S2%/
007102  124  000                    .ASCII  /T/<00>
007104  045  104  064     P.ADR:     .ASCII  /%D4/
007107  045  132  063                .ASCII  /%Z3/
007112  045  104  063                .ASCII  /%D3/
007115  045  101  054                .ASCII  /%A,/
007120  045  132  063                .ASCII  /%Z3/
007123  045  101  054                .ASCII  /%A,/
007126  045  132  063                .ASCII  /%Z3/
007131  000                         .ASCII  <C0>
007132  045  104  064     P.ADS:     .ASCII  /%D4/
007135  045  104  064                .ASCII  /%D4/
007140  045  104  064                .ASCII  /%D4/
007143  045  104  064                .ASCII  /%D4/
007146  045  104  064                .ASCII  /%D4/
007151  045  104  064                .ASCII  /%D4/
007154  045  104  064                .ASCII  /%D4/
007157  045  104  064                .ASCII  /%D4/
007162  000  000                    .ASCII  <00><00>
007164  045  116  045     P.ADT:     .ASCII  /%N%/
007167  101  040  056                .ASCII  /A ./
007172  040  040  040                .ASCII  /   /
007175  056  040  040                .ASCII  /.  /
007200  103  116  124                .ASCII  /CNT/
007203  122  040  040                .ASCII  /R  /
007206  040  040  040                .ASCII  /   /
007211  040  056  040                .ASCII  / . /
007214  040  056  056                .ASCII  / ../
007217  056  056  056                .ASCII  /.../
007222  056  056  056                .ASCII  /.../
007225  056  040  040                .ASCII  /.  /
```

```
007230  040     040     040             .ASCII  /   /
007233  040     056     040             .ASCII  / . /
007236  040     056     056             .ASCII  / ../
007241  056     056     056             .ASCII  /.../
007244  056     056     056             .ASCII  /.../
007247  056     000     000             .ASCII  /./<00><00>
007252  045     101     040     P.ADU:  .ASCII  /%A /
007255  040     040     056             .ASCII  /  ./
007260  040     040     040             .ASCII  /   /
007263  056     045     104             .ASCII  /.%D/
007266  064     045     101             .ASCII  /4%A/
007271  040     040     040             .ASCII  /   /
007274  056     040     040             .ASCII  /.  /
007277  040     056     040             .ASCII  / . /
007302  040     040     056             .ASCII  /  ./
007305  045     104     064             .ASCII  /%D4/
007310  045     101     040             .ASCII  /%A /
007313  040     040     056             .ASCII  /  ./
007316  000     000                     .ASCII  <00><00>
007320  045     101     040     P.ADV:  .ASCII  /%A /
007323  040     040     056             .ASCII  /  ./
007326  040     040     040             .ASCII  /   /
007331  056     045     104             .ASCII  /.%D/
007334  064     045     101             .ASCII  /4%A/
007337  040     040     040             .ASCII  /   /
007342  056     040     040             .ASCII  /.  /
007345  040     056     040             .ASCII  / . /
007350  040     040     056             .ASCII  /  ./
007353  045     104     064             .ASCII  /%D4/
007356  045     101     040             .ASCII  /%A /
007361  040     040     056             .ASCII  /  ./
007364  000     000                     .ASCII  <00><00>
007366  045     116     045     P.ADW:  .ASCII  /%N%/
007371  116     045     101             .ASCII  /N%A/
007374  125     116     111             .ASCII  /UNI/
007377  124     040     040             .ASCII  /T  /
007402  104     111     123             .ASCII  /DIS/
007405  113     040     040             .ASCII  /K  /
007410  040     040     040             .ASCII  /   /
007413  040     040     040             .ASCII  /   /
007416  040     040     040             .ASCII  /   /
007421  040     043     040             .ASCII  / # /
007424  117     106     040             .ASCII  /OF /
007427  040     040     043             .ASCII  /  #/
007432  040     102     114             .ASCII  / BL/
007435  113     123     040             .ASCII  /KS /
007440  040     040     040             .ASCII  /   /
007443  040     040     040             .ASCII  /   /
007446  043     040     117             .ASCII  /# O/
007451  106     040     040             .ASCII  /F  /
007454  040     040     043             .ASCII  /  #/
007457  040     102     114             .ASCII  / BL/
007462  113     123     040             .ASCII  /KS /
```

H6


```
007465  000                             .ASCII  <00>
007466  045  116  045          P.ADX:    .ASCII  /%N%/
007471  101  040  040                    .ASCII  /A  /
007474  043  040  040                    .ASCII  /#  /
007477  040  040  040                    .ASCII  /   /
007502  043  040  040                    .ASCII  /#  /
007505  040  040  040                    .ASCII  /   /
007510  124  131  120                    .ASCII  /TYP/
007513  105  040  040                    .ASCII  /E  /
007516  040  122  105                    .ASCII  / RE/
007521  101  104  123                    .ASCII  /ADS/
007524  040  040  040                    .ASCII  /   /
007527  040  040  122                    .ASCII  /  R/
007532  105  101  104                    .ASCII  /EAD/
007535  040  040  040                    .ASCII  /   /
007540  040  040  040                    .ASCII  /   /
007543  127  122  111                    .ASCII  /WRI/
007546  124  105  123                    .ASCII  /TES/
007551  040  040  127                    .ASCII  /  W/
007554  122  111  124                    .ASCII  /RIT/
007557  124  105  116                    .ASCII  /TEN/
007562  040  000                        .ASCII  / /<00>
007564  045  116  045          P.ADY:    .ASCII  /%N%/
007567  101  055  055                    .ASCII  /A--/
007572  055  055  040                    .ASCII  /-- /
007575  040  055  055                    .ASCII  / --/
007600  055  055  040                    .ASCII  /-- /
007603  040  055  055                    .ASCII  / --/
007606  055  055  055                    .ASCII  /---/
007611  055  055  040                    .ASCII  /-- /
007614  040  055  055                    .ASCII  / --/
007617  055  055  055                    .ASCII  /---/
007622  055  040  040                    .ASCII  /-  /
007625  040  055  055                    .ASCII  / --/
007630  055  055  055                    .ASCII  /---/
007633  055  040  040                    .ASCII  /-  /
007636  040  040  040                    .ASCII  /   /
007641  055  055  055                    .ASCII  /---/
007644  055  055  055                    .ASCII  /---/
007647  040  040  040                    .ASCII  /   /
007652  055  055  055                    .ASCII  /---/
007655  055  055  055                    .ASCII  /---/
007660  040  040  000                    .ASCII  /  /<00>
007663  000                             .ASCII  <00>
007664  045  116  045          P.ADZ:    .ASCII  /%N%/
007667  123  061  045                    .ASCII  /S1%/
007672  104  062  045                    .ASCII  /D2%/
007675  123  064  045                    .ASCII  /S4%/
007700  104  062  045                    .ASCII  /D2%/
007703  101  040  040                    .ASCII  /A  /
007706  040  104  102                    .ASCII  / DB/
007711  116  040  111                    .ASCII  /N I/
007714  057  117  040                    .ASCII  <57>/O /
```

I6


```
007717  040  045  104          .ASCII  / %D/
007722  066  045  123          .ASCII  /6%S/
007725  063  045  104          .ASCII  /3%D/
007730  066  045  123          .ASCII  /6%S/
007733  065  045  104          .ASCII  /5%D/
007736  066  045  123          .ASCII  /6%S/
007741  063  045  104          .ASCII  /3%D/
007744  066  000               .ASCII  /6/<00>
007746  124  117  117  P.AEA:   .ASCII  /TOO/
007751  040  115  101          .ASCII  / MA/
007754  116  131  040          .ASCII  /NY /
007757  125  116  111          .ASCII  /UNI/
007762  124  123  000          .ASCII  /TS/<00>
007765  000                    .ASCII  <00>
007766  116  117  124  P.AEB:   .ASCII  /NOT/
007771  040  105  116          .ASCII  / EN/
007774  117  125  107          .ASCII  /OUG/
007777  110  040  106          .ASCII  /H F/
010002  122  105  105          .ASCII  /REE/
010005  040  115  105          .ASCII  / ME/
010010  115  117  122          .ASCII  /MOR/
010013  131  040  106          .ASCII  /Y F/
010016  117  122  040          .ASCII  /OR /
010021  101  114  114          .ASCII  /ALL/
010024  117  103  101          .ASCII  /OCA/
010027  124  111  116          .ASCII  /TIN/
010032  107  040  122          .ASCII  /G R/
010035  105  101  104          .ASCII  /EAD/
010040  057  127  122          .ASCII  <57>/WR/
010043  111  124  105          .ASCII  /ITE/
010046  040  102  125          .ASCII  / BU/
010051  106  106  105          .ASCII  /FFE/
010054  122  123  000          .ASCII  /RS/<00>
010057  000                    .ASCII  <00>
010060  122  105  107  P.AEC:   .ASCII  /REG/
010063  111  123  124          .ASCII  /IST/
010066  105  122  040          .ASCII  /ER /
010071  105  130  111          .ASCII  /EXI/
010074  123  124  105          .ASCII  /STE/
010077  116  103  105          .ASCII  /NCE/
010102  040  124  105          .ASCII  / TE/
010105  123  124  040          .ASCII  /ST /
010110  106  101  111          .ASCII  /FAI/
010113  114  105  104          .ASCII  /LED/
010116  000  000               .ASCII  <00><00>
010120  126  105  103  P.AED:   .ASCII  /VEC/
010123  124  117  122          .ASCII  /TOR/
010126  040  124  105          .ASCII  / TE/
010131  123  124  040          .ASCII  /ST /
010134  106  101  111          .ASCII  /FAI/
010137  114  105  104          .ASCII  /LED/
010142  000  000               .ASCII  <00><00>
010144  102  122  040  P.AEE:   .ASCII  /BR /
```

J6


```
010147  114     105     126             .ASCII  /LEV/
010152  105     114     040             .ASCII  /EL /
010155  124     105     123             .ASCII  /TES/
010160  124     040     106             .ASCII  /T F/
010163  101     111     114             .ASCII  /AIL/
010166  105     104     000             .ASCII  /ED/<00>
010171  000                             .ASCII  <00>
010172  111     116     111     P.AEF:  .ASCII  /INI/
010175  124     040     123             .ASCII  /T S/
010200  105     121     125             .ASCII  /EQU/
010203  105     116     103             .ASCII  /ENC/
010206  105     040     106             .ASCII  /E F/
010211  101     111     114             .ASCII  /AIL/
010214  105     104     000             .ASCII  /ED/<00>
010217  000                             .ASCII  <00>
010220  106     101     124     P.AEG:  .ASCII  /FAT/
010223  101     114     040             .ASCII  /AL /
010226  103     117     116             .ASCII  /CON/
010231  124     122     117             .ASCII  /TRO/
010234  114     114     105             .ASCII  /LLE/
010237  122     040     105             .ASCII  /R E/
010242  122     122     117             .ASCII  /RRO/
010245  122     000     000             .ASCII  /R/<00><00>
010250  117     116     114     P.AEH:  .ASCII  /ONL/
010253  111     116     105             .ASCII  /INE/
010256  040     106     101             .ASCII  / FA/
010261  111     114     105             .ASCII  /ILE/
010264  104     000                     .ASCII  /D/<00>
010266  127     122     111     P.AEI:  .ASCII  /WRI/
010271  124     105     055             .ASCII  /TE-/
010274  120     122     117             .ASCII  /PRO/
010277  124     105     103             .ASCII  /TEC/
010302  124     040     103             .ASCII  /T C/
010305  117     116     106             .ASCII  /ONF/
010310  114     111     103             .ASCII  /LIC/
010313  124     000     000             .ASCII  /T/<00><00>
010316  101     103     103     P.AEJ:  .ASCII  /ACC/
010321  105     123     123             .ASCII  /ESS/
010324  040     106     101             .ASCII  / FA/
010327  111     114     105             .ASCII  /ILE/
010332  104     000                     .ASCII  /D/<00>
010334  106     101     124     P.AEK:  .ASCII  /FAT/
010337  101     114     040             .ASCII  /AL /
010342  111     057     117             .ASCII  /I/<57>/O/
010345  040     105     122             .ASCII  / ER/
010350  122     117     122             .ASCII  /ROR/
010353  000                             .ASCII  <00>
010354  104     111     123     P.AEL:  .ASCII  /DIS/
010357  113     040     124             .ASCII  /K T/
010362  131     120     105             .ASCII  /YPE/
010365  040     125     116             .ASCII  / UN/
010370  113     116     117             .ASCII  /KNO/
010373  127     116     040             .ASCII  /WN /
```

K6


```
010376  124  117  040                .ASCII  /TO /
010401  105  130  105                .ASCII  /EXE/
010404  122  103  111                .ASCII  /RCI/
010407  123  105  122                .ASCII  /SER/
010412  000  000                     .ASCII  <00><00>
010414  106  101  111        P.AEM:  .ASCII  /FAI/
010417  114  105  104                .ASCII  /LED/
010422  040  124  117                .ASCII  / TO/
010425  040  123  105                .ASCII  / SE/
010430  116  104  040                .ASCII  /ND /
010433  123  105  124                .ASCII  /SET/
010436  055  103  117                .ASCII  /-CO/
010441  116  124  122                .ASCII  /NTR/
010444  117  114  114                .ASCII  /OLL/
010447  105  122  055                .ASCII  /ER-/
010452  103  110  101                .ASCII  /CHA/
010455  122  101  103                .ASCII  /RAC/
010460  124  105  122                .ASCII  /TER/
010463  111  123  124                .ASCII  /IST/
010466  111  103  123                .ASCII  /ICS/
010471  040  103  117                .ASCII  / CO/
010474  115  115  101                .ASCII  /MMA/
010477  116  104  000                .ASCII  /ND/<00>
010502  123  105  124        P.AEN:  .ASCII  /SET/
010505  055  103  117                .ASCII  /-CO/
010510  116  124  122                .ASCII  /NTR/
010513  117  114  114                .ASCII  /OLL/
010516  105  122  055                .ASCII  /ER-/
010521  103  110  101                .ASCII  /CHA/
010524  122  101  103                .ASCII  /RAC/
010527  124  105  122                .ASCII  /TER/
010532  111  123  124                .ASCII  /IST/
010535  111  103  123                .ASCII  /ICS/
010540  040  122  105                .ASCII  / RE/
010543  123  120  117                .ASCII  /SPO/
010546  116  123  105                .ASCII  /NSE/
010551  040  110  101                .ASCII  / HA/
010554  123  040  102                .ASCII  /S B/
010557  101  104  040                .ASCII  /AD /
010562  105  116  104                .ASCII  /END/
010565  103  117  104                .ASCII  /COD/
010570  105  040  117                .ASCII  /E O/
010573  122  040  106                .ASCII  /R F/
010576  114  101  107                .ASCII  /LAG/
010601  123  040  111                .ASCII  /S I/
010604  116  040  105                .ASCII  /N E/
010607  122  122  117                .ASCII  /RRO/
010612  122  000                     .ASCII  /R/<00>
010614  106  101  111        P.AEO:  .ASCII  /FAI/
010617  114  105  104                .ASCII  /LED/
010622  040  124  117                .ASCII  / TO/
010625  040  123  105                .ASCII  / SE/
010630  116  104  040                .ASCII  /ND /
```

L6


```
010633  117     116     055             .ASCII  /ON-/
010636  114     111     116             .ASCII  /LIN/
010641  105     040     103             .ASCII  /E C/
010644  117     115     115             .ASCII  /OMM/
010647  101     116     104             .ASCII  /AND/
010652  000     000                     .ASCII  <00><00>
010654  117     116     055     P.AEP:  .ASCII  /ON-/
010657  114     111     116             .ASCII  /LIN/
010662  105     040     122             .ASCII  /E R/
010665  105     123     120             .ASCII  /ESP/
010670  117     116     123             .ASCII  /ONS/
010673  105     040     110             .ASCII  /E H/
010676  101     123     040             .ASCII  /AS /
010701  102     101     104             .ASCII  /BAD/
010704  040     105     116             .ASCII  / EN/
010707  104     103     117             .ASCII  /DCO/
010712  104     105     000             .ASCII  /DE/<00>
010715  000                             .ASCII  <00>
010716  117     116     055     P.AEQ:  .ASCII  /ON-/
010721  114     111     116             .ASCII  /LIN/
010724  105     040     122             .ASCII  /E R/
010727  105     123     120             .ASCII  /ESP/
010732  117     116     123             .ASCII  /ONS/
010735  105     040     110             .ASCII  /E H/
010740  101     123     040             .ASCII  /AS /
010743  125     116     113             .ASCII  /UNK/
010746  116     117     127             .ASCII  /NOW/
010751  116     040     104             .ASCII  /N D/
010754  105     126     111             .ASCII  /EVI/
010757  103     105     000             .ASCII  /CE/<00>
010762  111     057     117     P.AER:  .ASCII  /I/<57>/O/
010765  040     122     105             .ASCII  / RE/
010770  121     125     105             .ASCII  /QUE/
010773  123     124     040             .ASCII  /ST /
010776  106     101     111             .ASCII  /FAI/
011001  114     105     104             .ASCII  /LED/
011004  000     000                     .ASCII  <00><00>
011006  045     101     115     P.AES:  .ASCII  /%AM/
011011  117     122     105             .ASCII  /ORE/
011014  040     124     110             .ASCII  / TH/
011017  101     116     040             .ASCII  /AN /
011022  045     104     062             .ASCII  /%D2/
011025  045     101     040             .ASCII  /%A /
011030  125     116     111             .ASCII  /UNI/
011033  124     123     040             .ASCII  /TS /
011036  123     120     105             .ASCII  /SPE/
011041  103     111     106             .ASCII  /CIF/
011044  111     105     104             .ASCII  /IED/
011047  000                             .ASCII  <00>
011050  045     101     052     P.AET:  .ASCII  /%A*/
011053  040     116     117             .ASCII  / NO/
011056  040     122     105             .ASCII  / RE/
011061  123     120     117             .ASCII  /SPO/
```

M6


```
011064  116  123  105                .ASCII  /NSE/
011067  040  101  124                .ASCII  / AT/
011072  040  101  104                .ASCII  / AD/
011075  104  122  105                .ASCII  /DRE/
011100  123  123  040                .ASCII  /SS /
011103  045  117  066                .ASCII  /%O6/
011106  000  000                     .ASCII  <00><00>
011110  045  101  052        P.AEU:  .ASCII  /%A*/
011113  040  111  116                .ASCII  / IN/
011116  103  117  122                .ASCII  /COR/
011121  122  105  103                .ASCII  /REC/
011124  124  040  102                .ASCII  /T B/
011127  122  040  114                .ASCII  /R L/
011132  105  126  105                .ASCII  /EVE/
011135  114  040  106                .ASCII  /L F/
011140  117  122  040                .ASCII  /OR /
011143  104  122  111                .ASCII  /DRI/
011146  126  105  040                .ASCII  /VE /
011151  045  117  066                .ASCII  /%O6/
011154  000  000                     .ASCII  <00><00>
011156  045  101  052        P.AEV:  .ASCII  /%A*/
011161  040  123  124                .ASCII  / ST/
011164  105  120  040                .ASCII  /EP /
011167  045  104  061                .ASCII  /%D1/
011172  045  101  040                .ASCII  /%A /
011175  122  105  101                .ASCII  /REA/
011200  104  040  105                .ASCII  /D E/
011203  122  122  117                .ASCII  /RRO/
011206  122  000                     .ASCII  /R/<00>
011210  045  101  052        P.AEW:  .ASCII  /%A*/
011213  040  102  101                .ASCII  / BA/
011216  104  040  123                .ASCII  /D S/
011221  101  040  103                .ASCII  /A C/
011224  117  104  105                .ASCII  /ODE/
011227  040  106  122                .ASCII  / FR/
011232  117  115  040                .ASCII  /OM /
011235  104  122  111                .ASCII  /DRI/
011240  126  105  040                .ASCII  /VE /
011243  045  117  066                .ASCII  /%O6/
011246  000  000                     .ASCII  <00><00>
011250  045  101  052        P.AEX:  .ASCII  /%A*/
011253  040  104  111                .ASCII  / DI/
011256  123  113  045                .ASCII  /SK%/
011261  104  062  045                .ASCII  /D2%/
011264  101  040  127                .ASCII  /A W/
011267  105  116  124                .ASCII  /ENT/
011272  040  117  106                .ASCII  / OF/
011275  106  114  111                .ASCII  /FLI/
011300  116  105  000                .ASCII  /NE/<00>
011303  000                         .ASCII  <00>
011304  045  101  052        P.AEY:  .ASCII  /%A*/
011307  040  104  122                .ASCII  / DR/
011312  111  126  105                .ASCII  /IVE/
```

N6


```
011315  040  045  117          .ASCII  / %O/
011320  066  045  101          .ASCII  /6%A/
011323  040  116  117          .ASCII  / NO/
011326  124  040  120          .ASCII  /T P/
011331  122  117  103          .ASCII  /ROC/
011334  105  123  123          .ASCII  /ESS/
011337  111  116  107          .ASCII  /ING/
011342  040  103  117          .ASCII  / CO/
011345  115  115  101          .ASCII  /MMA/
011350  116  104  040          .ASCII  /ND /
011353  120  101  103          .ASCII  /PAC/
011356  113  105  124          .ASCII  /KET/
011361  123  000  000          .ASCII  /S/<00><00>
011364  045  101  052  P.AEZ:  .ASCII  /%A*,
011367  040  104  111          .ASCII  / DI/
011372  123  113  045          .ASCII  /SK%/
011375  104  062  045          .ASCII  /D2%/
011400  101  040  127          .ASCII  /A W/
011403  105  116  124          .ASCII  /ENT/
011406  040  124  117          .ASCII  / TO/
011411  040  124  110          .ASCII  / TH/
011414  105  040  042          .ASCII  /E "/
011417  101  126  101          .ASCII  /AVA/
011422  111  114  101          .ASCII  /ILA/
011425  102  114  105          .ASCII  /BLE/
011430  042  040  123          .ASCII  /" S/
011433  124  101  124          .ASCII  /TAT/
011436  105  000               .ASCII  /E/<00>
011440  040  055  040  P.AFA:  .ASCII  / - /
011443  125  116  122          .ASCII  /UNR/
011446  105  103  117          .ASCII  /ECO/
011451  107  116  111          .ASCII  /GNI/
011454  132  105  104          .ASCII  /ZED/
011457  040  115  105          .ASCII  / ME/
011462  123  123  101          .ASCII  /SSA/
011465  107  105  040          .ASCII  /GE /
011470  124  131  120          .ASCII  /TYP/
011473  105  000  000          .ASCII  /E/<00><00>
011476  040  055  040  P.AFB:  .ASCII  / - /
011501  125  116  122          .ASCII  /UNR/
011504  105  103  117          .ASCII  /ECO/
011507  107  116  111          .ASCII  /GNI/
011512  132  105  104          .ASCII  /ZED/
011515  040  103  117          .ASCII  / CO/
011520  116  116  105          .ASCII  /NNE/
011523  103  124  111          .ASCII  /CTI/
011526  117  116  040          .ASCII  /ON /
011531  111  104  000          .ASCII  /ID/<00>
011534  040  055  040  P.AFC:  .ASCII  / - /
011537  125  116  122          .ASCII  /UNR/
011542  105  103  117          .ASCII  /ECO/
011545  107  116  111          .ASCII  /GNI/
011550  132  105  104          .ASCII  /ZED/
```

```
011553  040  122  105                .ASCII  / RE/
011556  124  125  122                .ASCII  /TUR/
011561  116  040  115                .ASCII  /N M/
011564  105  123  123                .ASCII  /ESS/
011567  101  107  105                .ASCII  /AGE/
011572  000  000                     .ASCII  <00><00>
011574  040  055  040       P.AFD:   .ASCII  / - /
011577  125  116  122                .ASCII  /UNR/
011602  105  103  117                .ASCII  /ECO/
011605  107  116  111                .ASCII  /GNI/
011610  132  105  104                .ASCII  /ZED/
011613  040  122  105                .ASCII  / RE/
011616  124  125  122                .ASCII  /TUR/
011621  116  040  120                .ASCII  /N P/
011624  101  103  113                .ASCII  /ACK/
011627  105  124  000                .ASCII  /ET/<00>
011632  040  055  040       P.AFE:   .ASCII  / - /
011635  125  116  122                .ASCII  /UNR/
011640  105  103  117                .ASCII  /ECO/
011643  107  116  111                .ASCII  /GNI/
011646  132  105  104                .ASCII  /ZED/
011651  040  103  122                .ASCII  / CR/
011654  116  000                     .ASCII  /N/<00>
011656  040  055  040       P.AFF:   .ASCII  / - /
011661  125  116  122                .ASCII  /UNR/
011664  105  103  117                .ASCII  /ECO/
011667  107  116  111                .ASCII  /GNI/
011672  132  105  104                .ASCII  /ZED/
011675  040  117  120                .ASCII  / OP/
011700  103  117  104                .ASCII  /COD/
011703  105  000  000                .ASCII  /E/<00><00>
011706  040  055  040       P.AFG:   .ASCII  / - /
011711  115  123  103                .ASCII  /MSC/
011714  120  040  123                .ASCII  /P S/
011717  124  101  124                .ASCII  /TAT/
011722  125  123  040                .ASCII  /US /
011725  103  117  104                .ASCII  /COD/
011730  105  040  105                .ASCII  /E E/
011733  122  122  000                .ASCII  /RR/<00>
011736  040  055  040       P.AFH:   .ASCII  / - /
011741  104  125  120                .ASCII  /DUP/
011744  040  123  124                .ASCII  / ST/
011747  101  124  125                .ASCII  /ATU/
011752  123  040  103                .ASCII  /S C/
011755  117  104  105                .ASCII  /ODE/
011760  040  105  122                .ASCII  / ER/
011763  122  000  000                .ASCII  /R/<00><00>
011766  040  055  040       P.AFI:   .ASCII  / - /
011771  125  116  122                .ASCII  /UNR/
011774  105  103  117                .ASCII  /ECO/
011777  107  116  111                .ASCII  /GNI/
012002  132  105  104                .ASCII  /ZED/
012005  040  123  124                .ASCII  / ST/
```

```
012010  101  124  125                    .ASCII  /ATU/
012013  123  040  103                    .ASCII  /S C/
012016  117  104  105                    .ASCII  /ODE/
012021  000                             .ASCII  <00>
012022  040  055  040          P.AFJ:    .ASCII  / - /
012025  114  102  116                    .ASCII  /LBN/
012030  040  110  117                    .ASCII  / HO/
012033  123  124  040                    .ASCII  /ST /
012036  103  117  115                    .ASCII  /COM/
012041  120  101  122                    .ASCII  /PAR/
012044  105  040  105                    .ASCII  /E E/
012047  122  122  000                    .ASCII  /RR/<00>
012052  040  055  040          P.AFK:    .ASCII  / - /
012055  104  102  116                    .ASCII  /DBN/
012060  040  110  117                    .ASCII  / HO/
012063  123  124  040                    .ASCII  /ST /
012066  103  117  115                    .ASCII  /COM/
012071  120  101  122                    .ASCII  /PAR/
012074  105  040  105                    .ASCII  /E E/
012077  122  122  000                    .ASCII  /RR/<00>
012102  040  055  040          P.AFL:    .ASCII  / - /
012105  125  116  101                    .ASCII  /UNA/
012110  102  114  105                    .ASCII  /BLE/
012113  040  124  117                    .ASCII  / TO/
012116  040  114  117                    .ASCII  / LO/
012121  101  104  040                    .ASCII  /AD /
012124  104  125  120                    .ASCII  /DUP/
012127  040  115  105                    .ASCII  / ME/
012132  104  111  101                    .ASCII  /DIA/
012135  000                             .ASCII  <00>
012136  040  055  040          P.AFM:    .ASCII  / - /
012141  105  122  122                    .ASCII  /ERR/
012144  040  111  116                    .ASCII  / IN/
012147  040  104  125                    .ASCII  / DU/
012152  120  040  120                    .ASCII  /P P/
012155  113  124  040                    .ASCII  /KT /
012160  127  110  105                    .ASCII  /WHE/
012163  116  040  125                    .ASCII  /N U/
012166  123  111  116                    .ASCII  /SIN/
012171  107  040  103                    .ASCII  /G C/
012174  124  114  122                    .ASCII  /TLR/
012177  040  114  103                    .ASCII  / LC/
012202  040  120  122                    .ASCII  / PR/
012205  117  107  000                    .ASCII  /OG/<00>
012210  045  101  052          P.AFN:    .ASCII  /%A*/
012213  040  104  111                    .ASCII  / DI/
012216  123  113  045                    .ASCII  /SK%/
012221  104  062  000                    .ASCII  /D2/<00>
012224  045  101  111          P.AFP:    .ASCII  /%AI/
012227  116  126  101                    .ASCII  /NVA/
012232  114  111  104                    .ASCII  /LID/
012235  040  103  117                    .ASCII  / CO/
012240  115  115  101                    .ASCII  /MMA/
```

```
012243  116  104  000          .ASCII  /ND/<00>
012246  045  101  103  P.AFQ:   .ASCII  /%AC/
012251  117  115  115          .ASCII  /OMM/
012254  101  116  104          .ASCII  /AND/
012257  040  101  102          .ASCII  / AB/
012262  117  122  124          .ASCII  /ORT/
012265  105  104  000          .ASCII  /ED/<00>
012270  045  101  125  P.AFR:   .ASCII  /%AU/
012273  116  111  124          .ASCII  /NIT/
012276  040  117  106          .ASCII  / OF/
012301  106  114  111          .ASCII  /FLI/
012304  116  105  000          .ASCII  /NE/<00>
012307  000                    .ASCII  <00>
012310  045  101  124  P.AFS:   .ASCII  /%AT/
012313  122  101  116          .ASCII  /RAN/
012316  123  111  124          .ASCII  /SIT/
012321  111  117  116          .ASCII  /ION/
012324  040  124  117          .ASCII  / TO/
012327  040  101  126          .ASCII  / AV/
012332  101  111  114          .ASCII  /AIL/
012335  101  102  114          .ASCII  /ABL/
012340  105  040  123          .ASCII  /E S/
012343  124  101  124          .ASCII  /TAT/
012346  105  000               .ASCII  /E/<00>
012350  045  101  115  P.AFT:   .ASCII  /%AM/
012353  105  104  111          .ASCII  /EDI/
012356  101  040  106          .ASCII  /A F/
012361  117  122  115          .ASCII  /ORM/
012364  101  124  040          .ASCII  /AT /
012367  105  122  122          .ASCII  /ERR/
012372  117  122  000          .ASCII  /OR/<00>
012375  000                    .ASCII  <00>
012376  045  101  127  P.AFU:   .ASCII  /%AW/
012401  122  111  124          .ASCII  /RIT/
012404  105  055  120          .ASCII  /E-P/
012407  122  117  124          .ASCII  /ROT/
012412  105  103  124          .ASCII  /ECT/
012415  105  104  000          .ASCII  /ED/<00>
012420  045  101  104  P.AFV:   .ASCII  /%AD/
012423  105  126  111          .ASCII  /EVI/
012426  103  105  040          .ASCII  /CE /
012431  103  117  115          .ASCII  /COM/
012434  120  101  122          .ASCII  /PAR/
012437  105  040  105          .ASCII  /E E/
012442  122  122  117          .ASCII  /RRO/
012445  122  000  000          .ASCII  /R/<00><00>
012450  045  101  104  P.AFW:   .ASCII  /%AD/
012453  101  124  101          .ASCII  /ATA/
012456  040  105  122          .ASCII  / ER/
012461  122  117  122          .ASCII  /ROR/
012464  000  000               .ASCII  <00><00>
012466  045  101  110  P.AFX:   .ASCII  /%AH/
012471  117  123  124          .ASCII  /OST/
```

E7


```
012474  040  102  125                .ASCII  / BU/
012477  106  106  105                .ASCII  /FFE/
012502  122  040  101                .ASCII  /R A/
012505  103  103  105                .ASCII  /CCE/
012510  123  123  040                .ASCII  /SS /
012513  105  122  122                .ASCII  /ERR/
012516  117  122  000                .ASCII  /OR/<00>
012521  000                        .ASCII  <00>
012522  045  101  103   P.AFY:      .ASCII  /%AC/
012525  117  116  124                .ASCII  /ONT/
012530  122  117  114                .ASCII  /ROL/
012533  114  105  122                .ASCII  /LER/
012536  040  105  122                .ASCII  / ER/
012541  122  117  122                .ASCII  /ROR/
012544  000  000                    .ASCII  <00><00>
012546  045  101  104   P.AFZ:      .ASCII  /%AD/
012551  122  111  126                .ASCII  /RIV/
012554  105  040  105                .ASCII  /E E/
012557  122  122  117                .ASCII  /RRO/
012562  122  000                    .ASCII  /R/<00>
012564  045  101  115   P.AGA:      .ASCII  /%AM/
012567  105  123  123                .ASCII  /ESS/
012572  101  107  105                .ASCII  /AGE/
012575  040  106  122                .ASCII  / FR/
012600  117  115  040                .ASCII  /OM /
012603  111  116  124                .ASCII  /INT/
012606  105  122  116                .ASCII  /ERN/
012611  101  114  040                .ASCII  /AL /
012614  104  111  101                .ASCII  /DIA/
012617  107  116  117                .ASCII  /GNO/
012622  123  124  111                .ASCII  /STI/
012625  103  123  000                .ASCII  /CS/<00>
012630  045  101  110   P.AGB:      .ASCII  /%AH/
012633  117  123  124                .ASCII  /OST/
012636  040  103  117                .ASCII  / CO/
012641  115  120  101                .ASCII  /MPA/
012644  122  105  040                .ASCII  /RE /
012647  105  122  122                .ASCII  /ERR/
012652  117  122  000                .ASCII  /OR/<00>
012655  000                        .ASCII  <00>
012656  045  101  103   P.AGC:      .ASCII  /%AC/
012661  117  115  115                .ASCII  /OMM/
012664  101  116  104                .ASCII  /AND/
012667  040  124  111                .ASCII  / TI/
012672  115  105  117                .ASCII  /MEO/
012675  125  124  000                .ASCII  /UT/<00>
012700  045  101  102   P.AGD:      .ASCII  /%AB/
012703  101  104  040                .ASCII  /AD /
012706  102  114  117                .ASCII  /BLO/
012711  103  113  040                .ASCII  /CK /
012714  122  105  120                .ASCII  /REP/
012717  114  101  103                .ASCII  /LAC/
012722  105  115  105                .ASCII  /EME/
```

F7


```
012725      116      124      040                    .ASCII  /NT /
012730      103      117      115                    .ASCII  /COM/
012733      120      114      105                    .ASCII  /PLE/
012736      124      111      117                    .ASCII  /TIO/
012741      116      000      000                    .ASCII  /N/<00><00>
012744   012224'                          P.AFO:    .WORD   P.AFP
012746   012246'                                    .WORD   P.AFQ
012750   012270'                                    .WORD   P.AFR
012752   012310'                                    .WORD   P.AFS
012754   012350'                                    .WORD   P.AFT
012756   012376'                                    .WORD   P.AFU
012760   012420'                                    .WORD   P.AFV
012762   012450'                                    .WORD   P.AFW
012764   012466'                                    .WORD   P.AFX
012766   012522'                                    .WORD   P.AFY
012770   012546'                                    .WORD   P.AFZ
012772   012564'                                    .WORD   P.AGA
012774   012630'                                    .WORD   P.AGB
012776   012656'                                    .WORD   P.AGC
013000   012700'                                    .WORD   P.AGD
013002      045      101      105          P.AGE:    .ASCII  /%AE/
013005      122      122      117                    .ASCII  /RRO/
013010      122      040      114                    .ASCII  /R L/
013013      117      107      040                    .ASCII  /OG /
013016      115      105      123                    .ASCII  /MES/
013021      123      101      107                    .ASCII  /SAG/
013024      105      040      122                    .ASCII  /E R/
013027      105      103      105                    .ASCII  /ECE/
013032      111      126      105                    .ASCII  /IVE/
013035      104      072      045                    .ASCII  /D:%/
013040      116      000                             .ASCII  /N/<00>
013042      045      101      052          P.AGG:    .ASCII  /%A*/
013045      040      103      117                    .ASCII  / CO/
013050      116      124      122                    .ASCII  /NTR/
013053      117      114      114                    .ASCII  /OLL/
013056      105      122      040                    .ASCII  /ER /
013061      105      122      122                    .ASCII  /ERR/
013064      117      122      045                    .ASCII  /OR%/
013067      116      000      000                    .ASCII  /N/<00><00>
013072      045      101      052          P.AGH:    .ASCII  /%A*/
013075      040      110      117                    .ASCII  / HO/
013100      123      124      040                    .ASCII  /ST /
013103      115      105      115                    .ASCII  /MEM/
013106      117      122      131                    .ASCII  /ORY/
013111      040      101      103                    .ASCII  / AC/
013114      103      105      123                    .ASCII  /CES/
013117      123      040      105                    .ASCII  /S E/
013122      122      122      117                    .ASCII  /RRO/
013125      122      045      116                    .ASCII  /R%N/
013130      000      000                             .ASCII  <00><00>
013132      045      101      052          P.AGI:    .ASCII  /%A*/
013135      040      104      111                    .ASCII  / DI/
013140      123      113      045                    .ASCII  /SK%/
```

```
013143  104  062  045          .ASCII  /D2%/
013146  101  040  055          .ASCII  /A -/
013151  040  104  111          .ASCII  / DI/
013154  123  113  040          .ASCII  /SK /
013157  124  122  101          .ASCII  /TRA/
013162  116  123  106          .ASCII  /NSF/
013165  105  122  040          .ASCII  /ER /
013170  105  122  122          .ASCII  /ERR/
013173  117  122  045          .ASCII  /OR%/
013176  116  000               .ASCII  /N/<00>
013200  045  101  052  P.AGJ:  .ASCII  /%A*/
013203  040  104  111          .ASCII  / DI/
013206  123  113  045          .ASCII  /SK%/
013211  104  062  045          .ASCII  /D2%/
013214  101  040  055          .ASCII  /A -/
013217  040  042  123          .ASCII  / "S/
013222  124  101  116          .ASCII  /TAN/
013225  104  101  122          .ASCII  /DAR/
013230  104  040  104          .ASCII  /D D/
013233  111  123  113          .ASCII  /ISK/
013236  040  111  116          .ASCII  / IN/
013241  124  105  122          .ASCII  /TER/
013244  103  117  116          .ASCII  /CON/
013247  116  105  103          .ASCII  /NEC/
013252  124  042  040          .ASCII  /T" /
013255  105  122  122          .ASCII  /ERR/
013260  117  122  045          .ASCII  /OR%/
013263  116  000  000          .ASCII  /N/<00><00>
013266  045  101  052  P.AGK:  .ASCII  /%A*/
013271  040  104  111          .ASCII  / DI/
013274  123  113  045          .ASCII  /SK%/
013277  104  062  045          .ASCII  /D2%/
013302  101  040  055          .ASCII  /A -/
013305  040  042  123          .ASCII  / "S/
013310  115  101  114          .ASCII  /MAL/
013313  114  040  104          .ASCII  /L D/
013316  111  123  113          .ASCII  /ISK/
013321  042  040  105          .ASCII  /" E/
013324  122  122  117          .ASCII  /RRO/
013327  122  045  116          .ASCII  /R%N/
013332  000  000               .ASCII  <00><00>
013334  045  101  052  P.AGL:  .ASCII  /%A*/
013337  040  104  111          .ASCII  / DI/
013342  123  113  045          .ASCII  /SK%/
013345  104  062  045          .ASCII  /D2%/
013350  101  040  055          .ASCII  /A -/
013353  040  042  102          .ASCII  / "B/
013356  101  104  040          .ASCII  /AD /
013361  102  114  117          .ASCII  /BLO/
013364  103  113  040          .ASCII  /CK /
013367  122  105  120          .ASCII  /REP/
013372  114  101  103          .ASCII  /LAC/
013375  105  115  105          .ASCII  /EME/
```

H7


```
013400     116     124     040                  .ASCII  /NT /
013403     101     124     124                  .ASCII  /ATT/
013406     105     115     120                  .ASCII  /EMP/
013411     124     042     045                  .ASCII  /T"%/
013414     116     000                          .ASCII  /N/<00>
013416  013042'                     P.AGF:      .WORD   P.AGG
013420  013072'                                 .WORD   P.AGH
013422  013132'                                 .WORD   P.AGI
013424  013200'                                 .WORD   P.AGJ
013426  013266'                                 .WORD   P.AGK
013430  013334'                                 .WORD   P.AGL
013432     045     116     045     P.AGM:      .ASCII  /%N%/
013435     101     052     040                 .ASCII  /A* /
013440     123     101     072                 .ASCII  /SA:/
013443     040     045     117                 .ASCII  / %O/
013446     066     000                         .ASCII  /6/<00>
013450     045     116     045     P.AGN:      .ASCII  /%N%/
013453     101     052     040                 .ASCII  /A* /
013456     123     124     101                 .ASCII  /STA/
013461     124     125     123                 .ASCII  /TUS/
013464     040     103     117                 .ASCII  / CO/
013467     104     105     072                 .ASCII  /DE:/
013472     040     045     117                 .ASCII  / %O/
013475     062     000     000                 .ASCII  /2/<00><00>
013500     045     117     064     P.AGO:      .ASCII  /%04/
013503     000                                 .ASCII  <00>
013504     045     116     045     P.AGP:      .ASCII  /%N%/
013507     101     052     040                 .ASCII  /A* /
013512     123     125     102                 .ASCII  /SUB/
013515     137     103     117                 .ASCII  / CO/
013520     104     105     072                 .ASCII  /DE:/
013523     040     000     000                 .ASCII  / /<00><00>
013526     045     116     045     P.AGQ:      .ASCII  /%N%/
013531     101     052     040                 .ASCII  /A* /
013534     103     117     115                 .ASCII  /COM/
013537     115     101     116                 .ASCII  /MAN/
013542     104     072     040                 .ASCII  /D: /
013545     000                                 .ASCII  <00>
013546     045     101     122     P.AGR:      .ASCII  /%AR/
013551     105     101     104                 .ASCII  /EAD/
013554     000     000                         .ASCII  <00><00>
013556     045     101     127     P.AGS:      .ASCII  /%AW/
013561     122     111     124                 .ASCII  /RIT/
013564     105     000                         .ASCII  /E/<00>
013566     045     101     055     P.AGT:      .ASCII  /%A-/
013571     103     117     115                 .ASCII  /COM/
013574     120     101     122                 .ASCII  /PAR/
013577     105     000     000                 .ASCII  /E/<00><00>
013602     045     101     117     P.AGU:      .ASCII  /%AO/
013605     116     114     111                 .ASCII  /NLI/
013610     116     105     000                 .ASCII  /NE/<00>
013613     000                                 .ASCII  <00>
013614     045     101     101     P.AGV:      .ASCII  /%AA/
```

```
013617  103  103  105                .ASCII  /CCE/
013622  123  123  000                .ASCII  /SS/<00>
013625  000                         .ASCII  <00>
013626  045  117  063      P.AGW:    .ASCII  /%03/
013631  000                         .ASCII  <00>
013632  045  116  045      P.AGX:    .ASCII  /%N%/
013635  101  052  040                .ASCII  /A* /
013640  102  101  104                .ASCII  /BAD/
013643  040  102  114                .ASCII  / BL/
013646  117  103  113                .ASCII  /OCK/
013651  072  040  045                .ASCII  /: %/
013654  117  066  045                .ASCII  /O6%/
013657  101  040  045                .ASCII  /A %/
013662  117  066  045                .ASCII  /O6%/
013665  101  040  040                .ASCII  /A  /
013670  050  117  103                .ASCII  /(OC/
013673  124  101  114                .ASCII  /TAL/
013676  051  000                    .ASCII  /)/<00>
013700  045  116  045      P.AGY:    .ASCII  /%N%/
013703  101  052  040                .ASCII  /A* /
013706  061  123  124                .ASCII  /1ST/
013711  040  102  101                .ASCII  / BA/
013714  104  040  102                .ASCII  /D B/
013717  114  117  103                .ASCII  /LOC/
013722  113  072  040                .ASCII  /K: /
013725  045  117  066                .ASCII  /%O6/
013730  045  101  040                .ASCII  /%A /
013733  045  117  066                .ASCII  /%O6/
013736  045  101  040                .ASCII  /%A /
013741  040  050  117                .ASCII  / (O/
013744  103  124  101                .ASCII  /CTA/
013747  114  051  000                .ASCII  /L)/<00>
013752  045  116  045      P.AGZ:    .ASCII  /%N%/
013755  101  052  040                .ASCII  /A* /
013760  102  101  104                .ASCII  /BAD/
013763  040  102  114                .ASCII  / BL/
013766  117  103  113                .ASCII  /OCK/
013771  040  122  105                .ASCII  / RE/
013774  120  114  101                .ASCII  /PLA/
013777  103  105  104                .ASCII  /CED/
014002  072  040  045                .ASCII  /: %/
014005  117  066  045                .ASCII  /O6%/
014010  101  040  045                .ASCII  /A %/
014013  117  066  045                .ASCII  /O6%/
014016  101  040  040                .ASCII  /A  /
014021  050  117  103                .ASCII  /(OC/
014024  124  101  114                .ASCII  /TAL/
014027  051  000  000                .ASCII  /)/<00><00>
014032  045  116  045      P.AHA:    .ASCII  /%N%/
014035  101  052  040                .ASCII  /A* /
014040  114  102  116                .ASCII  /LBN/
014043  072  040  045                .ASCII  /: %/
014046  117  066  045                .ASCII  /O6%/
```

J7


```
014051  101  040  045                .ASCII  /A %/
014054  117  066  045                .ASCII  /O6%/
014057  101  040  040                .ASCII  /A  /
014062  050  117  103                .ASCII  /(OC/
014065  124  101  114                .ASCII  /TAL/
014070  051  000                     .ASCII  /)/<00>
014072  045  116  045     P.AHB:     .ASCII  /%N%/
014075  101  052  040                .ASCII  /A* /
014100  120  102  116                .ASCII  /PBN/
014103  072  040  045                .ASCII  /: %/
014106  117  066  045                .ASCII  /O6%/
014111  101  040  045                .ASCII  /A %/
014114  117  066  045                .ASCII  /O6%/
014117  101  040  040                .ASCII  /A  /
014122  050  117  103                .ASCII  /(OC/
014125  124  101  114                .ASCII  /TAL/
014130  051  000                     .ASCII  /)/<00>
014132  045  116  045     P.AHC:     .ASCII  /%N%/
014135  101  052  040                .ASCII  /A* /
014140  114  102  116                .ASCII  /LBN/
014143  040  122  105                .ASCII  / RE/
014146  101  104  072                .ASCII  /AD:/
014151  040  045  117                .ASCII  / %O/
014154  066  045  101                .ASCII  /6%A/
014157  040  045  117                .ASCII  / %O/
014162  066  045  101                .ASCII  /6%A/
014165  040  040  050                .ASCII  /  (/
014170  117  103  124                .ASCII  /OCT/
014173  101  114  051                .ASCII  /AL)/
014176  000  000                     .ASCII  <00><00>
014200  045  116  045     P.AHD:     .ASCII  /%N%/
014203  101  052  040                .ASCII  /A* /
014206  114  102  116                .ASCII  /LBN/
014211  040  127  122                .ASCII  / WR/
014214  111  124  124                .ASCII  /ITT/
014217  105  116  072                .ASCII  /EN:/
014222  040  045  117                .ASCII  / %O/
014225  066  045  101                .ASCII  /6%A/
014230  040  045  117                .ASCII  / %O/
014233  066  045  101                .ASCII  /6%A/
014236  040  040  050                .ASCII  /  (/
014241  117  103  124                .ASCII  /OCT/
014244  101  114  051                .ASCII  /AL)/
014247  000                         .ASCII  <00>
014250  045  116  045     P.AHE:     .ASCII  /%N%/
014253  101  052  040                .ASCII  /A* /
014256  122  102  116                .ASCII  /RBN/
014261  072  040  045                .ASCII  /: %/
014264  117  066  045                .ASCII  /O6%/
014267  101  040  045                .ASCII  /A %/
014272  117  066  045                .ASCII  /O6%/
014275  101  040  040                .ASCII  /A  /
014300  050  117  103                .ASCII  /(OC/
```

K7


```
014303  124  101  114                    .ASCII  /TAL/
014306  051  000                         .ASCII  /)/<00>
014310  045  116  045          P.AHF:    .ASCII  /%N%/
014313  101  052  040                    .ASCII  /A* /
014316  102  131  124                    .ASCII  /BYT/
014321  105  040  103                    .ASCII  /E C/
014324  117  125  116                    .ASCII  /OUN/
014327  124  040  111                    .ASCII  /T I/
014332  116  040  103                    .ASCII  /N C/
014335  117  115  115                    .ASCII  /OMM/
014340  101  116  104                    .ASCII  /AND/
014343  072  040  045                    .ASCII  /: %/
014346  104  065  045                    .ASCII  /D5%/
014351  101  056  000                    .ASCII  /A./<00>
014354  045  116  045          P.AHG:    .ASCII  /%N%/
014357  101  052  040                    .ASCII  /A* /
014362  102  131  124                    .ASCII  /BYT/
014365  105  040  103                    .ASCII  /E C/
014370  117  125  116                    .ASCII  /OUN/
014373  124  040  111                    .ASCII  /T I/
014376  116  040  122                    .ASCII  /N R/
014401  105  101  104                    .ASCII  /EAD/
014404  040  103  117                    .ASCII  / CO/
014407  115  115  101                    .ASCII  /MMA/
014412  116  104  072                    .ASCII  /ND:/
014415  040  045  104                    .ASCII  / %D/
014420  065  045  101                    .ASCII  /5%A/
014423  056  000  000                    .ASCII  /./<00><00>
014426  045  116  045          P.AHH:    .ASCII  /%N%/
014431  101  052  040                    .ASCII  /A* /
014434  102  131  124                    .ASCII  /BYT/
014437  105  040  103                    .ASCII  /E C/
014442  117  125  116                    .ASCII  /OUN/
014445  124  040  111                    .ASCII  /T I/
014450  116  040  127                    .ASCII  /N W/
014453  122  111  124                    .ASCII  /RIT/
014456  105  040  103                    .ASCII  /E C/
014461  117  115  115                    .ASCII  /OMM/
014464  101  116  104                    .ASCII  /AND/
014467  072  040  045                    .ASCII  /: %/
014472  104  065  045                    .ASCII  /D5%/
014475  101  056  000                    .ASCII  /A./<00>
014500  045  116  045          P.AHI:    .ASCII  /%N%/
014503  101  052  040                    .ASCII  /A* /
014506  101  103  124                    .ASCII  /ACT/
014511  125  101  114                    .ASCII  /UAL/
014514  040  043  040                    .ASCII  / # /
014517  117  106  040                    .ASCII  /OF /
014522  102  131  124                    .ASCII  /BYT/
014525  105  123  040                    .ASCII  /ES /
014530  124  122  101                    .ASCII  /TRA/
014533  116  123  106                    .ASCII  /NSF/
014536  105  122  122                    .ASCII  /ERR/
```

L7


```
014541  105  104  072                    .ASCII  /ED:/
014544  040  045  104                    .ASCII  / %D/
014547  065  045  101                    .ASCII  /5%A/
014552  056  000                         .ASCII  /./<00>
014554  045  116  045          P.AHJ:    .ASCII  /%N%/
014557  101  052  040                    .ASCII  /A* /
014562  111  057  117                    .ASCII  /I/<57>/O/
014565  040  102  125                    .ASCII  / BU/
014570  106  106  105                    .ASCII  /FFE/
014573  122  040  101                    .ASCII  /R A/
014576  104  104  122                    .ASCII  /DDR/
014601  105  123  123                    .ASCII  /ESS/
014604  040  050  063                    .ASCII  / (3/
014607  062  040  142                    .ASCII  /2 b/
014612  151  164  163                    .ASCII  /its/
014615  051  072  040                    .ASCII  /): /
014620  045  117  066                    .ASCII  /%O6/
014623  045  101  040                    .ASCII  /%A /
014626  045  117  066                    .ASCII  /%O6/
014631  000                             .ASCII  <00>
014632  045  116  045          P.AHK:    .ASCII  /%N%/
014635  101  052  040                    .ASCII  /A* /
014640  111  057  117                    .ASCII  /I/<57>/O/
014643  040  102  125                    .ASCII  / BU/
014646  106  106  105                    .ASCII  /FFE/
014651  122  040  101                    .ASCII  /R A/
014654  104  104  122                    .ASCII  /DDR/
014657  105  123  123                    .ASCII  /ESS/
014662  040  106  117                    .ASCII  / FO/
014665  122  040  122                    .ASCII  /R R/
014670  105  101  104                    .ASCII  /EAD/
014673  040  050  063                    .ASCII  / (3/
014676  062  040  142                    .ASCII  /2 b/
014701  151  164  163                    .ASCII  /its/
014704  051  072  040                    .ASCII  /): /
014707  045  117  066                    .ASCII  /%O6/
014712  045  101  040                    .ASCII  /%A /
014715  045  117  066                    .ASCII  /%O6/
014720  000  000                         .ASCII  <00><00>
014722  045  116  045          P.AHL:    .ASCII  /%N%/
014725  101  052  040                    .ASCII  /A* /
014730  111  057  117                    .ASCII  /I/<57>/O/
014733  040  102  125                    .ASCII  / BU/
014736  106  106  105                    .ASCII  /FFE/
014741  122  040  101                    .ASCII  /R A/
014744  104  104  122                    .ASCII  /DDR/
014747  105  123  123                    .ASCII  /ESS/
014752  040  106  117                    .ASCII  / FO/
014755  122  040  127                    .ASCII  /R W/
014760  122  111  124                    .ASCII  /RIT/
014763  105  040  050                    .ASCII  /E (/
014766  063  062  040                    .ASCII  /32 /
014771  142  151  164                    .ASCII  /bit/
```

```
014774  163  051  072                    .ASCII  /s):/
014777  040  045  117                    .ASCII  / %O/
015002  066  045  101                    .ASCII  /6%A/
015005  040  045  117                    .ASCII  / %O/
015010  066  000                         .ASCII  /6/<00>
015012  045  116  045            P.AHM:  .ASCII  /%N%/
015015  101  103  117                    .ASCII  /ACO/
015020  116  124  105                    .ASCII  /NTE/
015023  116  124  123                    .ASCII  /NTS/
015026  040  117  106                    .ASCII  / OF/
015031  040  103  117                    .ASCII  / CO/
015034  115  115  101                    .ASCII  /MMA/
015037  116  104  057                    .ASCII  /ND/<57>
015042  122  105  123                    .ASCII  /RES/
015045  120  117  116                    .ASCII  /PON/
015050  123  105  040                    .ASCII  /SE /
015053  120  101  103                    .ASCII  /PAC/
015056  113  105  124                    .ASCII  /KET/
015061  040  123  101                    .ASCII  / SA/
015064  126  105  040                    .ASCII  /VE /
015067  101  122  105                    .ASCII  /ARE/
015072  101  072  045                    .ASCII  /A:%/
015075  116  000  000                    .ASCII  /N/<00><00>
015100  045  101  040            P.AHN:  .ASCII  /%A /
015103  045  117  066                    .ASCII  /%06/
015106  000  000                         .ASCII  <00><00>
015110  045  116  045            P.AHO:  .ASCII  /%N%/
015113  101  124  111                    .ASCII  /ATI/
015116  115  105  072                    .ASCII  /ME:/
015121  040  045  132                    .ASCII  / %Z/
015124  062  045  101                    .ASCII  /2%A/
015127  072  045  132                    .ASCII  /:%Z/
015132  062  045  101                    .ASCII  /2%A/
015135  040  110  117                    .ASCII  / HO/
015140  125  122  123                    .ASCII  /URS/
015143  045  116  000                    .ASCII  /%N/<00>
015146  045  116  045            P.AHP:  .ASCII  /%N%/
015151  101  052  040                    .ASCII  /A* /
015154  102  101  104                    .ASCII  /BAD/
015157  040  114  102                    .ASCII  / LB/
015162  116  072  040                    .ASCII  /N: /
015165  045  117  066                    .ASCII  /%06/
015170  045  101  040                    .ASCII  /%A /
015173  045  117  066                    .ASCII  /%06/
015176  045  101  040                    .ASCII  /%A /
015201  040  050  117                    .ASCII  / (O/
015204  103  124  101                    .ASCII  /CTA/
015207  114  051  000                    .ASCII  /L)/<00>
015212  045  116  045            P.AHQ:  .ASCII  /%N%/
015215  101  040  052                    .ASCII  /A */
015220  040  104  111                    .ASCII  / DI/
015223  123  113  040                    .ASCII  /SK /
015226  072  040  045                    .ASCII  /: %/
```

```
015231  104  062  000                .ASCII  /D2/<00>
015234  045  116  045      P.AHR:    .ASCII  /%N%/
015237  101  104  102                .ASCII  /AD3/
015242  116  072  040                .ASCII  /N: /
015245  045  104  065                .ASCII  /%D5/
015250  045  101  056                .ASCII  /%A./
015253  040  050  117                .ASCII  / (O/
015256  103  124  040                .ASCII  /CT /
015261  045  117  066                .ASCII  /%O6/
015264  045  101  051                .ASCII  /%A)/
015267  000                        .ASCII  <00>
015270  045  116  045      P.AHS:    .ASCII  /%N%/
015273  101  102  131                .ASCII  /ABY/
015276  124  105  040                .ASCII  /TE /
015301  116  125  115                .ASCII  /NUM/
015304  102  105  122                .ASCII  /BER/
015307  072  040  045                .ASCII  /: %/
015312  104  063  000                .ASCII  /D3/<00>
015315  000                        .ASCII  <00>
015316  045  116  045      P.AHT:    .ASCII  /%N%/
015321  101  122  101                .ASCII  /ARA/
015324  116  104  117                .ASCII  /NDO/
015327  115  040  127                .ASCII  /M W/
015332  122  111  124                .ASCII  /RIT/
015335  124  105  116                .ASCII  /TEN/
015340  040  127  117                .ASCII  / WO/
015343  122  104  040                .ASCII  /RD /
015346  072  045  102                .ASCII  /:%B/
015351  061  066  000                .ASCII  /16/<00>
015354  045  116  045      P.AHU:    .ASCII  /%N%/
015357  101  122  101                .ASCII  /ARA/
015362  116  104  117                .ASCII  /NDO/
015365  115  040  122                .ASCII  /M R/
015370  105  101  104                .ASCII  /EAD/
015373  040  127  117                .ASCII  / WO/
015376  122  104  040                .ASCII  /RD /
015401  142  151  156                .ASCII  /bin/
015404  072  045  102                .ASCII  /:%B/
015407  061  066  045                .ASCII  /16%/
015412  101  040  157                .ASCII  /A o/
015415  143  164  072                .ASCII  /ct:/
015420  045  117  066                .ASCII  /%O6/
015423  000                        .ASCII  <00>
015424  045  116  045      P.AHV:    .ASCII  /%N%/
015427  101  104  125                .ASCII  /ADU/
015432  120  114  111                .ASCII  /PLI/
015435  103  101  124                .ASCII  /CAT/
015440  105  040  125                .ASCII  /E U/
015443  116  111  124                .ASCII  /NIT/
015446  072  045  104                .ASCII  /:%D/
015451  062  045  101                .ASCII  /2%A/
015454  040  101  124                .ASCII  / AT/
015457  040  111  120                .ASCII  / IP/
```

```
015462  072  040  045                .ASCII  /: %/
015465  117  066  000                .ASCII  /06/<00>
015470  045  116  045      P.AHW:    .ASCII  /%N%/
015473  101  115  117                .ASCII  /AMO/
015476  122  105  040                .ASCII  /RE /
015501  124  110  101                .ASCII  /THA/
015504  116  040  045                .ASCII  /N %/
015507  104  061  045                .ASCII  /D1%/
015512  101  040  104                .ASCII  /A D/
015515  111  106  106                .ASCII  /IFF/
015520  105  122  105                .ASCII  /ERE/
015523  116  124  040                .ASCII  /NT /
015526  111  120  040                .ASCII  /IP /
015531  101  104  104                .ASCII  /ADD/
015534  122  105  123                .ASCII  /RES/
015537  123  105  123                .ASCII  /SES/
015542  000  000                     .ASCII  <00><00>
015544  045  101  123      P.AHX:    .ASCII  /%AS/
015547  120  111  116                .ASCII  /PIN/
015552  055  104  117                .ASCII  /-DO/
015555  127  116  040                .ASCII  /WN /
015560  111  107  116                .ASCII  /IGN/
015563  117  122  105                .ASCII  /ORE/
015566  104  000                     .ASCII  /D/<00>
015570  045  101  123      P.AHY:    .ASCII  /%AS/
015573  124  111  114                .ASCII  /TIL/
015576  114  040  103                .ASCII  /L C/
015601  117  116  116                .ASCII  /ONN/
015604  105  103  124                .ASCII  /ECT/
015607  105  104  000                .ASCII  /ED/<00>
015612  045  101  104      P.AHZ:    .ASCII  /%AD/
015615  125  120  114                .ASCII  /UPL/
015620  111  103  101                .ASCII  /ICA/
015623  124  105  040                .ASCII  /TE /
015626  125  116  111                .ASCII  /UNI/
015631  124  040  116                .ASCII  /T N/
015634  125  115  102                .ASCII  /UMB/
015637  105  122  000                .ASCII  /ER/<00>
015642  045  101  101      P.AIA:    .ASCII  /%AA/
015645  114  122  105                .ASCII  /LRE/
015650  101  104  131                .ASCII  /ADY/
015653  040  117  116                .ASCII  / ON/
015656  114  111  116                .ASCII  /LIN/
015661  105  000  000                .ASCII  /E/<00><00>
015664  045  101  123      P.AIB:    .ASCII  /%AS/
015667  124  111  114                .ASCII  /TIL/
015672  114  040  117                .ASCII  /L O/
015675  116  114  111                .ASCII  /NLI/
015700  116  105  000                .ASCII  /NE/<00>
015703  000                         .ASCII  <00>
015704  045  101  111      P.AIC:    .ASCII  /%AI/
015707  116  103  117                .ASCII  /NCO/
015712  115  120  114                .ASCII  /MPL/
```

```
015715  105  124  105                .ASCII  /ETE/
015720  040  122  105                .ASCII  / RE/
015723  120  114  101                .ASCII  /PLA/
015726  103  105  115                .ASCII  /CEM/
015731  105  116  124                .ASCII  /ENT/
015734  000  000                     .ASCII  <00><00>
015736  045  101  111      P.AID:    .ASCII  /%AI/
015741  116  126  101                .ASCII  /NVA/
015744  114  111  104                .ASCII  /LID/
015747  040  122  103                .ASCII  / RC/
015752  124  000                     .ASCII  /T/<00>
015754  045  101  125      P.AIE:    .ASCII  /%AU/
015757  116  111  124                .ASCII  /NIT/
015762  040  125  116                .ASCII  / UN/
015765  113  116  117                .ASCII  /KNO/
015770  127  116  040                .ASCII  /WN /
015773  117  122  040                .ASCII  /OR /
015776  117  116  114                .ASCII  /ONL/
016001  111  116  105                .ASCII  /INE/
016004  040  124  117                .ASCII  / TO/
016007  040  101  116                .ASCII  / AN/
016012  117  124  110                .ASCII  /OTH/
016015  105  122  040                .ASCII  /ER /
016020  103  117  116                .ASCII  /CON/
016023  124  122  117                .ASCII  /TRO/
016026  114  114  105                .ASCII  /LLE/
016031  122  000  000                .ASCII  /R/<00><00>
016034  045  101  116      P.AIF:    .ASCII  /%AN/
016037  117  040  126                .ASCII  /O V/
016042  117  114  125                .ASCII  /OLU/
016045  115  105  040                .ASCII  /ME /
016050  115  117  125                .ASCII  /MOU/
016053  116  124  105                .ASCII  /NTE/
016056  104  040  117                .ASCII  /D O/
016061  122  040  104                .ASCII  /R D/
016064  122  111  126                .ASCII  /RIV/
016067  105  040  104                .ASCII  /E D/
016072  111  123  101                .ASCII  /ISA/
016075  102  114  105                .ASCII  /BLE/
016100  104  040  102                .ASCII  /D B/
016103  131  040  123                .ASCII  /Y S/
016106  127  111  124                .ASCII  /WIT/
016111  103  110  000                .ASCII  /CH/<00>
016114  045  101  125      P.AIG:    .ASCII  /%AU/
016117  116  111  124                .ASCII  /NIT/
016122  040  111  116                .ASCII  / IN/
016125  117  120  105                .ASCII  /OPE/
016130  122  101  124                .ASCII  /RAT/
016133  111  126  105                .ASCII  /IVE/
016136  040  050  122                .ASCII  / (R/
016141  104  065  061                .ASCII  /D51/
016144  057  065  062                .ASCII  <57>/52/
016147  040  167  162                .ASCII  / wr/
```

D8


```
016152  151  164  145                .ASCII  /ite/
016155  040  146  141                .ASCII  / fa/
016160  165  154  164                .ASCII  /ult/
016163  051  000  000                .ASCII  /)/<00><00>
016166  045  101  125  P.AIH:        .ASCII  /%AU/
016171  116  111  124                .ASCII  /NIT/
016174  040  104  111                .ASCII  / DI/
016177  123  101  102                .ASCII  /SAB/
016202  114  105  104                .ASCII  /LED/
016205  040  102  131                .ASCII  / BY/
016210  040  106  111                .ASCII  / FI/
016213  105  114  104                .ASCII  /ELD/
016216  040  123  105                .ASCII  / SE/
016221  122  126  111                .ASCII  /RVI/
016224  103  105  040                .ASCII  /CE /
016227  117  122  040                .ASCII  /OR /
016232  111  116  124                .ASCII  /INT/
016235  105  122  116                .ASCII  /ERN/
016240  101  114  040                .ASCII  /AL /
016243  104  111  101                .ASCII  /DIA/
016246  107  116  117                .ASCII  /GNO/
016251  123  124  111                .ASCII  /STI/
016254  103  123  000                .ASCII  /CS/<00>
016257  000                        .ASCII  <00>
016260  045  101  042  P.AII:        .ASCII  /%A"/
016263  106  117  122                .ASCII  /FOR/
016266  103  105  104                .ASCII  /CED/
016271  040  105  122                .ASCII  / ER/
016274  122  117  122                .ASCII  /ROR/
016277  042  040  104                .ASCII  /" D/
016302  105  124  105                .ASCII  /ETE/
016305  103  124  105                .ASCII  /CTE/
016310  104  040  127                .ASCII  /D W/
016313  110  111  114                .ASCII  /HIL/
016316  105  040  101                .ASCII  /E A/
016321  103  103  105                .ASCII  /CCE/
016324  123  123  111                .ASCII  /SSI/
016327  116  107  040                .ASCII  /NG /
016332  106  103  124                .ASCII  /FCT/
016335  040  117  122                .ASCII  / OR/
016340  040  122  103                .ASCII  / RC/
016343  124  000  000                .ASCII  /T/<00><00>
016346  045  101  123  P.AIJ:        .ASCII  /%AS/
016351  105  103  124                .ASCII  /ECT/
016354  117  122  040                .ASCII  /OR /
016357  110  101  104                .ASCII  /HAD/
016362  040  102  105                .ASCII  / BE/
016365  105  116  040                .ASCII  /EN /
016370  127  122  111                .ASCII  /WRI/
016373  124  124  105                .ASCII  /TTE/
016376  116  040  127                .ASCII  /N W/
016401  111  124  110                .ASCII  /ITH/
016404  040  042  106                .ASCII  / "F/
```

```
016407  117  122  103                    .ASCII  /ORC/
016412  105  104  040                    .ASCII  /ED /
016415  105  122  122                    .ASCII  /ERR/
016420  117  122  042                    .ASCII  /OR"/
016423  040  115  117                    .ASCII  / MO/
016426  104  111  106                    .ASCII  /DIF/
016431  111  105  122                    .ASCII  /IER/
016434  000  000                         .ASCII  <00><00>
016436  045  101  106          P.AIK:    .ASCII  /%AF/
016441  103  124  040                    .ASCII  /CT /
016444  117  122  040                    .ASCII  /OR /
016447  122  103  124                    .ASCII  /RCT/
016452  040  125  116                    .ASCII  / UN/
016455  122  105  101                    .ASCII  /REA/
016460  104  101  102                    .ASCII  /DAB/
016463  114  105  040                    .ASCII  /LE /
016466  055  040  111                    .ASCII  /- I/
016471  116  126  101                    .ASCII  /NVA/
016474  114  111  104                    .ASCII  /LID/
016477  040  123  105                    .ASCII  / SE/
016502  103  124  117                    .ASCII  /CTO/
016505  122  040  110                    .ASCII  /R H/
016510  105  101  104                    .ASCII  /EAD/
016513  105  122  000                    .ASCII  /ER/<00>
016516  045  101  110          P.AIL:    .ASCII  /%AH/
016521  105  101  104                    .ASCII  /EAD/
016524  105  122  040                    .ASCII  /ER /
016527  103  117  115                    .ASCII  /COM/
016532  120  101  122                    .ASCII  /PAR/
016535  105  040  105                    .ASCII  /E E/
016540  122  122  117                    .ASCII  /RRO/
016543  122  040  050                    .ASCII  /R (/
016546  126  141  154                    .ASCII  /Val/
016551  151  144  040                    .ASCII  /id /
016554  150  145  141                    .ASCII  /hea/
016557  144  145  162                    .ASCII  /der/
016562  040  156  157                    .ASCII  / no/
016565  164  040  146                    .ASCII  /t f/
016570  157  165  156                    .ASCII  /oun/
016573  144  051  000                    .ASCII  /d)/<00>
016576  045  101  106          P.AIM:    .ASCII  /%AF/
016601  103  124  040                    .ASCII  /CT /
016604  117  122  040                    .ASCII  /OR /
016607  122  103  124                    .ASCII  /RCT/
016612  040  125  116                    .ASCII  / UN/
016615  122  105  101                    .ASCII  /REA/
016620  104  101  102                    .ASCII  /DAB/
016623  114  105  040                    .ASCII  /LE /
016626  055  040  104                    .ASCII  /- D/
016631  101  124  101                    .ASCII  /ATA/
016634  040  123  131                    .ASCII  / SY/
016637  116  103  040                    .ASCII  /NC /
016642  124  111  115                    .ASCII  /TIM/
```

F8


```
016645  105  117  125                    .ASCII  /EQU/
016650  124  000                         .ASCII  /T/<00>
016652  045  101  104          P.AIN:    .ASCII  /%AD/
016655  101  124  101                    .ASCII  /ATA/
016660  040  123  131                    .ASCII  / SY/
016663  116  103  040                    .ASCII  /NC /
016666  116  117  124                    .ASCII  /NOT/
016671  040  106  117                    .ASCII  / FO/
016674  125  116  104                    .ASCII  /UND/
016677  040  050  104                    .ASCII  / (D/
016702  141  164  141                    .ASCII  /ata/
016705  040  163  171                    .ASCII  / sy/
016710  156  143  040                    .ASCII  /nc /
016713  164  151  155                    .ASCII  /tim/
016716  145  157  165                    .ASCII  /equ/
016721  164  051  000                    .ASCII  /t)/<00>
016724  045  101  106          P.AIO:    .ASCII  /%AF/
016727  103  124  040                    .ASCII  /CT /
016732  117  122  040                    .ASCII  /OR /
016735  122  103  124                    .ASCII  /RCT/
016740  040  125  116                    .ASCII  / UN/
016743  122  105  101                    .ASCII  /REA/
016746  104  101  102                    .ASCII  /DAB/
016751  114  105  040                    .ASCII  /LE /
016754  055  040  125                    .ASCII  /- U/
016757  116  103  117                    .ASCII  /NCO/
016762  122  122  105                    .ASCII  /RRE/
016765  103  124  101                    .ASCII  /CTA/
016770  102  114  105                    .ASCII  /BLE/
016773  040  105  103                    .ASCII  / EC/
016776  103  040  105                    .ASCII  /C E/
017001  122  122  117                    .ASCII  /RRO/
017004  122  000                         .ASCII  /R/<00>
017006  045  101  125          P.AIP:    .ASCII  /%AU/
017011  116  103  117                    .ASCII  /NCO/
017014  122  122  105                    .ASCII  /RRE/
017017  103  124  101                    .ASCII  /CTA/
017022  102  114  105                    .ASCII  /BLE/
017025  040  105  103                    .ASCII  / EC/
017030  103  040  105                    .ASCII  /C E/
017033  122  122  117                    .ASCII  /RRO/
017036  122  000                         .ASCII  /R/<00>
017040  045  101  122          P.AIQ:    .ASCII  /%AR/
017043  103  124  040                    .ASCII  /CT /
017046  103  117  122                    .ASCII  /COR/
017051  122  125  120                    .ASCII  /RUP/
017054  124  105  104                    .ASCII  /TED/
017057  000                             .ASCII  <00>
017060  045  101  116          P.AIR:    .ASCII  /%AN/
017063  117  040  122                    .ASCII  /O R/
017066  105  120  114                    .ASCII  /EPL/
017071  101  103  105                    .ASCII  /ACE/
017074  115  105  116                    .ASCII  /MEN/
```

```
017077  124  040  102                .ASCII  /T B/
017102  114  117  103                .ASCII  /LOC/
017105  113  040  101                .ASCII  /K A/
017110  126  101  111                .ASCII  /VAI/
017113  114  101  102                .ASCII  /LAB/
017116  114  105  040                .ASCII  /LE /
017121  050  122  103                .ASCII  /(RC/
017124  124  040  146                .ASCII  /T f/
017127  165  154  154                .ASCII  /ull/
017132  051  000                    .ASCII  /)/<00>
017134  045  101  104     P.AIS:     .ASCII  /%AD/
017137  111  123  113                .ASCII  /ISK/
017142  040  116  117                .ASCII  / NO/
017145  124  040  106                .ASCII  /T F/
017150  117  122  115                .ASCII  /ORM/
017153  101  124  124                .ASCII  /ATT/
017156  105  104  040                .ASCII  /ED /
017161  127  111  124                .ASCII  /WIT/
017164  110  040  065                .ASCII  /H 5/
017167  061  062  040                .ASCII  /12 /
017172  102  131  124                .ASCII  /BYT/
017175  105  040  123                .ASCII  /E S/
017200  105  103  124                .ASCII  /ECT/
017203  117  122  123                .ASCII  /ORS/
017206  000  000                    .ASCII  <00><00>
017210  045  101  104     P.AIT:     .ASCII  /%AD/
017213  111  123  113                .ASCII  /ISK/
017216  040  116  117                .ASCII  / NO/
017221  124  040  106                .ASCII  /T F/
017224  117  122  115                .ASCII  /ORM/
017227  101  124  124                .ASCII  /ATT/
017232  105  104  040                .ASCII  /ED /
017235  117  122  040                .ASCII  /OR /
017240  106  103  124                .ASCII  /FCT/
017243  040  103  117                .ASCII  / CO/
017246  122  122  125                .ASCII  /RRU/
017251  120  124  105                .ASCII  /PTE/
017254  104  000                    .ASCII  /D/<00>
017256  045  101  117     P.AIU:     .ASCII  /%AO/
017261  116  105  040                .ASCII  /NE /
017264  123  131  115                .ASCII  /SYM/
017267  102  117  114                .ASCII  /BOL/
017272  040  105  103                .ASCII  / EC/
017275  103  040  105                .ASCII  /C E/
017300  122  122  117                .ASCII  /RRO/
017303  122  000  000                .ASCII  /R/<00><00>
017306  045  101  124     P.AIV:     .ASCII  /%AT/
017311  127  117  040                .ASCII  /WO /
017314  123  131  115                .ASCII  /SYM/
017317  102  117  114                .ASCII  /BOL/
017322  040  105  103                .ASCII  / EC/
017325  103  040  105                .ASCII  /C E/
017330  122  122  117                .ASCII  /RRO/
```

```
017333  122  000  000          .ASCII  /R/<00><00>
017336  045  101  124  P.AIW:   .ASCII  /%AT/
017341  110  122  105          .ASCII  /HRE/
017344  105  040  123          .ASCII  /E S/
017347  131  115  102          .ASCII  /YMB/
017352  117  114  040          .ASCII  /OL /
017355  105  103  103          .ASCII  /ECC/
017360  040  105  122          .ASCII  / ER/
017363  122  117  122          .ASCII  /ROR/
017366  000  000               .ASCII  <00><00>
017370  045  101  106  P.AIX:   .ASCII  /%AF/
017373  117  125  122          .ASCII  /OUR/
017376  040  123  131          .ASCII  / SY/
017401  115  102  117          .ASCII  /MBO/
017404  114  040  105          .ASCII  /L E/
017407  103  103  040          .ASCII  /CC /
017412  105  122  122          .ASCII  /ERR/
017415  117  122  000          .ASCII  /OR/<00>
017420  045  101  106  P.AIY:   .ASCII  /%AF/
017423  111  126  105          .ASCII  /IVE/
017426  040  123  131          .ASCII  / SY/
017431  115  102  117          .ASCII  /MBO/
017434  114  040  105          .ASCII  /L E/
017437  103  103  040          .ASCII  /CC /
017442  105  122  122          .ASCII  /ERR/
017445  117  122  000          .ASCII  /OR/<00>
017450  045  101  123  P.AIZ:   .ASCII  /%AS/
017453  111  130  040          .ASCII  /IX /
017456  123  131  115          .ASCII  /SYM/
017461  102  117  114          .ASCII  /BOL/
017464  040  105  103          .ASCII  / EC/
017467  103  040  105          .ASCII  /C E/
017472  122  122  117          .ASCII  /RRO/
017475  122  000  000          .ASCII  /R/<00><00>
017500  045  101  123  P.AJA:   .ASCII  /%AS/
017503  105  126  105          .ASCII  /EVE/
017506  116  040  123          .ASCII  /N S/
017511  131  115  102          .ASCII  /YMB/
017514  117  114  040          .ASCII  /OL /
017517  105  103  103          .ASCII  /ECC/
017522  040  105  122          .ASCII  / ER/
017525  122  117  122          .ASCII  /ROR/
017530  000  000               .ASCII  <00><00>
017532  045  101  105  P.AJB:   .ASCII  /%AE/
017535  111  107  110          .ASCII  /IGH/
017540  124  040  123          .ASCII  /T S/
017543  131  115  102          .ASCII  /YMB/
017546  117  114  040          .ASCII  /OL /
017551  105  103  103          .ASCII  /ECC/
017554  040  105  122          .ASCII  / ER/
017557  122  117  122          .ASCII  /ROR/
017562  000  000               .ASCII  <00><00>
017564  045  101  103  P.AJC:   .ASCII  /%AC/
```

```
017567  117  122  122                .ASCII  /ORR/
017572  105  103  124                .ASCII  /ECT/
017575  101  102  114                .ASCII  /ABL/
017600  105  040  105                .ASCII  /E E/
017603  122  122  117                .ASCII  /RRO/
017606  122  040  111                .ASCII  /R I/
017611  116  040  105                .ASCII  /N E/
017614  103  103  040                .ASCII  /CC /
017617  106  111  105                .ASCII  /FIE/
017622  114  104  000                .ASCII  /LD/<00>
017625  000                         .ASCII  <00>
017626  045  101  125       P.AJD:   .ASCII  /%AU/
017631  116  111  124                .ASCII  /NIT/
017634  040  123  117                .ASCII  / SO/
017637  106  124  127                .ASCII  /FTW/
017642  101  122  105                .ASCII  /ARE/
017645  040  127  122                .ASCII  / WR/
017650  111  124  105                .ASCII  /ITE/
017653  040  120  122                .ASCII  / PR/
017656  117  124  105                .ASCII  /OTE/
017661  103  124  105                .ASCII  /CTE/
017664  104  000                    .ASCII  /D/<00>
017666  045  101  125       P.AJE:   .ASCII  /%AU/
017671  116  111  124                .ASCII  /NIT/
017674  040  110  101                .ASCII  / HA/
017677  122  104  127                .ASCII  /RDW/
017702  101  122  105                .ASCII  /ARE/
017705  040  127  122                .ASCII  / WR/
017710  111  124  105                .ASCII  /ITE/
017713  040  120  122                .ASCII  / PR/
017716  117  124  105                .ASCII  /OTE/
017721  103  124  105                .ASCII  /CTE/
017724  104  000                    .ASCII  /D/<00>
017726  045  101  125       P.AJF:   .ASCII  /%AU/
017731  116  111  124                .ASCII  /NIT/
017734  040  104  101                .ASCII  / DA/
017737  124  101  040                .ASCII  /TA /
017742  123  101  106                .ASCII  /SAF/
017745  105  124  131                .ASCII  /ETY/
017750  040  127  122                .ASCII  / WR/
017753  111  124  105                .ASCII  /ITE/
017756  040  120  122                .ASCII  / PR/
017761  117  124  105                .ASCII  /OTE/
017764  103  124  105                .ASCII  /CTE/
017767  104  000  000                .ASCII  /D/<00><00>
017772  045  101  117       P.AJG:   .ASCII  /%AO/
017775  104  104  040                .ASCII  /DD /
020000  124  122  101                .ASCII  /TRA/
020003  116  123  106                .ASCII  /NSF/
020006  105  122  040                .ASCII  /ER /
020011  101  104  104                .ASCII  /ADD/
020014  122  105  123                .ASCII  /RES/
020017  123  000  000                .ASCII  /S/<00><00>
```

```
020022  045  101  117        P.AJH:  .ASCII  /%AO/
020025  104  104  040                .ASCII  /DD /
020030  102  131  124                .ASCII  /BYT/
020033  105  040  103                .ASCII  /E C/
020036  117  125  116                .ASCII  /OUN/
020041  124  000  000                .ASCII  /T/<00><00>
020044  045  101  116        P.AJI:  .ASCII  /%AN/
020047  117  116  055                .ASCII  /ON-/
020052  105  130  111                .ASCII  /EXI/
020055  123  124  105                .ASCII  /STE/
020060  116  124  040                .ASCII  /NT /
020063  110  117  123                .ASCII  /HOS/
020066  124  040  115                .ASCII  /T M/
020071  105  115  117                .ASCII  /EMO/
020074  122  131  000                .ASCII  /RY/<00>
020077  000                        .ASCII  <00>
020100  045  101  110        P.AJJ:  .ASCII  /%AH/
020103  117  123  124                .ASCII  /OST/
020106  040  115  105                .ASCII  / ME/
020111  115  117  122                .ASCII  /MOR/
020114  131  040  120                .ASCII  /Y P/
020117  101  122  111                .ASCII  /ARI/
020122  124  131  040                .ASCII  /TY /
020125  105  122  122                .ASCII  /ERR/
020130  117  122  000                .ASCII  /OR/<00>
020133  000                        .ASCII  <00>
020134  045  101  103        P.AJK:  .ASCII  /%AC/
020137  117  115  115                .ASCII  /OMM/
020142  101  116  104                .ASCII  /AND/
020145  040  124  111                .ASCII  / TI/
020150  115  117  125                .ASCII  /MOU/
020153  124  040  117                .ASCII  /T O/
020156  122  040  122                .ASCII  /R R/
020161  105  124  122                .ASCII  /ETR/
020164  131  040  114                .ASCII  /Y L/
020167  111  115  111                .ASCII  /IMI/
020172  124  040  105                .ASCII  /T E/
020175  130  103  105                .ASCII  /XCE/
020200  105  104  105                .ASCII  /EDE/
020203  104  000  000                .ASCII  /D/<00><00>
020206  045  101  123        P.AJL:  .ASCII  /%AS/
020211  105  122  111                .ASCII  /ERI/
020214  101  114  111                .ASCII  /ALI/
020217  132  105  122                .ASCII  /ZER/
020222  057  104  105                .ASCII  <57>/DE/
020225  123  105  122                .ASCII  /SER/
020230  111  101  114                .ASCII  /IAL/
020233  111  132  105                .ASCII  /IZE/
020236  122  040  117                .ASCII  /R O/
020241  126  105  122                .ASCII  /VER/
020244  122  125  116                .ASCII  /RUN/
020247  040  117  122                .ASCII  / OR/
020252  040  125  116                .ASCII  / UN/
```

K8


```
020255  104  105  122                .ASCII  /DER/
020260  122  125  116                .ASCII  /RUN/
020263  000                         .ASCII  <00>
020264  045  101  042       P.AJM:   .ASCII  /%A"/
020267  105  122  122                .ASCII  /ERR/
020272  117  122  040                .ASCII  /OR /
020275  104  105  124                .ASCII  /DET/
020300  105  103  124                .ASCII  /ECT/
020303  111  117  116                .ASCII  /ION/
020306  040  103  117                .ASCII  / CO/
020311  104  105  042                .ASCII  /DE"/
020314  040  105  122                .ASCII  / ER/
020317  122  117  122                .ASCII  /ROR/
020322  000  000                     .ASCII  <00><00>
020324  045  101  111       P.AJN:   .ASCII  /%AI/
020327  116  103  117                .ASCII  /NCO/
020332  116  123  111                .ASCII  /NSI/
020335  123  124  105                .ASCII  /STE/
020340  116  124  040                .ASCII  /NT /
020343  111  116  124                .ASCII  /INT/
020346  105  122  116                .ASCII  /ERN/
020351  101  114  040                .ASCII  /AL /
020354  104  101  124                .ASCII  /DAT/
020357  101  040  123                .ASCII  /A S/
020362  124  122  125                .ASCII  /TRU/
020365  103  124  125                .ASCII  /CTU/
020370  122  105  000                .ASCII  /RE/<00>
020373  000                         .ASCII  <00>
020374  045  101  104       P.AJO:   .ASCII  /%AD/
020377  122  111  126                .ASCII  /RIV/
020402  105  040  103                .ASCII  /E C/
020405  117  115  115                .ASCII  /OMM/
020410  101  116  104                .ASCII  /AND/
020413  040  124  111                .ASCII  / TI/
020416  115  105  117                .ASCII  /MEO/
020421  125  124  040                .ASCII  /UT /
020424  050  116  157                .ASCII  /(No/
020427  040  162  145                .ASCII  / re/
020432  163  160  157                .ASCII  /spo/
020435  156  163  145                .ASCII  /nse/
020440  040  157  162                .ASCII  / or/
020443  040  163  145                .ASCII  / se/
020446  145  153  040                .ASCII  /ek /
020451  151  156  143                .ASCII  /inc/
020454  157  155  160                .ASCII  /omp/
020457  154  145  164                .ASCII  /let/
020462  145  051  000                .ASCII  /e)/<00>
020465  000                         .ASCII  <00>
020466  045  101  103       P.AJP:   .ASCII  /%AC/
020471  117  116  124                .ASCII  /ONT/
020474  122  117  114                .ASCII  /ROL/
020477  114  105  122                .ASCII  /LER/
020502  040  104  105                .ASCII  / DE/
```

```
020505  124  105  103                    .ASCII  /TEC/
020510  124  105  104                    .ASCII  /TED/
020513  040  124  122                    .ASCII  / TR/
020516  101  116  123                    .ASCII  /ANS/
020521  115  111  123                    .ASCII  /MIS/
020524  123  111  117                    .ASCII  /SIO/
020527  116  040  117                    .ASCII  /N O/
020532  122  040  120                    .ASCII  /R P/
020535  122  117  124                    .ASCII  /ROT/
020540  117  103  117                    .ASCII  /OCO/
020543  114  040  105                    .ASCII  /L E/
020546  122  122  117                    .ASCII  /RRO/
020551  122  000  000                    .ASCII  /R/<00><00>
020554  045  101  120          P.AJQ:    .ASCII  /%AP/
020557  117  123  111                    .ASCII  /OSI/
020562  124  111  117                    .ASCII  /TIO/
020565  116  040  105                    .ASCII  /N E/
020570  122  122  117                    .ASCII  /RRO/
020573  122  040  050                    .ASCII  /R (/
020576  115  151  163                    .ASCII  /Mis/
020601  055  163  145                    .ASCII  /-se/
020604  145  153  051                    .ASCII  /ek)/
020607  000                             .ASCII  <00>
020610  045  101  114          P.AJR:    .ASCII  /%AL/
020613  117  123  124                    .ASCII  /OST/
020616  040  122  105                    .ASCII  / RE/
020621  101  104  057                    .ASCII  /AD/<57>
020624  127  122  111                    .ASCII  /WRI/
020627  124  105  040                    .ASCII  /TE /
020632  122  105  101                    .ASCII  /REA/
020635  104  131  040                    .ASCII  /DY /
020640  104  125  122                    .ASCII  /DUR/
020643  111  116  107                    .ASCII  /ING/
020646  057  102  105                    .ASCII  <57>/BE/
020651  124  127  105                    .ASCII  /TWE/
020654  105  116  040                    .ASCII  /EN /
020657  124  122  101                    .ASCII  /TRA/
020662  116  123  106                    .ASCII  /NSF/
020665  105  122  123                    .ASCII  /ERS/
020670  000  000                        .ASCII  <00><00>
020672  045  101  104          P.AJS:    .ASCII  /%AD/
020675  122  111  126                    .ASCII  /RIV/
020700  105  040  103                    .ASCII  /E C/
020703  114  117  103                    .ASCII  /LOC/
020706  113  040  104                    .ASCII  /K D/
020711  122  117  120                    .ASCII  /ROP/
020714  117  125  124                    .ASCII  /OUT/
020717  000                             .ASCII  <00>
020720  045  101  114          P.AJT:    .ASCII  /%AL/
020723  117  123  124                    .ASCII  /OST/
020726  040  122  105                    .ASCII  / RE/
020731  103  105  111                    .ASCII  /CEI/
020734  126  105  122                    .ASCII  /VER/
```

```
020737  040  122  105                    .ASCII  / RE/
020742  101  104  131                    .ASCII  /ADY/
020745  040  102  105                    .ASCII  / BE/
020750  124  127  105                    .ASCII  /TWE/
020753  105  116  040                    .ASCII  /EN /
020756  123  105  103                    .ASCII  /SEC/
020761  124  117  122                    .ASCII  /TOR/
020764  123  000                        .ASCII  /S/<00>
020766  045  101  104          P.AJU:    .ASCII  /%AD/
020771  122  111  126                    .ASCII  /RIV/
020774  105  040  104                    .ASCII  /E D/
020777  105  124  105                    .ASCII  /ETE/
021002  103  124  105                    .ASCII  /CTE/
021005  104  040  105                    .ASCII  /D E/
021010  122  122  117                    .ASCII  /RRO/
021013  122  000  000                    .ASCII  /R/<00><00>
021016  045  101  103          P.AJV:    .ASCII  /%AC/
021021  117  116  124                    .ASCII  /ONT/
021024  122  117  114                    .ASCII  /ROL/
021027  114  105  122                    .ASCII  /LER/
021032  040  104  105                    .ASCII  / DE/
021035  124  105  103                    .ASCII  /TEC/
021040  124  105  104                    .ASCII  /TED/
021043  040  120  125                    .ASCII  / PU/
021046  114  123  105                    .ASCII  /LSE/
021051  040  117  122                    .ASCII  / OR/
021054  040  123  124                    .ASCII  / ST/
021057  101  124  105                    .ASCII  /ATE/
021062  040  120  101                    .ASCII  / PA/
021065  122  111  124                    .ASCII  /RIT/
021070  131  040  105                    .ASCII  /Y E/
021073  122  122  117                    .ASCII  /RRO/
021076  122  000                        .ASCII  /R/<00>
021100  045  101  102          P.AJW:    .ASCII  /%AB/
021103  101  104  040                    .ASCII  /AD /
021106  102  114  117                    .ASCII  /BLO/
021111  103  113  040                    .ASCII  /CK /
021114  123  125  103                    .ASCII  /SUC/
021117  103  105  123                    .ASCII  /CES/
021122  123  106  125                    .ASCII  /SFU/
021125  114  114  131                    .ASCII  /LLY/
021130  040  122  105                    .ASCII  / RE/
021133  120  114  101                    .ASCII  /PLA/
021136  103  105  104                    .ASCII  /CED/
021141  000                            .ASCII  <00>
021142  045  101  102          P.AJX:    .ASCII  /%AB/
021145  114  117  103                    .ASCII  /LOC/
021150  113  040  126                    .ASCII  /K V/
021153  105  122  111                    .ASCII  /ERI/
021156  106  111  105                    .ASCII  /FIE/
021161  104  040  117                    .ASCII  /D O/
021164  113  040  055                    .ASCII  /K -/
021167  055  040  116                    .ASCII  /- N/
```

```
021172  117  124  040                    .ASCII  /OT /
021175  101  040  102                    .ASCII  /A B/
021200  101  104  040                    .ASCII  /AD /
021203  102  114  117                    .ASCII  /BLO/
021206  103  113  000                    .ASCII  /CK/<00>
021211  000                             .ASCII  <00>
021212  045  101  122          P.AJY:    .ASCII  /%AR/
021215  105  120  114                    .ASCII  /EPL/
021220  101  103  105                    .ASCII  /ACE/
021223  115  105  116                    .ASCII  /MEN/
021226  124  040  106                    .ASCII  /T F/
021231  101  111  114                    .ASCII  /AIL/
021234  125  122  105                    .ASCII  /URE/
021237  000                             .ASCII  <00>
021240  045  101  103          P.AKA:    .ASCII  /%AC/
021243  117  116  124                    .ASCII  /ONT/
021246  122  117  114                    .ASCII  /ROL/
021251  114  105  122                    .ASCII  /LER/
021254  040  124  111                    .ASCII  / TI/
021257  115  105  117                    .ASCII  /MEO/
021262  125  124  000                    .ASCII  /UT/<00>
021265  000                             .ASCII  <00>
021266  045  101  105          P.AKB:    .ASCII  /%AE/
021271  116  126  105                    .ASCII  /NVE/
021274  114  117  120                    .ASCII  /LOP/
021277  105  057  120                    .ASCII  /E/<57>/P/
021302  101  103  113                    .ASCII  /ACK/
021305  105  124  040                    .ASCII  /ET /
021310  122  105  101                    .ASCII  /REA/
021313  104  040  105                    .ASCII  /D E/
021316  122  122  117                    .ASCII  /RRO/
021321  122  040  050                    .ASCII  /R (/
021324  120  141  162                    .ASCII  /Par/
021327  151  164  171                    .ASCII  /ity/
021332  040  157  162                    .ASCII  / or/
021335  040  164  151                    .ASCII  / ti/
021340  155  145  157                    .ASCII  /meo/
021343  165  164  051                    .ASCII  /ut)/
021346  000  000                        .ASCII  <00><00>
021350  045  101  105          P.AKC:    .ASCII  /%AE/
021353  116  126  105                    .ASCII  /NVE/
021356  114  117  120                    .ASCII  /LOP/
021361  105  057  120                    .ASCII  /E/<57>/P/
021364  101  103  113                    .ASCII  /ACK/
021367  105  124  040                    .ASCII  /ET /
021372  127  122  111                    .ASCII  /WRI/
021375  124  105  040                    .ASCII  /TE /
021400  105  122  122                    .ASCII  /ERR/
021403  117  122  040                    .ASCII  /OR /
021406  050  120  141                    .ASCII  /(Pa/
021411  162  151  164                    .ASCII  /rit/
021414  171  040  157                    .ASCII  /y o/
021417  162  040  164                    .ASCII  /r t/
```

B9


```
021422   151   155   145                .ASCII  /ime/
021425   157   165   164                .ASCII  /out/
021430   051   000                      .ASCII  /)/<00>
021432   045   101   103        P.AKD:  .ASCII  /%AC/
021435   117   116   124                .ASCII  /ONT/
021440   122   117   114                .ASCII  /ROL/
021443   114   105   122                .ASCII  /LER/
021446   040   122   117                .ASCII  / RO/
021451   115   040   101                .ASCII  /M A/
021454   116   104   040                .ASCII  /ND /
021457   122   101   115                .ASCII  /RAM/
021462   040   120   101                .ASCII  / PA/
021465   122   111   124                .ASCII  /RIT/
021470   131   040   105                .ASCII  /Y E/
021473   122   122   117                .ASCII  /RRO/
021476   122   000                      .ASCII  /R/<00>
021500   045   101   103        P.AKE:  .ASCII  /%AC/
021503   117   116   124                .ASCII  /ONT/
021506   122   117   114                .ASCII  /ROL/
021511   114   105   122                .ASCII  /LER/
021514   040   122   101                .ASCII  / RA/
021517   115   040   120                .ASCII  /M P/
021522   101   122   111                .ASCII  /ARI/
021525   124   131   040                .ASCII  /TY /
021530   105   122   122                .ASCII  /ERR/
021533   117   122   000                .ASCII  /OR/<00>
021536   045   101   103        P.AKF:  .ASCII  /%AC/
021541   117   116   124                .ASCII  /ONT/
021544   122   117   114                .ASCII  /ROL/
021547   114   105   122                .ASCII  /LER/
021552   040   122   117                .ASCII  / RO/
021555   115   040   120                .ASCII  /M P/
021560   101   122   111                .ASCII  /ARI/
021563   124   131   040                .ASCII  /TY /
021566   105   122   122                .ASCII  /ERR/
021571   117   122   000                .ASCII  /OR/<00>
021574   045   101   122        P.AKG:  .ASCII  /%AR/
021577   111   116   107                .ASCII  /ING/
021602   040   122   105                .ASCII  / RE/
021605   101   104   040                .ASCII  /AD /
021610   105   122   122                .ASCII  /ERR/
021613   117   122   040                .ASCII  /OR /
021616   050   120   141                .ASCII  /(Pa/
021621   162   151   164                .ASCII  /rit/
021624   171   040   157                .ASCII  /y o/
021627   162   040   164                .ASCII  /r t/
021632   151   155   145                .ASCII  /ime/
021635   157   165   164                .ASCII  /out/
021640   051   000                      .ASCII  /)/<00>
021642   045   101   122        P.AKH:  .ASCII  /%AR/
021645   111   116   107                .ASCII  /ING/
021650   040   127   122                .ASCII  / WR/
021653   111   124   105                .ASCII  /ITE/
```

C9


```
021656  040  105  122                    .ASCII  / ER/
021661  122  117  122                    .ASCII  /ROR/
021664  040  050  120                    .ASCII  / (P/
021667  141  162  151                    .ASCII  /ari/
021672  164  171  040                    .ASCII  /ty /
021675  157  162  040                    .ASCII  /or /
021700  164  151  155                    .ASCII  /tim/
021703  145  157  165                    .ASCII  /equ/
021706  164  051  000                    .ASCII  /t)/<00>
021711  000                             .ASCII  <00>
021712  111  116  124          P.AKI:    .ASCII  /INT/
021715  105  122  122                    .ASCII  /ERR/
021720  125  120  124                    .ASCII  /UPT/
021723  040  115  101                    .ASCII  / MA/
021726  123  124  105                    .ASCII  /STE/
021731  122  040  106                    .ASCII  /R F/
021734  101  111  114                    .ASCII  /AIL/
021737  125  122  105                    .ASCII  /URE/
021742  000  000                        .ASCII  <00><00>
021744  045  101  110          P.AKJ:    .ASCII  /%AH/
021747  117  123  124                    .ASCII  /OST/
021752  040  101  103                    .ASCII  / AC/
021755  103  105  123                    .ASCII  /CES/
021760  123  040  124                    .ASCII  /S T/
021763  111  115  105                    .ASCII  /IME/
021766  117  125  124                    .ASCII  /OUT/
021771  040  050  110                    .ASCII  / (H/
021774  151  147  150                    .ASCII  /igh/
021777  145  162  040                    .ASCII  /er /
022002  154  145  166                    .ASCII  /lev/
022005  145  154  040                    .ASCII  /el /
022010  160  162  157                    .ASCII  /pro/
022013  164  157  143                    .ASCII  /toc/
022016  157  154  040                    .ASCII  /ol /
022021  144  145  160                    .ASCII  /dep/
022024  145  156  144                    .ASCII  /end/
022027  145  156  164                    .ASCII  /ent/
022032  051  000                        .ASCII  /)/<00>
022034  045  101  103          P.AKK:    .ASCII  /%AC/
022037  122  105  104                    .ASCII  /RED/
022042  111  124  040                    .ASCII  /IT /
022045  114  111  115                    .ASCII  /LIM/
022050  111  124  040                    .ASCII  /IT /
022053  105  130  103                    .ASCII  /EXC/
022056  105  105  104                    .ASCII  /EED/
022061  105  104  000                    .ASCII  /ED/<00>
022064  045  101  121          P.AKL:    .ASCII  /%AQ/
022067  055  102  125                    .ASCII  /-BU/
022072  123  040  115                    .ASCII  /S M/
022075  101  123  124                    .ASCII  /AST/
022100  105  122  040                    .ASCII  /ER /
022103  105  122  122                    .ASCII  /ERR/
022106  117  122  000                    .ASCII  /OR/<00>
```

```
022111  000                             .ASCII  <00>
022112  045     101     103     P.AKM:  .ASCII  /%AC/
022115  117     116     124             .ASCII  /ONT/
022120  122     117     114             .ASCII  /ROL/
022123  114     105     122             .ASCII  /LER/
022126  040     106     101             .ASCII  / FA/
022131  124     101     114             .ASCII  /TAL/
022134  040     105     122             .ASCII  / ER/
022137  122     117     122             .ASCII  /ROR/
022142  000     000                     .ASCII  <00><00>
022144  045     101     111     P.AKN:  .ASCII  /%AI/
022147  116     123     124             .ASCII  /NST/
022152  122     125     103             .ASCII  /RUC/
022155  124     111     117             .ASCII  /TIO/
022160  116     040     114             .ASCII  /N L/
022163  117     117     120             .ASCII  /OOP/
022166  040     124     111             .ASCII  / TI/
022171  115     105     117             .ASCII  /MEO/
022174  125     124     000             .ASCII  /UT/<00>
022177  000                             .ASCII  <00>
022200  045     101     111     P.AKO:  .ASCII  /%AI/
022203  114     114     105             .ASCII  /LLE/
022206  107     101     114             .ASCII  /GAL/
022211  040     126     111             .ASCII  / VI/
022214  122     124     125             .ASCII  /RTU/
022217  101     114     040             .ASCII  /AL /
022222  103     111     122             .ASCII  /CIR/
022225  103     125     111             .ASCII  /CUI/
022230  124     040     111             .ASCII  /T I/
022233  104     000     000             .ASCII  /D/<00><00>
022236  045     101     111     P.AKP:  .ASCII  /%AI/
022241  116     124     105             .ASCII  /NTE/
022244  122     122     125             .ASCII  /RRU/
022247  120     124     040             .ASCII  /PT /
022252  126     105     103             .ASCII  /VEC/
022255  124     117     122             .ASCII  /TOR/
022260  040     111     114             .ASCII  / IL/
022263  114     105     107             .ASCII  /LEG/
022266  101     114     000             .ASCII  /AL/<00>
022271  000                             .ASCII  <00>
022272  045     101     115     P.AKQ:  .ASCII  /%AM/
022275  101     111     116             .ASCII  /AIN/
022300  124     105     116             .ASCII  /TEN/
022303  101     116     103             .ASCII  /ANC/
022306  105     040     122             .ASCII  /E R/
022311  105     101     104             .ASCII  /EAD/
022314  057     127     122             .ASCII  <57>/WR/
022317  111     124     105             .ASCII  /ITE/
022322  040     111     116             .ASCII  / IN/
022325  126     101     114             .ASCII  /VAL/
022330  111     104     040             .ASCII  /ID /
022333  122     105     107             .ASCII  /REG/
022336  111     117     116             .ASCII  /ION/
```

E9


```
022341   040   111   104                 .ASCII  / ID/
022344   105   116   124                 .ASCII  /ENT/
022347   111   106   111                 .ASCII  /IFI/
022352   105   122   000                 .ASCII  /ER/<00>
022355   000                             .ASCII  <00>
022356   045   101   115        P.AKR:   .ASCII  /%AM/
022361   101   111   116                 .ASCII  /AIN/
022364   124   105   116                 .ASCII  /TEN/
022367   101   116   103                 .ASCII  /ANC/
022372   105   040   127                 .ASCII  /E W/
022375   122   111   124                 .ASCII  /RIT/
022400   105   040   114                 .ASCII  /E L/
022403   117   101   104                 .ASCII  /OAD/
022406   040   124   117                 .ASCII  / TO/
022411   040   116   117                 .ASCII  / NO/
022414   116   055   114                 .ASCII  /N-L/
022417   117   101   104                 .ASCII  /OAD/
022422   101   102   114                 .ASCII  /ABL/
022425   105   040   103                 .ASCII  /E C/
022430   117   116   124                 .ASCII  /ONT/
022433   122   117   114                 .ASCII  /ROL/
022436   114   105   122                 .ASCII  /LER/
022441   000                             .ASCII  <00>
022442   045   101   103        P.AKS:   .ASCII  /%AC/
022445   117   116   124                 .ASCII  /ONT/
022450   122   117   114                 .ASCII  /ROL/
022453   114   105   122                 .ASCII  /LER/
022456   040   122   101                 .ASCII  / RA/
022461   115   040   105                 .ASCII  /M E/
022464   122   122   117                 .ASCII  /RRO/
022467   122   040   050                 .ASCII  /R (/
022472   116   157   156                 .ASCII  /Non/
022475   055   160   141                 .ASCII  /-pa/
022500   162   151   164                 .ASCII  /rit/
022503   171   051   000                 .ASCII  /y)/<00>
022506   045   101   111        P.AKT:   .ASCII  /%AI/
022511   116   111   124                 .ASCII  /NIT/
022514   040   123   105                 .ASCII  / SE/
022517   121   125   105                 .ASCII  /QUE/
022522   116   103   105                 .ASCII  /NCE/
022525   040   105   122                 .ASCII  / ER/
022530   122   117   122                 .ASCII  /ROR/
022533   000                             .ASCII  <00>
022534   045   101   110        P.AKU:   .ASCII  /%AH/
022537   111   107   110                 .ASCII  /IGH/
022542   105   122   040                 .ASCII  /ER /
022545   114   105   126                 .ASCII  /LEV/
022550   105   114   040                 .ASCII  /EL /
022553   120   122   117                 .ASCII  /PRO/
022556   124   117   103                 .ASCII  /TOC/
022561   117   114   040                 .ASCII  /OL /
022564   111   116   103                 .ASCII  /INC/
022567   117   115   120                 .ASCII  /OMP/
```

F9


```
022572  101  124  111                    .ASCII  /ATI/
022575  102  111  114                    .ASCII  /BIL/
022600  111  124  131                    .ASCII  /ITY/
022603  040  105  122                    .ASCII  / ER/
022606  122  117  122                    .ASCII  /ROR/
022611  000                             .ASCII  <00>
022612  045  101  120          P.AKV:    .ASCII  /%AP/
022615  125  122  107                    .ASCII  /URG/
022620  105  057  120                    .ASCII  /E/<57>/P/
022623  117  114  114                    .ASCII  /OLL/
022626  040  110  101                    .ASCII  / HA/
022631  122  104  127                    .ASCII  /RDW/
022634  101  122  105                    .ASCII  /ARE/
022637  040  106  101                    .ASCII  / FA/
022642  111  114  125                    .ASCII  /ILU/
022645  122  105  000                    .ASCII  /RE/<00>
022650  045  101  115          P.AKW:    .ASCII  /%AM/
022653  101  120  120                    .ASCII  /APP/
022656  111  116  107                    .ASCII  /ING/
022661  040  122  105                    .ASCII  / RE/
022664  107  111  123                    .ASCII  /GIS/
022667  124  105  122                    .ASCII  /TER/
022672  040  122  105                    .ASCII  / RE/
022675  101  104  040                    .ASCII  /AD /
022700  106  101  111                    .ASCII  /FAI/
022703  114  125  122                    .ASCII  /LUR/
022706  105  040  050                    .ASCII  /E (/
022711  120  141  162                    .ASCII  /Par/
022714  151  164  171                    .ASCII  /ity/
022717  040  157  162                    .ASCII  / or/
022722  040  164  151                    .ASCII  / ti/
022725  155  145  157                    .ASCII  /meo/
022730  165  164  051                    .ASCII  /ut)/
022733  000                             .ASCII  <00>
022734  021240'                 P.AJZ:  .WORD   P.AKA
022736  021266'                         .WORD   P.AKB
022740  021350'                         .WORD   P.AKC
022742  021432'                         .WORD   P.AKD
022744  021500'                         .WORD   P.AKE
022746  021536'                         .WORD   P.AKF
022750  021574'                         .WORD   P.AKG
022752  021642'                         .WORD   P.AKH
022754  021712'                         .WORD   P.AKI
022756  021744'                         .WORD   P.AKJ
022760  022034'                         .WORD   P.AKK
022762  022064'                         .WORD   P.AKL
022764  022112'                         .WORD   P.AKM
022766  022144'                         .WORD   P.AKN
022770  022200'                         .WORD   P.AKO
022772  022236'                         .WORD   P.AKP
022774  022272'                         .WORD   P.AKQ
022776  022356'                         .WORD   P.AKR
023000  022442'                         .WORD   P.AKS
```

G9


```
023002  022506'                         .WORD   P.AKT
023004  022534'                         .WORD   P.AKU
023006  022612'                         .WORD   P.AKV
023010  022650'                         .WORD   P.AKW
023012     045     101     124  P.AKY:  .ASCII  /%AT/
023015     061     061     040          .ASCII  /11 /
023020     103     120     125          .ASCII  /CPU/
023023     040     106     101          .ASCII  / FA/
023026     111     114     125          .ASCII  /ILU/
023031     122     105     000          .ASCII  /RE/<00>
023034     045     101     116  P.AKZ:  .ASCII  /%AN/
023037     117     116     055          .ASCII  /ON-/
023042     120     101     122          .ASCII  /PAR/
023045     111     124     131          .ASCII  /ITY/
023050     040     122     101          .ASCII  / RA/
023053     115     040     105          .ASCII  /M E/
023056     122     122     117          .ASCII  /RRO/
023061     122     000     000          .ASCII  /R/<00><00>
023064     045     101     123  P.ALA:  .ASCII  /%AS/
023067     124     101     124          .ASCII  /TAT/
023072     105     040     115          .ASCII  /E M/
023075     101     103     110          .ASCII  /ACH/
023100     111     116     105          .ASCII  /INE/
023103     040     106     101          .ASCII  / FA/
023106     111     114     125          .ASCII  /ILU/
023111     122     105     040          .ASCII  /RE /
023114     055     040     124          .ASCII  /- T/
023117     061     061     040          .ASCII  /11 /
023122     101     104     104          .ASCII  /ADD/
023125     122     105     123          .ASCII  /RES/
023130     123     040     122          .ASCII  /S R/
023133     105     107     111          .ASCII  /EGI/
023136     123     124     105          .ASCII  /STE/
023141     122     000     000          .ASCII  /R/<00><00>
023144     045     101     123  P.ALB:  .ASCII  /%AS/
023147     124     101     124          .ASCII  /TAT/
023152     105     040     115          .ASCII  /E M/
023155     101     103     110          .ASCII  /ACH/
023160     111     116     105          .ASCII  /INE/
023163     040     106     101          .ASCII  / FA/
023166     111     114     125          .ASCII  /ILU/
023171     122     105     040          .ASCII  /RE /
023174     055     040     121          .ASCII  /- Q/
023177     055     102     125          .ASCII  /-BU/
023202     123     040     101          .ASCII  /S A/
023205     104     104     122          .ASCII  /DDR/
023210     105     123     123          .ASCII  /ESS/
023213     040     122     105          .ASCII  / RE/
023216     107     111     123          .ASCII  /GIS/
023221     124     105     122          .ASCII  /TER/
023224     000     000                  .ASCII  <00><00>
023226     045     101     123  P.ALC:  .ASCII  /%AS/
023231     124     101     124          .ASCII  /TAT/
```

H9


```
023234  105     040     115             .ASCII  /E M/
023237  101     103     110             .ASCII  /ACH/
023242  111     116     105             .ASCII  /INE/
023245  040     106     101             .ASCII  / FA/
023250  111     114     125             .ASCII  /ILU/
023253  122     105     040             .ASCII  /RE /
023256  055     040     103             .ASCII  /- C/
023261  122     103     040             .ASCII  /RC /
023264  122     105     107             .ASCII  /REG/
023267  111     123     124             .ASCII  /IST/
023272  105     122     000             .ASCII  /ER/<00>
023275  000                             .ASCII  <00>
023276  045     101     123     P.ALD:  .ASCII  /%AS/
023301  124     101     124             .ASCII  /TAT/
023304  105     040     115             .ASCII  /E M/
023307  101     103     110             .ASCII  /ACH/
023312  111     116     105             .ASCII  /INE/
023315  040     106     101             .ASCII  / FA/
023320  111     114     125             .ASCII  /ILU/
023323  122     105     040             .ASCII  /RE /
023326  055     040     123             .ASCII  /- S/
023331  105     122     111             .ASCII  /ERI/
023334  101     114     111             .ASCII  /ALI/
023337  132     105     122             .ASCII  /ZER/
023342  057     104     105             .ASCII  <57>/DE/
023345  123     105     122             .ASCII  /SER/
023350  111     101     114             .ASCII  /IAL/
023353  111     132     105             .ASCII  /IZE/
023356  122     040     122             .ASCII  /R R/
023361  105     107     111             .ASCII  /EGI/
023364  123     124     105             .ASCII  /STE/
023367  122     000     000             .ASCII  /R/<00><00>
023372  045     101     123     P.ALE:  .ASCII  /%AS/
023375  124     101     124             .ASCII  /TAT/
023400  105     040     115             .ASCII  /E M/
023403  101     103     110             .ASCII  /ACH/
023406  111     116     105             .ASCII  /INE/
023411  040     106     101             .ASCII  / FA/
023414  111     114     125             .ASCII  /ILU/
023417  122     105     040             .ASCII  /RE /
023422  055     040     127             .ASCII  /- W/
023425  122     117     116             .ASCII  /RON/
023430  107     040     110             .ASCII  /G H/
023433  101     122     104             .ASCII  /ARD/
023436  127     101     122             .ASCII  /WAR/
023441  105     040     126             .ASCII  /E V/
023444  105     122     123             .ASCII  /ERS/
023447  111     117     116             .ASCII  /ION/
023452  000     000                     .ASCII  <00><00>
023454  023012'                 P.AKX:  .WORD   P.AKY
023456  023034'                         .WORD   P.AKZ
023460  023064'                         .WORD   P.ALA
023462  023144'                         .WORD   P.ALB
```

```
023464  023226'                         .WORD   P.ALC
023466  023276'                         .WORD   P.ALD
023470  023372'                         .WORD   P.ALE
023472     045     116     045  P.ALF:  .ASCII  /%N%/
023475     101     132     122          .ASCII  /AZR/
023500     121     104     040          .ASCII  /QD /
023503     104     105     126          .ASCII  /DEV/
023506     040     106     124          .ASCII  / FT/
023511     114     040     040          .ASCII  /L  /
023514     045     132     065          .ASCII  /%Z5/
023517     045     101     040          .ASCII  /%A /
023522     117     116     040          .ASCII  /ON /
023525     125     116     111          .ASCII  /UNI/
023530     124     040     045          .ASCII  /T %/
023533     132     062     045          .ASCII  /Z2%/
023536     101     040     124          .ASCII  /A T/
023541     123     124     040          .ASCII  /ST /
023544     060     060     061          .ASCII  /001/
023547     040     123     125          .ASCII  / SU/
023552     102     040     060          .ASCII  /B 0/
023555     060     060     040          .ASCII  /00 /
023560     120     103     072          .ASCII  /PC:/
023563     040     045     117          .ASCII  / %O/
023566     066     000                  .ASCII  /6/<00>
023570     045     116     045  P.ALG:  .ASCII  /%N%/
023573     101     132     122          .ASCII  /AZR/
023576     121     104     040          .ASCII  /QD /
023601     110     122     104          .ASCII  /HRD/
023604     040     105     122          .ASCII  / ER/
023607     122     040     040          .ASCII  /R  /
023612     045     132     065          .ASCII  /%Z5/
023615     045     101     040          .ASCII  /%A /
023620     117     116     040          .ASCII  /ON /
023623     125     116     111          .ASCII  /UNI/
023626     124     040     045          .ASCII  /T %/
023631     132     062     045          .ASCII  /Z2%/
023634     101     040     124          .ASCII  /A T/
023637     123     124     040          .ASCII  /ST /
023642     060     060     061          .ASCII  /001/
023645     040     123     125          .ASCII  / SU/
023650     102     040     060          .ASCII  /B 0/
023653     060     060     040          .ASCII  /00 /
023656     120     103     072          .ASCII  /PC:/
023661     040     045     117          .ASCII  / %O/
023664     066     000                  .ASCII  /6/<00>
023666     045     116     045  P.ALH:  .ASCII  /%N%/
023671     101     132     122          .ASCII  /AZR/
023674     121     104     040          .ASCII  /QD /
023677     123     106     124          .ASCII  /SFT/
023702     040     105     122          .ASCII  / ER/
023705     122     040     040          .ASCII  /R  /
023710     045     132     065          .ASCII  /%Z5/
023713     045     101     040          .ASCII  /%A /
```

J9


```
023716  117     116     040                     .ASCII  /ON /
023721  125     116     111                     .ASCII  /UNI/
023724  124     040     045                     .ASCII  /T %/
023727  132     062     045                     .ASCII  /Z2%/
023732  101     040     124                     .ASCII  /A T/
023735  123     124     040                     .ASCII  /ST /
023740  060     060     061                     .ASCII  /001/
023743  040     123     125                     .ASCII  / SU/
023746  102     040     060                     .ASCII  /B 0/
023751  060     060     040                     .ASCII  /00 /
023754  120     103     072                     .ASCII  /PC:/
023757  040     045     117                     .ASCII  / %O/
023762  066     045     116                     .ASCII  /6%N/
023765  000                                     .ASCII  <00>
023766  045     116     045     P.ALI:          .ASCII  /%N%/
023771  101     111     057                     .ASCII  /AI/<57>
023774  117     040     122                     .ASCII  /O R/
023777  105     121     125                     .ASCII  /EQU/
024002  105     123     124                     .ASCII  /EST/
024005  040     106     101                     .ASCII  / FA/
024010  111     114     105                     .ASCII  /ILE/
024013  104     045     116                     .ASCII  /D%N/
024016  000     000                             .ASCII  <00><00>
024020  045     123     064     P.ALJ:          .ASCII  /%S4/
024023  000                                     .ASCII  <00>
024024  045     116     000     P.ALK:          .ASCII  /%N/<00>
024027  000                                     .ASCII  <00>
024030  045     101     040     P.ALL:          .ASCII  /%A /
024033  055     040     000                     .ASCII  /- /<00>
024036  045     101     052     P.ALM:          .ASCII  /%A*/
024041  040     000     000                     .ASCII  / /<00><00>
024044  000000C                 L$HWLEN::
                                                .WORD   <<L$NDHW-L$HWLEN>/2>
024046  172150                  HWPT.IP.ADDR::
                                                .WORD   -5630
024050  000154                  HWPT.VECTOR::
                                                .WORD   154
024052  000004                  HWPT.BR.LEVEL::
                                                .WORD   4
024054  000340                  HWPT.DISK::
                                                .WORD   340
024056  000000                  HWPTS0.LBN::
                                                .WORD   0
024060  000000                  HWPTS1.LBN::
                                                .WORD   0
024062  177777                  HWPTE0.LBN::
                                                .WORD   -1
024064  000000                  HWPTE1.LBN::
                                                .WORD   0
024066  020040                  NAME.LO::
                                                .WORD   20040
024070  020040                  NAME.HI::
                                                .WORD   20040
```

K9


```
024072
024074  000000C                 L$NDHW::.BLKW   1
                                L$SWLEN::
                                        .WORD   <<L$NDSW-L$SWLEN>/2>
024076  000040                  SWP.ERROR::
                                        .WORD   40
024100  000000                  SWP.XFER::
                                        .WORD   0
024102  054046                  SWP.FLAGS::
                                        .WORD   54046
024104  000000                  SWP.DPAT::
                                        .WORD   0
024106  000143                  SWP.RAT::
                                        .WORD   143
024110  000000                  SWP.TIME::
                                        .WORD   0
024112  000020                  SWP.UCNT::
                                        .WORD   20
024114                          SWP.UDPAT::
                                        .BLKW   20
024154  000013                  DUPROUND::
                                        .WORD   13
024156  000000                  MAN.TST::
                                        .WORD   0
024160  000001                  TST.PAR::
                                        .WORD   1
024162                          L$NDSW::.BLKW   1
024164  000000                  L$PROT::.WORD   0
024166  177777                          .WORD   -1
024170  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::
```

L9


```
                        .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::
```

M9


```
                                .BLKW   1
005746                  NEX::   .BLKW   1
005750                  CRN.LOW::
                                .BLKW   1
005752                  CRN.HIGH::
                                .BLKW   1
005754                  TEMP1:: .BLKW   1
005756                  TEMP2:: .BLKW   1
005760                  CREDIT.BAL::
                                .BLKW   1
005762                  NEXT.PKT.USE::
                                .BLKB   1
005763                  HOURS:: .BLKB   1
005764                  MINUTES::
                                .BLKB   1
                                .EVEN
005766                  CLK.TICKS::
                                .BLKW   1
005770                  FERO.LBN::
                                .BLKW   1
005772                  FER1.LBN::
                                .BLKW   1
005774                  CLK.PRESENT::
                                .BLKB   1
005775                  HOE.FLAG::
                                .BLKB   1
005776                  TYPER:: .BLKW   4
006006                  TYPEW:: .BLKW   4
006016                  BAL.IN.PROGRESS::
                                .BLKW   4
006026                  FORCE.WR::
                                .BLKW   4
006036                  CSR.MEM::
                                .BLKW   1
006040  172100          CSR.ADD::
                                .WORD   -5700
006042                  S.PATTERN::
                                .BLKW   1
006044                  S.DUPPKT::
                                .BLKW   1
006046                  P.INDEX::
                                .BLKW   1
006050  000000          RD.COUNT::
                                .WORD   0
006052                  BRLEVEL::
                                .BLKW   1
006054                  D.FAIL::.BLKB   1
006055                  FORCED.ERROR::
                                .BLKB   1
006056                  FER.LBN::
                                .BLKW   1
006060                  FER.BC::.BLKW   1
006062                  INIT.OCCURED::
```

N9


```
006062  000                             .BYTE   0
006063                          ADDR.VECT.OK::
006063  000                             .BYTE   0

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

000001                          ON==            1
000002                          OFF==           2
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
000035                          EF.NEW==        35
000034                          EF.PWR==        34
000040                          EF.START==      40
000037                          EF.RESTART==    37
000036                          EF.CONTINUE==   36
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
```

B10


```
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$ERRTBL==      ERRTYP
024076'                 L$SW==          L$SWLEN+2
024046'                 L$HW==          L$HWLEN+2
000011'                 L$DEPO==        L$REV+1
000136'                 PTCH1==         P.AAA
000210'                 PTCH2==         P.AAB
000262'                 PTCH3==         P.AAC
000334'                 PTCH4==         P.AAD
000406'                 PTCH5==         P.AAE
000460'                 HWQ1==          P.AAF
000474'                 HWQ2==          P.AAG
000504'                 HWQ3==          P.AAH
000546'                 HWQ4==          P.AAI
000564'                 HWQ5==          P.AAJ
000634'                 HWQ6A==         P.AAK
000706'                 HWQ6B==         P.AAL
000762'                 HWQ7A==         P.AAM
001032'                 HWQ7B==         P.AAN
001102'                 HWQ8==          P.AAO
001160'                 HWQ9==          P.AAP
001260'                 HWQ10==         P.AAQ
001310'                 HWQ11==         P.AAR
001342'                 SWQ1==          P.AAS
001364'                 SWQ2==          P.AAT
001444'                 SWQ4==          P.AAU
001466'                 SWQ7==          P.AAV
001540'                 SWQ9==          P.AAW
001614'                 SWQ10==         P.AAX
001660'                 SWQ11==         P.AAY
001712'                 SWQ12==         P.AAZ
002010'                 SWQ13==         P.ABA
002066'                 SWQ14==         P.ABB
002140'                 SWQ15==         P.ABC
002210'                 SWQ17==         P.ABD
002306'                 SWQ19==         P.ABE
002376'                 SWQ20==         P.ABF
002464'                 SWQ21==         P.ABG
002550'                 SWQ22==         P.ABH
002610'                 SWQ23==         P.ABI
002662'                 SWQ24==         P.ABJ
002726'                 SWQ26==         P.ABK
```

C10


```
002770'                 SWM1==          P.ABL
003066'                 SWQ27==         P.ABM
003112'                 SWQ28==         P.ABN
003162'                 NULL==          P.ABO
003164'                 DBM5==          P.ABP
003212'                 DBM12==         P.ABQ
003266'                 DBM15==         P.ABR
003316'                 DBM18==         P.ABS
003370'                 DBM19==         P.ABT
003454'                 DBM20==         P.ABU
003530'                 DBM21==         P.ABV
003612'                 DBM22==         P.ABW
003656'                 DBM23==         P.ABX
003714'                 DBM25==         P.ABY
003762'                 DBM26==         P.ABZ
004014'                 DBM27==         P.ACA
004066'                 DBM28A==        P.ACB
004126'                 DBM28B==        P.ACC
004166'                 DBM29==         P.ACD
004234'                 DBM32==         P.ACE
004310'                 DBM101==        P.ACF
004336'                 DBM104==        P.ACG
004400'                 DBM105==        P.ACH
004436'                 DBM107==        P.ACI
004474'                 DBM108==        P.ACJ
004564'                 DBM109==        P.ACK
004644'                 DBM111==        P.ACL
004744'                 DBM112==        P.ACM
005046'                 DBM120==        P.ACN
005140'                 DBM121==        P.ACO
005230'                 DBM123==        P.ACP
005272'                 DBM125==        P.ACQ
005342'                 DBM126==        P.ACR
005422'                 DBM127==        P.ACS
005500'                 DBM128==        P.ACT
005566'                 DU.MSG==        P.ACU
006306'                 DU.RSN==        P.ACV
006334'                 MSG.01==        P.ADH
006366'                 MSG.02==        P.ADI
006422'                 MSG.03==        P.ADJ
006454'                 RPT1==          P.ADK
006540'                 RPT2==          P.ADL
006604'                 RPT3==          P.ADM
006670'                 RPT4==          P.ADN
006734'                 RPT5==          P.ADO
007022'                 RPT6==          P.ADP
007066'                 RPT7==          P.ADQ
007104'                 RPT8==          P.ADR
007132'                 RPT9==          P.ADS
007164'                 RPT10==         P.ADT
007252'                 RPT11==         P.ADU
007320'                 RPT12==         P.ADV
007366'                 RPT13==         P.ADW
```

D10


```
007466'                 RPT14==         P.ADX
007564'                 RPT15==         P.ADY
007664'                 RPT16==         P.ADZ
007746'                 EGS.01==        P.AEA
007766'                 EGS.02==        P.AEB
010060'                 EGD.10==        P.AEC
010120'                 EGD.11==        P.AED
010144'                 EGD.12==        P.AEE
010172'                 EGD.13==        P.AEF
010220'                 EGD.14==        P.AEG
010250'                 EGD.15==        P.AEH
010266'                 EGD.16==        P.AEI
010316'                 EGD.17==        P.AEJ
010334'                 EGD.18==        P.AEK
010354'                 EGD.19==        P.AEL
010414'                 EGD.20==        P.AEM
010502'                 EGD.21==        P.AEN
010614'                 EGD.22==        P.AEO
010654'                 EGD.23==        P.AEP
010716'                 EGD.24==        P.AEQ
010762'                 EGH.30==        P.AER
011006'                 EBS.01==        P.AES
011050'                 EBD.10==        P.AET
011110'                 EBD.12==        P.AEU
011156'                 EBD.13==        P.AEV
011210'                 EBD.14==        P.AEW
011250'                 EBD.18==        P.AEX
011304'                 EBD.19==        P.AEY
011364'                 EBD.24==        P.AEZ
011440'                 EH.0==          P.AFA
011476'                 EH.1==          P.AFB
011534'                 EH.2==          P.AFC
011574'                 EH.3==          P.AFD
011632'                 EH.4==          P.AFE
011656'                 EH.5==          P.AFF
011706'                 EH.6==          P.AFG
011736'                 EH.7==          P.AFH
011766'                 EH.8==          P.AFI
012022'                 EH.9==          P.AFJ
012052'                 EH.10==         P.AFK
012102'                 EH.12==         P.AFL
012136'                 EH.13==         P.AFM
012210'                 ERR.00==        P.AFN
012744'                 ERR.COD==       P.AFO
013002'                 ELG.00==        P.AGE
013416'                 ELG.FMT==       P.AGF
013432'                 EX.SA==         P.AGM
013450'                 EX.SC==         P.AGN
013500'                 EX.SBO==        P.AGO
013504'                 EX.SB==         P.AGP
013526'                 EX.CMD==        P.AGQ
013546'                 EX.RD==         P.AGR
013556'                 EX.WRT==        P.AGS
```

E10


```
013566'                 EX.CMP==        P.AGT
013602'                 EX.ONL==        P.AGU
013614'                 EX.ACC==        P.AGV
013626'                 EX.OP==         P.AGW
013632'                 EX.BB2==        P.AGX
013700'                 EX.BB12==       P.AGY
013752'                 EX.BBU2==       P.AGZ
014032'                 EX.LBN2==       P.AHA
014072'                 EX.PBN2==       P.AHB
014132'                 EX.LBR2==       P.AHC
014200'                 EX.LBW2==       P.AHD
014250'                 EX.RBN2==       P.AHE
014310'                 EX.CBC==        P.AHF
014354'                 EX.CBR==        P.AHG
014426'                 EX.CBW==        P.AHH
014500'                 EX.BC==         P.AHI
014554'                 EX.BD==         P.AHJ
014632'                 EX.BDR==        P.AHK
014722'                 EX.BDW==        P.AHL
015012'                 EX.RP==         P.AHM
015100'                 EX.WRD==        P.AHN
015110'                 EX.TIM==        P.AHO
015146'                 EX.LB==         P.AHP
015212'                 XX13==          P.AHQ
015234'                 XX23==          P.AHR
015270'                 XX32==          P.AHS
015316'                 XX33==          P.AHT
015354'                 XX34==          P.AHU
015424'                 CER.01==        P.AHV
015470'                 CER.02==        P.AHW
015544'                 SC.SDI==        P.AHX
015570'                 SC.CON==        P.AHY
015612'                 SC.DUP==        P.AHZ
015642'                 SC.ONL==        P.AIA
015664'                 SC.SON==        P.AIB
015704'                 SC.INR==        P.AIC
015736'                 SC.INV==        P.AID
015754'                 SC.UNK==        P.AIE
016034'                 SC.VOL==        P.AIF
016114'                 SC.IOP==        P.AIG
016166'                 SC.DIS==        P.AIH
016260'                 SC.FER==        P.AII
016346'                 SC.FE2==        P.AIJ
016436'                 SC.ISH==        P.AIK
016516'                 SC.IS2==        P.AIL
016576'                 SC.DST==        P.AIM
016652'                 SC.DS2==        P.AIN
016724'                 SC.ECC==        P.AIO
017006'                 SC.ECD==        P.AIP
017040'                 SC.RCT==        P.AIQ
017060'                 SC.FUL==        P.AIR
017134'                 SC.576==        P.AIS
017210'                 SC.FCT==        P.AIT
```

F10


```
017256'                 SC.EC1==        P.AIU
017306'                 SC.EC2==        P.AIV
017336'                 SC.EC3==        P.AIW
017370'                 SC.EC4==        P.AIX
017420'                 SC.EC5==        P.AIY
017450'                 SC.EC6==        P.AIZ
017500'                 SC.EC7==        P.AJA
017532'                 SC.EC8==        P.AJB
017564'                 SC.EC9==        P.AJC
017626'                 SC.SWP==        P.AJD
017666'                 SC.HWP==        P.AJE
017726'                 SC.SAF==        P.AJF
017772'                 SC.ODA==        P.AJG
020022'                 SC.ODB==        P.AJH
020044'                 SC.NXM==        P.AJI
020100'                 SC.PAR==        P.AJJ
020134'                 SC.CTO==        P.AJK
020206'                 SC.SDS==        P.AJL
020264'                 SC.EDC==        P.AJM
020324'                 SC.IDS==        P.AJN
020374'                 SC.SRT==        P.AJO
020466'                 SC.SRI==        P.AJP
020554'                 SC.POE==        P.AJQ
020610'                 SC.RDY==        P.AJR
020672'                 SC.CLK==        P.AJS
020720'                 SC.RSP==        P.AJT
020766'                 SC.SUR==        P.AJU
021016'                 SC.PSP==        P.AJV
021100'                 SC.REP==        P.AJW
021142'                 SC.NBB==        P.AJX
021212'                 SC.REF==        P.AJY
022734'                 CNTR.ERR==      P.AJZ
023454'                 RDRX.ERR==      P.AKX
023472'                 DF.MSG==        P.ALF
023570'                 HRD.MSG==       P.ALG
023666'                 SFT.MSG==       P.ALH
023766'                 HRD.SUB==       P.ALI
024020'                 SPACE4==        P.ALJ
024024'                 CRLF==          P.ALK
024030'                 DASH==          P.ALL
024036'                 ASTERISK==      P.ALM
024046'                 DFPTBL==        L$HWLEN+2
024076'                 SFPTBL==        L$SWLEN+2


;                       PSECT SUMMARY
;
;       Psect Name      Words     Attributes
;        $CODE$          5181       RO  ;  I  ;  LCL,  REL,  CON
;        $FFF$           1562       RW  ;  D  ;  GBL,  REL,  CON
```

```
;                          Library Statistics
;
;                                        -------- Symbols --------      Pages       Processing
;      File                              Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER:[DUNCAN.RELEASE]ZRQDA0.L16;1
;                                         412       184        44        21         00:00.1



;                                  COMMAND QUALIFIERS
;      BLISS/PDP11 ZRQDA0.BL1/LIST=ZRQDA0.LS1/OBJECT=ZRQDA0.OB1/SOURCE=PAGE:53
```

H10


```
;  3573  0     module ZRQDM2 (
;  3574  0
;  3575  0     %title 'RD/RX EXERCISER'
;  3576  0                 ident = 'V02.3',
;  3577  0                 addressing_mode (absolute),
;  3578  0                 environment (noeis)
;  3579  0                 ) =
;  3580  0
;  3581  1     begin
;  3582  1
;  3583  1     %sbttl 'DECLARATIONS'
;  3584  1
;  3585  1     library 'ZRQDA0.L16';                          ! RDRX EXERCISER GLOBAL LIBRARY
;  3586  1
;  3587  1     !MMM require 'BLSMAC.REQ';                     ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;  3588  1     require 'HSAXA0.BLB';                          ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;  5329  1
;  5330  1     forward routine
;  5331  1         NEX_TRAP : L$ISR novalue,
;  5332  1         EMS_01 : novalue,
;  5333  1         EMS_TIM : novalue,
;  5334  1         EMS_DBN : NOVALUE,                                                  !ZZZ
;  5335  1         EMS_BLK : NOVALUE,                                                  !ZZZ
;  5336  1         SET_CPAR : novalue,
;  5337  1         SET_UPAR : novalue;
;  5338  1
;  5339  1     external
;  5340  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;  5341  1                         ! RUN-TIME CONTROLLER STATUS TABLES
;  5342  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;  5343  1                         ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;  5344  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;  5345  1                         ! DRIVER CONTROLLER TABLES
;  5346  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;  5347  1                         ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;  5348  1         RDRX_ADDR : ref rdrx field (RC_REG),
;  5349  1                         ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;  5350  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;  5351  1                         ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;  5352  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                             !ZZZ
;  5353  1                         !CONTAINS LBNS (HI + LO FIELDS) FOR SEQUENTIAL    !ZZZ
;  5354  1                         !I/O TRANSFER FOR EACH UNIT.                      !ZZZ
;  5355  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;  5356  1                         ! STATISTICS TABLES
;  5357  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;  5358  1                         ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;  5359  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;  5360  1                         !INFO FROM RECEIVE AND SEND CMDS                  ZZZ
;  5361  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TRACK DIRECTION ZZZ
;  5362  1         RDM_CNT : WORD,                             !NO OF RANDOM NOS   \\KEEP    ZZZ
;  5363  1         RANDOM : VECTOR [RDM_LEN, WORD],            !RANDOM NO TABLE    //TOGETHER  ZZZ
;  5364  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;  5365  1                         ! STATISTICS TABLE FOR CONTROLLER ERRORS
```

```
;  5366  1        MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;  5367  1                           ! MSCP PACKET POOL
;  5368  1        IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  5369  1                           ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;  5370  1        PKT_USE : vector [PKT_CNT, byte, signed],
;  5371  1                           ! MSCP PACKET POOL ALLOCATION TABLE
;  5372  1        RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  5373  1                           ! RETURN PACKET POOL
;  5374  1        RP_USE : vector [RP_CNT, byte, signed],
;  5375  1                           ! RETURN PACKET POOL ALLOCATION TABLE
;  5376  1        RP_INDX : word,    ! CURRENT RETURN PACKET INDEX
;  5377  1        RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;  5378  1                           ! CURRENT RETURN PACKET ADDRESS
;  5379  1        ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;  5380  1                           ! ERROR-LOG PACKET SAVE AREA
;  5381  1        BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;  5382  1        BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;  5383  1        IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;  5384  1        IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;  5385  1        IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;  5386  1        ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;  5387  1        EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;  5388  1        DUP_FLAGS : WORD,                                    !DUP FLAGS       ZZZ
;  5389  1        CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;  5390  1        CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  5391  1        CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;  5392  1        CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;  5393  1        DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
;  5394  1        QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;  5395  1        FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;  5396  1        BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;  5397  1        ST_CODE : word,                                      ! CURRENT STATUS CODE
;  5398  1        SB_CODE : word,                                      ! CURRENT SUB-CODE
;  5399  1        STEP : word,                                         ! CURRENT STEP IN HARD INIT
;  5400  1        OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;  5401  1        SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;  5402  1        CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;  5403  1        NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;  5404  1        CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;  5405  1        CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;  5406  1        TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;  5407  1        TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
;  5408  1        CREDIT_BAL : word,                                   ! CREDIT BALANCE
;  5409  1        NEXT_PRT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;  5410  1        HOURS: byte,                                         ! TIME OF DAY (HOURS)
;  5411  1        MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;  5412  1        CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
;  5413  1        FER0_LBN : word,                                     !LO LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;  5414  1        FER1_LBN : word,                                     !HI LBN ADR OF THE "FORCED ERROR" BLOCK          ZZZ
;  5415  1        CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  5416  1        HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;  5417  1        FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
;  5418  1        FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
```

```
;    5419  1          FER_BC : word,                          ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    5420  1          INIT_OCCURED : byte,                    ! EXERCISER INITIALIZATION COMPLETE
;    5421  1          ADDR_VECT_OK : byte,                    ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    5422  1          DBM5,
;    5423  1          DBM125,                                 !MMM
;    5424  1          DBM126,                                 !MMM
;    5425  1
;    5426  1          TYPER : VECTOR [MAX_UNITS, WORD],   !READ I/O COUNTER                    ZZZ MMM
;    5427  1          TYPEW : VECTOR [MAX_UNITS, WORD],   !WRITE I/O COUNTER                   ZZZ MMM
;    5428  1          BAL_IN_PROGRESS : VECTOR [MAX_UNITS, WORD],     !FLAG SET TO BALANCE I/O TYPES     ZZZ MMM
;    5429  1          FORCE_WR : VECTOR [MAX_UNITS, WORD],            ! MMM
;    5430  1          CSR_MEM : WORD,                                 ! MMM
;    5431  1          CSR_ADD : WORD,                                 ! MMM
;    5432  1          P_INDEX : SIGNED WORD,              !CURRENT MESSAGE PACKET INDEX      ZZZ
;    5433  1          S_PATTERN : WORD,                   !PATTERN FOR DUP WRITES            ZZZ
;    5434  1          S_DUPPKT : WORD,                    !DBN BYTE COUNTER                  ZZZ
;    5435  1          RD_COUNT : WORD,                    ! NUMBER OF WINCHESTER UNITS       ZZZ
;    5436  1          BRLEVEL : WORD,                     !BUS REQUEST PRIORITY LEVEL        ZZZ
;    5437  1          D_FAIL : BYTE,                      !SIGNIFIES DUP TYPE ERROR          ZZZ
;    5438  1          DBM107,
;    5439  1          DU_MSG,
;    5440  1          DU_RSN : vector [11],
;    5441  1          RPT1,
;    5442  1          RPT2,
;    5443  1          RPT3,
;    5444  1          RPT4,
;    5445  1          RPT5,
;    5446  1          RPT6,
;    5447  1          RPT7,
;    5448  1          RPT8,
;    5449  1          RPT9,
;    5450  1          RPT10,
;    5451  1          RPT11,
;    5452  1          RPT12,
;    5453  1          RPT13,
;    5454  1          RPT14,
;    5455  1          RPT15,
;    5456  1          RPT16,
;    5457  1      !ZZZ    RPT17,
;    5458  1      !ZZZ    RPT18,
;    5459  1      !ZZZ    RPT19,
;    5460  1
;    5461  1          MSG_01,
;    5462  1          EGS_01,
;    5463  1          EBS_01,
;    5464  1          EBD_10,
;    5465  1          EBD_12,
;    5466  1          EBD_13,
;    5467  1          EBD_14,
;    5468  1          EBD_18,
;    5469  1          EBD_19,
;    5470  1          EBD_24,
;    5471  1          ERR_00,
```

K10


```
;    5472  1          ERR_COD : vector [14],
;    5473  1          ELG_OO,
;    5474  1          ELG_FMT : vector [5],
;    5475  1          EX_TIM,
;    5476  1          !EX_DUP,                                                    !MMM
;    5477  1          EX_CB,                                                      !MMM
;    5478  1          XX13,                                                       !ZZZ
;    5479  1          XX23,                                                       !ZZZ
;    5480  1          XX32,                                                       !ZZZ
;    5481  1          XX33,                                                       !ZZZ
;    5482  1          XX34,                                                       !ZZZ
;    5483  1          EX_SA,
;    5484  1          EX_SC,
;    5485  1          EX_SBO,
;    5486  1          EX_SB,
;    5487  1          EX_RP,
;    5488  1          EX_WRD,
;    5489  1          EX_CMD,
;    5490  1          EX_RD,
;    5491  1          EX_WRT,
;    5492  1          EX_CMP,
;    5493  1          EX_ONL,
;    5494  1          EX_ACC,
;    5495  1          EX_OP,
;    5496  1          EX_BB2,
;    5497  1          EX_BB12,
;    5498  1          EX_BBU2,
;    5499  1          EX_LBN2,
;    5500  1          EX_PBN2,
;    5501  1          EX_LBR2,
;    5502  1          EX_LBW2,
;    5503  1          EX_RBN2,
;    5504  1          EX_CBC,
;    5505  1          EX_CBR,
;    5506  1          EX_CBW,
;    5507  1          EX_BC,
;    5508  1          EX_BD,
;    5509  1          EX_BDR,
;    5510  1          EX_BDW,
;    5511  1          SC_SDI,
;    5512  1          SC_CON,
;    5513  1          SC_DUP,
;    5514  1          SC_ONL,
;    5515  1          SC_SON,
;    5516  1          SC_INR,                                  !MMM
;    5517  1          SC_INV,                                  !MMM
;    5518  1          SC_UNK,
;    5519  1          SC_VOL,
;    5520  1          SC_IOP,
;    5521  1          SC_DIS,
;    5522  1          SC_FER,
;    5523  1          SC_FE2,
;    5524  1          SC_ISH,
```

```
;    5525  1          SC_IS2,
;    5526  1          SC_DST,
;    5527  1          SC_DS2,
;    5528  1          SC_ECC,
;    5529  1          SC_ECD,
;    5530  1          SC_RCT,
;    5531  1          SC_FUL,
;    5532  1          SC_576,
;    5533  1          SC_FCT,
;    5534  1          SC_SWP,
;    5535  1          SC_HWP,
;    5536  1          SC_SAF,                    !MMM
;    5537  1          SC_EC1,
;    5538  1          SC_EC2,
;    5539  1          SC_EC3,
;    5540  1          SC_EC4,
;    5541  1          SC_EC5,
;    5542  1          SC_EC6,
;    5543  1          SC_EC7,
;    5544  1          SC_EC8,
;    5545  1          SC_EC9,
;    5546  1          SC_ODA,
;    5547  1          SC_ODB,
;    5548  1          SC_NXM,
;    5549  1          SC_PAR,
;    5550  1          SC_CTO,
;    5551  1          SC_SDS,
;    5552  1          SC_EDC,
;    5553  1          SC_IDS,
;    5554  1          SC_SRT,
;    5555  1          SC_SRI,
;    5556  1          SC_POE,
;    5557  1          SC_RDY,
;    5558  1          SC_CLK,
;    5559  1          SC_RSP,
;    5560  1          SC_SUR,
;    5561  1          SC_PSP,
;    5562  1          SC_REP,                        !MMM
;    5563  1          SC_NBB,                        !MMM
;    5564  1          SC_REF,                        !MMM
;    5565  1          CER_01,
;    5566  1          CER_02,
;    5567  1          CNTR_ERR : vector [23],
;    5568  1          RDRX_ERR : vector [7],
;    5569  1          SPACE4,
;    5570  1          CRLF,
;    5571  1          DASH,
;    5572  1          ASTERISK,
;    5573  1          HWQ1,
;    5574  1          HWQ2,
;    5575  1          HWQ3,
;    5576  1          HWQ4,
;    5577  1          HWQ5,
```

```
;    5578  1          HWQ6A,
;    5579  1          HWQ6B,
;    5580  1          HWQ7A,
;    5581  1          HWQ7B,
;    5582  1          HWQ8,
;    5583  1          HWQ9,
;    5584  1          HWQ10,
;    5585  1          HWQ11,                                                  !ZZZ
;    5586  1          SWQ1,                                                   !ZZZ
;    5587  1          SWQ2,
;    5588  1          SWQ4,
;    5589  1          SWQ7,
;    5590  1          SWQ9,
;    5591  1          SWQ10,
;    5592  1          SWQ11,
;    5593  1          SWQ12,
;    5594  1          SWQ13,
;    5595  1          SWQ14,
;    5596  1          SWQ15,
;    5597  1          SWQ17,
;    5598  1          SWQ19,
;    5599  1          SWQ20,
;    5600  1          SWQ21,
;    5601  1          SWQ22,
;    5602  1          SWQ23,
;    5603  1          SWQ24,
;    5604  1          !MMMSWQ25,
;    5605  1          SWQ26,                                                                  !ZZZ
;    5606  1          SWQ27,                                                                  !MMM
;    5607  1          SWQ28,                                                                  !MMM
;    5608  1          EH_0,                                                                   !ZZZ
;    5609  1          EH_1,                                                                   !ZZZ
;    5610  1          EH_2,                                                                   !ZZZ
;    5611  1          EH_3,                                                                   !ZZZ
;    5612  1          EH_4,                                                                   !ZZZ
;    5613  1          EH_5,                                                                   !ZZZ
;    5614  1          EH_6,                                                                   !ZZZ
;    5615  1          EH_7,                                                                   !ZZZ
;    5616  1          EH_8,                                                                   !ZZZ
;    5617  1          EH_9,                                                                   !ZZZ
;    5618  1          EH_10,                                                                  !ZZZ
;    5619  1          EH_12,                                                                  !ZZZ
;    5620  1          EH_13,                                                                  !ZZZ
;    5621  1          SWM1,                                                                   !ZZZ
;    5622  1          NULL,
;    5623  1          SWP_FLAGS : word,
;    5624  1          L$HIMEM,
;    5625  1          L$LUN,
;    5626  1          L$UNIT;
;    5627  1      !   O_BRK;
;    5628  1
;    5629  1      own
;    5630  1      !MMM    TBL_SUC : vector [17] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
```

N10


```
;    5631  1        !MMM                NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON),
;    5632  1            TBL_SUC : vector [I9] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
;    5633  1                               NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SON, SC_INR, SC_INV),      !MMM
;    5634  1            TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
;    5635  1                               NULL, SC_DIS),
;    5636  1            TBL_MFE : vector [I1] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC_576,
;    5637  1                               SC_FCT, SC_ECC, SC_RCT, SC_FUL, SC_ECI),
;    5638  1        !MMM    TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
;    5639  1            TBL_WPT : vector [4] initial (NULL, SC_SWP, SC_HWP, SC_SAF),                                              !MMM
;    5640  1            TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
;    5641  1                               SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
;    5642  1            TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
;    5643  1            TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_EDC, SC_IDS),
;    5644  1            TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC_POE, SC_RDY, SC_CLK, SC_RSP,
;    5645  1                               SC_SUR, SC_PSP),
;    5646  1            TBL_BRC : vector [5] initial (SC_REP, SC_NBB, SC_REF, SC_REF, SC_REF);                                    !MMM
;    5647  1
;    5648  1
```

```
;   5649  1     %sbttl 'TYPE AND DESCRIPTION'
;   5650  1
;   5651  1     EQUALS;
;   5652  1
;   5653  1     DEVTYP (%asciz'RQDX or RUX50');                ! NAME OF DEVICE SUPPORTED BY PROGRAM
;   5654  1     DESCRIPT (%asciz'RD/RX EXERCISER');     ! TEST DESCRIPTION
```

```
;    5655  1     %sbttl 'HARDWARE PARAMETER CODING SECTION'
;    5656  1
;    5657  1     !+
;    5658  1     ! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5659  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5660  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5661  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5662  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5663  1     ! WITH THE OPERATOR.
;    5664  1     !-
;    5665  1
;    5666  1     BGNHRD;
;    5667  1
;    5668  1     GPRMA (HWQ1, 0, 0, %o'160000', %o'177777', YES, 1);                    ! IP ADDRESS
;    5669  1     GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                            ! VECTOR
;    5670  1     GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                     ! BR LEVEL
;    5671  1     GPRMD (HWQ4, 6, 0, %o'17', %decimal'0', %decimal'15', YES, 1);         ! RDRX DRIVE NUMBER
;    5672  1     GPRML (HWQ10, 6, %o'000040', YES, 1);                                  !ALSO RUN DUP EXERCISER     ZZZ
;    5673  1     XFERF (NODU);                                                          !                           ZZZ
;    5674  1     GPRML (HWQ11, 6, %o'000100', YES, 1);                                  !WRITE DIAG AREA            ZZZ
;    5675  1     $L (NODU);                                                             !                           ZZZ
;    5676  1     GPRML (HWQ5, 6, %o'000200', YES, 1);                                   !TEST ENTIRE CUSTOMER AREA? ZZZ
;    5677  1     XFERT (TOQ8);                                                          !BR IF YES                  ZZZ
;    5678  1     GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);      ! STARTING LBN LO           ZZZ
;    5679  1     GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! STARTING LBN HI           ZZZ
;    5680  1     GPRMD (HWQ7A, 12, 0, %o'177777', GP$ATLO (8), %o'177777', YES, 1);     ! ENDING LBN LO             ZZZ
;    5681  1     GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! ENDING LBN HI             ZZZ
;    5682  1     $L (TOQ8);                                                             !                           ZZZ
;    5683  1     GPRML (HWQ8, 6, %o'100000', NO, 0);                                    !WRITE ON CUST DATA AREA
;    5684  1     XFERF (HWDONE);                                                        !NO - DONE
;    5685  1     GPRML (HWQ9, 6, %o'100000', NO, 1);                                    !** WARNING / CONFIRM
;    5686  1     $L (HWDONE);
;    5687  1
;    5688  1     ENDHRD;
```

```
;    5689  1     %sbttl 'SOFTWARE PARAMETER CODING SECTION'
;    5690  1
;    5691  1     !+
;    5692  1     ! THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5693  1     ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5694  1     ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5695  1     ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5696  1     ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5697  1     ! WITH THE OPERATOR.
;    5698  1     !-
;    5699  1
;    5700  1     BGNSFT;
;    5701  1
;    5702  1     !GPRML (SWQ16, 4, SWF_TRC, YES, 1);                          ! ENABLE DIAGNOSTIC TRACE
;    5703  1     GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);           ! START TIME
;    5704  1     GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);            ! ERROR LIMIT
;    5705  1     GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);               ! TRANSFER LIMIT
;    5706  1     GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);             ! PERCENT OF RD OPERATIONS
;    5707  1     GPRML (SWQ15, 4, SWF_CST, YES, 1);                           ! CLEAR STATISTICAL TABLES ?
;    5708  1     GPRML (SWQ20, 4, SWF_FER, YES, 1);                           ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
;    5709  1     GPRML (SWQ23, 4, SWF_BLK, YES, 1);                           ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
;    5710  1     GPRML (SWQ21, 4, SWF_HRD, YES, 1);                           ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
;    5711  1     GPRML (SWQ22, 4, SWF_SFT, YES, 1);                           ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
;    5712  1     !MMMGPRML (SWQ25, 4, SWF_TRY, YES, 1);                       ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
;    5713  1     GPRML (SWQ4, 4, SWF_RDM, YES, 1);                            ! RANDOM SEEK MODE ?
;    5714  1     XFERF (SW1);                                                 ! IF NO, DO NEXT QUESTION
;    5715  1     XFER (SW2);                                                  !
;    5716  1     $L (SW1);                                                    !
;    5717  1     GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                           ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
;    5718  1     $L (SW2);                                                    !
;    5719  1     GPRML (SWQ7, 4, SWF_CRC, YES, 1);                            ! READ-COMPARES AT CONTROLLER ?
;    5720  1     GPRML (SWQ26, 4, SWF_APT, YES, 1);                           !RUNNING UNDER A.P.T. MONITOR?          ZZZ
;    5721  1     DISPLAY (SWM1);                                              ! REMAINING QUESTIONS ONLY APPLY ...
;    5722  1     GPRML (SWQ9, 4, SWF_CWC, YES, 1);                            ! WRITE-COMPARES AT CONTROLLER ?
;    5723  1     XFERF (SW3);                                                 ! IF NO, DO NEXT QUESTION
;    5724  1     XFER (SW4);                                                  !
;    5725  1     $L (SW3);                                                    !
;    5726  1     GPRML (SWQ10, 4, SWF_HWC, YES, 1);                           ! CHECK WRITES AT HOST BY READING ?
;    5727  1     $L (SW4);                                                    !
;    5728  1     GPRML (SWQ11, 4, SWF_UDP, YES, 1);                           ! USER-DEFINED DATA PATTERN ?
;    5729  1     XFERF (SW5);                                                 ! IF NO, DO NEXT QUESTION
;    5730  1     XFER (SW6);                                                  !
;    5731  1     $L (SW5);                                                    !
;    5732  1     GPRMD (SWQ12, 6, D, %o'177777', 0, DP_CNT, YES, 1);          ! SELECT PRE-DEFINED DATA PATTERN
;    5733  1     XFER (SW7);                                                  ! DONE
;    5734  1     $L (SW6);                                                    !
;    5735  1     GPRMD (SWQ13, 12, D, %o'177777', 1, MAX_UDP_CNT, YES, 1);    ! NO. OF WORDS IN USER DATA PATTERN
;    5736  1     !MMMGPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', NO, 12); ! PATTERN VALUES
;    5737  1     GPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', YES, 12);    ! PATTERN VALUES MMM
;    5738  1     $L (SW7);                                                    !
;    5739  1     GPRML (SWQ27, 48, %o'000001', YES, 1);                       !       ! Manufacturing Test (Reduce Duty Cycle) MMM
;    5740  1     GPRML (SWQ28, 50, %o'000001', YES, 1);                       !        ! Enable Host Memory Parity MMM
;    5741  1     ENDSFT;
```

```
;  5742  1
;  5743  1
;  5744  1     %sbttl 'REPORT CODING SECTION'
;  5745  1
;  5746  1
;  5747  1     !+
;  5748  1     ! THE REPORT CODING SECTION CONTAINS THE
;  5749  1     ! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;  5750  1     !-
;  5751  1
;  5752  1
;  5753  2     BGNRPT;
;  5754  2
;  5755  2     local
;  5756  2         CUR_PRIORITY : word;
;  5757  2
;  5758  2     GETPRI (CUR_PRIORITY);
;  5759  2     !ZZZ SETPRI (PRI04);                                    !ZZZ
;  5760  2     SETPRI (.BRLEVEL);                          !ZZZ
;  5761  2
;  5762  2
;  5763  2     PRINTS (RPT1);
;  5764  2     PRINTS (RPT2);
;  5765  2     PRINTS (RPT3);
;  5766  2     PRINTS (RPT4);
;  5767  2     PRINTS (RPT5);
;  5768  2     PRINTS (RPT6);
;  5769  2
;  5770  2     incr CTLR from 0 to MAX_CTLR - 1 do
;  5771  2
;  5772  3         begin
;  5773  3         SET_CPAR (.CTLR);
;  5774  3
;  5775  3         incr DISK from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  5776  3
;  5777  4             begin
;  5778  4             SET_UPAR (.DISK);
;  5779  4
;  5780  4
;  5781  4             if .CST_ADDR [.DISK + OF_DATA, D_PRES] eql PRESENT
;  5782  4             then
;  5783  4
;  5784  5                 begin
;P 5785  5                 PRINTS (RPT7,
;  5786  5                     .L$LUN, .CST_ADDR [.DISK + OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK + OF_NAME_0, D_NAME_0]);
;P 5787  5                 PRINTS (RPT8,
;P 5788  5                     .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
;  5789  5                     .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
;P 5790  5                 PRINTS (RPT8,
;P 5791  5                     .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
;  5792  5                     .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT_LO]);
;P 5793  5                 PRINTS (RPT9,
;P 5794  5                     .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
```

```
;   5795  5                       .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
;   5796  4                   end;
;   5797  3               end;
;   5798  3
;   5799  3
;   5800  3           if .CST [.CTLR, STATE] eql PRESENT
;   5801  3           then
;   5802  3
;   5803  4               begin
;   5804  4               PRINTS (RPT10);
;   5805  4               PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
;   5806  3               end;
;   5807  3
;   5808  3
;   5809  2           end;
;   5810  2
;   5811  2       SETPRI (.CUR_PRIORITY);
;   5812  2
;   5813  2       IF .RD_COUNT NEQ 0                                          !IF THERE IS A WINCHESTER     ZZZ
;   5814  2       THEN                                                        !THEN OUTPUT EXTRA LINES      ZZZ
;   5815  2
;   5816  3       begin                                                       ! PRINTS DUP DATA
;   5817  3       prints(crlf);
;   5818  3       PRINTS(RPT13);
;   5819  3       PRINTS(RPT14);
;   5820  3       PRINTS(RPT15);
;   5821  3       INCR CTLR FROM 0 TO MAX_CTLR-1 DO
;   5822  4               BEGIN
;   5823  4               SET_CPAR(.CTLR);
;   5824  4               INCR DISK FROM (0+OF_UN) TO (3*UNIT_SIZE+OF_UN) BY UNIT_SIZE DO
;   5825  5                       BEGIN
;   5826  5                       SET_UPAR(.DISK);
;   5827  5                       IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_51 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;   5828  5                       THEN
;P  5829  5                   PRINTS (RPT16,
;P  5830  5                       .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;   5831  5                       .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;   5832  5
;   5833  5       !ZZZ            IF .CST_ADDR[.DISK, D_TYPE] EQLU RD_52 and .CST_ADDR [.DISK, D_PRES] eql PRESENT
;   5834  5       !ZZZ            THEN
;   5835  5       !ZZZ        PRINTS (RPT18,
;   5836  5       !ZZZ            .L$LUN, .CST_ADDR [.DISK, D_DISK_NUM],
;   5837  5       !ZZZ            .T_ADDR [T_DBN_RD], .T_ADDR [T_BLK_RD], .T_ADDR [T_DBN_WT], .T_ADDR [T_BLK_WT]);
;   5838  4                       END;
;   5839  3               END;
;   5840  3                  PRINTS (CRLF);
;   5841  2       END;
;   5842  2
;   5843  2       PRINTS (CRLF);                                              !MMM
;   5844  2       EMS_TIM ();                                                 !MMM
;   5845  2
;   5846  1       ENDRPT;
```

```
                                        .TITLE  ZRQDM2 RD/RX EXERCISER
                                        .IDENT  /V02.3/
                                        .ENABL  AMA


000000                                  .PSECT  $CODE$,  RO
000000      122     121     104     L$DVTYP::
                                        .ASCII  /RQD/
000003      130     040     157             .ASCII  /X o/
000006      162     040     122             .ASCII  /r R/
000011      125     130     065             .ASCII  /UX5/
000014      060     000                     .ASCII  /0/<00>
000016                                      .BLKB   2
000020      122     104     057     L$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>
```

H11


```
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
```

I11


```
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   SWQ4
000352  000002                          .WORD   2
000354  000000C                 $SW1:   .WORD   <<<<$LSW1-$SW1>*400>+4>+40>
000356  000000C                 $SW2:   .WORD   <<<$LSW2-$SW2>*400>+4>
000360  001004                  $LSW1:  .WORD   1004
000362  002130                  GP$24:: .WORD   2130
000364  000000G                         .WORD   SWQ19
000366  001000                          .WORD   1000
000370  001004                  $LSW2:  .WORD   1004
000372  002130                  GP$25:: .WORD   2130
000374  000000G                         .WORD   SWQ7
000376  000004                          .WORD   4
000400  002130                  GP$26:: .WORD   2130
000402  000000G                         .WORD   SWQ26
000404  000001                          .WORD   1
000406  000003                  GP$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
```

J11


```
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  007232                  GP$32:: .WORD   7232
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  030130                  GP$33:: .WORD   30130
000522  000000G                         .WORD   SWQ27
000524  000001                          .WORD   1
000526  031130                  GP$34:: .WORD   31130
000530  000000G                         .WORD   SWQ28
000532  000001                          .WORD   1
000534                          L$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                         .WORD   SC.INR
000044  000000G                         .WORD   SC.INV
000046  000000G                 TBL.OFL:.WORD   SC.UNK
000050  000000G                         .WORD   SC.VOL
000052  000000G                         .WORD   SC.IOP
000054  000000G                         .WORD   NULL
```

K11


```
000056  000000G                         .WORD   SC.DUP
000060  000000G                         .WORD   NULL
000062  000000G                         .WORD   NULL
000064  000000G                         .WORD   NULL
000066  000000G                         .WORD   SC.DIS
000070  000000G                 TBL.MFE:.WORD   SC.FER
000072  000000G                         .WORD   NULL
000074  000000G                         .WORD   SC.ISH
000076  000000G                         .WORD   SC.DST
000100  000000G                         .WORD   SC.EC9
000102  000000G                         .WORD   SC.576
000104  000000G                         .WORD   SC.FCT
000106  000000G                         .WORD   SC.ECC
000110  000000G                         .WORD   SC.RCT
000112  000000G                         .WORD   SC.FUL
000114  000000G                         .WORD   SC.EC1
000116  000000G                 TBL.WPT:.WORD   NULL
000120  000000G                         .WORD   SC.SWP
000122  000000G                         .WORD   SC.HWP
000124  000000G                         .WORD   SC.SAF
000126  000000G                 TBL.DAT:.WORD   SC.FE2
000130  000000G                         .WORD   NULL
000132  000000G                         .WORD   SC.IS2
000134  000000G                         .WORD   SC.DS2
000136  000000G                         .WORD   SC.EC9
000140  000000G                         .WORD   NULL
000142  000000G                         .WORD   NULL
000144  000000G                         .WORD   SC.ECD
000146  000000G                         .WORD   SC.EC1
000150  000000G                         .WORD   SC.EC2
000152  000000G                         .WORD   SC.EC3
000154  000000G                         .WORD   SC.EC4
000156  000000G                         .WORD   SC.EC5
000160  000000G                         .WORD   SC.EC6
000162  000000G                         .WORD   SC.EC7
000164  000000G                         .WORD   SC.EC8
000166  000000G                 TBL.HST:.WORD   NULL
000170  000000G                         .WORD   SC.ODA
000172  000000G                         .WORD   SC.ODB
000174  000000G                         .WORD   SC.NXM
000176  000000G                         .WORD   SC.PAR
000200  000000G                 TBL.CNT:.WORD   SC.CTO
000202  000000G                         .WORD   SC.SDS
000204  000000G                         .WORD   SC.EDC
000206  000000G                         .WORD   SC.IDS
000210  000000G                 TBL.DRV:.WORD   NULL
000212  000000G                         .WORD   SC.SRT
000214  000000G                         .WORD   SC.SRI
000216  000000G                         .WORD   SC.POE
000220  000000G                         .WORD   SC.RDY
000222  000000G                         .WORD   SC.CLK
000224  000000G                         .WORD   SC.RSP
000226  000000G                         .WORD   SC.SUR
```

```
000230  000000G                         .WORD   SC.PSP
000232  000000G               TBL.BRC:  .WORD   SC.REP
000234  000000G                         .WORD   SC.NBB
000236  000000G                         .WORD   SC.REF
000240  000000G                         .WORD   SC.REF
000242  000000G                         .WORD   SC.REF

                        .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, TEMP1
                        .GLOBL  TEMP2, CREDIT.BAL, NEXT.PKT.USE
                        .GLOBL  HOURS, MINUTES, CLK.TICKS, FERO.LBN
                        .GLOBL  FER1.LBN, CLK.PRESENT, HOE.FLAG
                        .GLOBL  FORCED.ERROR, FER.LBN, FER.BC
                        .GLOBL  INIT.OCCURED, ADDR.VECT.OK, DBM5
                        .GLOBL  DBM125, DBM126, TYPER, TYPEW, BAL.IN.PROGRESS
                        .GLOBL  FORCE.WR, CSR.MEM, CSR.ADD, P.INDEX
                        .GLOBL  S.PATTERN, S.DUPPKT, RD.COUNT
                        .GLOBL  BRLEVEL, D.FAIL, DBM107, DU.MSG
                        .GLOBL  DU.RSN, RPT1, RPT2, RPT3, RPT4
                        .GLOBL  RPT5, RPT6, RPT7, RPT8, RPT9, RPT10
                        .GLOBL  RPT11, RPT12, RPT13, RPT14, RPT15
                        .GLOBL  RPT16, MSG.01, EGS.01, EBS.01
                        .GLOBL  EBD.10, EBD.12, EBD.13, EBD.14
                        .GLOBL  EBD.18, EBD.19, EBD.24, ERR.00
                        .GLOBL  ERR.COD, ELG.00, ELG.FMT, EX.TIM
                        .GLOBL  EX.LB, XX13, XX23, XX32, XX33
                        .GLOBL  XX34, EX.SA, EX.SC, EX.SBO, EX.SB
                        .GLOBL  EX.RP, EX.WRD, EX.CMD, EX.RD, EX.WRT
                        .GLOBL  EX.CMP, EX.ONL, EX.ACC, EX.OP
                        .GLOBL  EX.BB2, EX.BB12, EX.BBU2, EX.LBN2
                        .GLOBL  EX.PBN2, EX.LBR2, EX.LBW2, EX.RBN2
                        .GLOBL  EX.CBC, EX.CBR, EX.CBW, EX.BC
                        .GLOBL  EX.BD, EX.BDR, EX.BDW, SC.SDI
                        .GLOBL  SC.CON, SC.DUP, SC.ONL, SC.SON
                        .GLOBL  SC.INR, SC.INV, SC.UNK, SC.VOL
                        .GLOBL  SC.IOP, SC.DIS, SC.FER, SC.FE2
                        .GLOBL  SC.ISH, SC.IS2, SC.DST, SC.DS2
                        .GLOBL  SC.ECC, SC.ECD, SC.RCT, SC.FUL
                        .GLOBL  SC.576, SC.FCT, SC.SWP, SC.HWP
                        .GLOBL  SC.SAF, SC.EC1, SC.EC2, SC.EC3
                        .GLOBL  SC.EC4, SC.EC5, SC.EC6, SC.EC7
                        .GLOBL  SC.EC8, SC.EC9, SC.ODA, SC.ODB
```

```
                        .GLOBL  SC.NXM, SC.PAR, SC.CTO, SC.SDS
                        .GLOBL  SC.EDC, SC.IDS, SC.SRT, SC.SRI
                        .GLOBL  SC.POE, SC.RDY, SC.CLK, SC.RSP
                        .GLOBL  SC.SUR, SC.PSP, SC.REP, SC.NBB
                        .GLOBL  SC.REF, CER.01, CER.02, CNTR.ERR
                        .GLOBL  RDRX.ERR, SPACE4, CRLF, DASH, ASTERISK
                        .GLOBL  HWQ1, 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  SWQ26, SWQ27, SWQ28, 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  SWM1, NULL, SWP.FLAGS, L$HIMEM
                        .GLOBL  L$LUN, L$UNIT

000001                  ON==            1
000002                  OFF==           2
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
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE==   36
000340                  PRI07==         340
```

```
000300                          PRIO6==         300
000240                          PRIO5==         240
000200                          PRIO4==         200
000140                          PRIO3==         140
000100                          PRIO2==         100
000040                          PRIO1==         40
000000                          PRIO0==         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
000536                          .PSECT  $CODE$,  RO

000000  004137  000000G  LRPT:  JSR     R1,$SAVE4                       ;                       5741
000004  104440                 TRAP    40                              ;                       5758
000006  010004                 MOV     R0,R4                           ; *,CUR.PRIORITY
000010  013700  000000G        MOV     BRLEVEL,R0                      ;                       5760
000014  104441                 TRAP    41
000016  012746  000000G        MOV     #RPT1,-(SP)                     ;                       5763
000022  012746  000001         MOV     #1,-(SP)
000026  010600                 MOV     SP,R0                           ; SP,*
000030  104416                 TRAP    16
000032  012716  000000G        MOV     #RPT2,(SP)                      ;                       5764
000036  012746  000001         MOV     #1,-(SP)
000042  010600                 MOV     SP,R0                           ; SP,*
000044  104416                 TRAP    16
000046  012716  000000G        MOV     #RPT3,(SP)                      ;                       5765
000052  012746  000001         MOV     #1,-(SP)
000056  010600                 MOV     SP,R0                           ; SP,*
000060  104416                 TRAP    16
000062  012716  000000G        MOV     #RPT4,(SP)                      ;                       5766
000066  012746  000001         MOV     #1,-(SP)
000072  010600                 MOV     SP,R0                           ; SP,*
000074  104416                 TRAP    16
000076  012716  000000G        MOV     #RPT5,(SP)                      ;                       5767
000102  012746  000001         MOV     #1,-(SP)
000106  010600                 MOV     SP,R0                           ; SP,*
000110  104415                 TRAP    16
```

B12


```
000112  012716  000000G                 MOV     #RPT6,(SP)                      ;                                       5768
000116  012746  000001                  MOV     #1,-(SP)
000122  010600                           MOV     SP,R0                           ; SP,*
000124  104416                           TRAP    16
000126  005002                           CLR     R2                              ; CTLR
000130  010216                  1$:      MOV     R2,(SP)                         ; CTLR,*                                5770
000132  004737  000000V                  JSR     PC,SET.CPAR                                                             5773
000136  012703  000003                  MOV     #3,R3                           ; *,DISK
000142  010316                  2$:      MOV     R3,(SP)                         ; DISK,*                                5775
000144  004737  000000V                  JSR     PC,SET.UPAR                                                             5778
000150  010301                           MOV     R3,R1                           ; DISK,*                                5781
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,*                                5786
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                       ;                                       5789
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                       ;                                       5792
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                       ;                                       5795
000360  005016                           CLR     (SP)
000362  116016  000055                  MOVB    55(R0),(SP)
```

C12


```
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                  ;
000460  062703  000012          3$:     ADD     #12,R3                  ; *,DISK                5784
000464  020327  000041                  CMP     R3,#41                  ; DISK,*                5775
000470  003624                          BLE     2$
000472  010216                          MOV     R2,(SP)                 ; CTLR,*                5800
000474  012746  000126                  MOV     #126,-(SP)
000500  004737  000000G                 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)             ;                       5804
000520  012746  000001                  MOV     #1,-(SP)
000524  010600                          MOV     SP,R0                   ; SP,*
000526  104416                          TRAP    16
000530  010200                          MOV     R2,R0                   ; CTLR,*                5805
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                  ;
000570  005202                  4$:     INC     R2                      ; CTLR                  5803
000572  000243                          .WORD CLV!CLC                                           5770

000574  003002                          BGT     5$
000576  000137  000666'                 JMP     1$
000602  010400                  5$:     MOV     R4,R0                   ; CUR.PRIORITY,*        5811
000604  104441                          TRAP    41
000606  005737  000000G                 TST     RD.COUNT                ;                       5813
000612  001522                          BEQ     9$
000614  012716  000000G                 MOV     #CRLF,(SP)              ;                       5817
```

D12


```
000620  012746  000001                        MOV     #1,-(SP)
000624  010600                                MOV     SP,R0
000626  104416                                TRAP    16                      ; SP,*
000630  012716  000000G                       MOV     #RPT13,(SP)             ;                       5818
000634  012746  000001                        MOV     #1,-(SP)
000640  010600                                MOV     SP,R0
000642  104416                                TRAP    16                      ; SP,*
000644  012716  000000G                       MOV     #RPT14,(SP)             ;                       5819
000650  012746  000001                        MOV     #1,-(SP)
000654  010600                                MOV     SP,R0
000656  104416                                TRAP    16                      ; SP,*
000660  012716  000000G                       MOV     #RPT15,(SP)             ;                       5820
000664  012746  000001                        MOV     #1,-(SP)
000670  010600                                MOV     SP,R0
000672  104416                                TRAP    16                      ; SP,*
000674  005003                                CLR     R3                      ; CTLR                  5821
000676  010316                        6$:     MOV     R3,(SP)                 ; CTLR,*                5823
000700  004737  000000V                       JSR     PC,SET.CPAR
000704  012702  000003                        MOV     #3,R2                   ; *,DISK                5824
000710  010216                        7$:     MOV     R2,(SP)                 ; DISK,*                5826
000712  004737  000000V                       JSR     PC,SET.UPAR
000716  010201                                MOV     R2,R1                   ; DISK,*                5827
000720  006301                                ASL     R1
000722  063701  000000G                       ADD     CST.ADDR,R1
000726  132711  000020                        BITB    #20,(R1)
000732  001432                                BEQ     8$
000734  032711  040000                        BIT     #40000,(R1)
000740  001427                                BEQ     8$
000742  013700  000000G                       MOV     T.ADDR,R0               ;                       5831
000746  016016  000056                        MOV     56(R0),(SP)
000752  016046  000060                        MOV     60(R0),-(SP)
000756  016046  000062                        MOV     62(R0),-(SP)
000762  016046  000064                        MOV     64(R0),-(SP)
000766  111146                                MOVB    (R1),-(SP)
000770  042716  177760                        BIC     #177760,(SP)
000774  013746  000000G                       MOV     L$LUN,-(SP)
001000  012746  000000G                       MOV     #RPT16,-(SP)
001004  012746  000007                        MOV     #7,-(SP)
001010  010600                                MOV     SP,R0
001012  104416                                TRAP    16                      ; SP,*
001014  062706  000016                        ADD     #16,SP
001020  062702  000012                        ADD     #12,R2                  ; *,DISK                5824
001024  020227  000041                8$:     CMP     R2,#41                  ; DISK,*
001030  003727                                BLE     7$
001032  005203                                INC     R3                      ; CTLR                  5821
001034  000243                                .WORD CLV!CLC

001036  003717                                BLE     6$
001040  012716  000000G                       MOV     #CRLF,(SP)              ;                       5840
001044  012746  000001                        MOV     #1,-(SP)
001050  010600                                MOV     SP,R0
001052  104416                                TRAP    16                      ; SP,*
001054  062706  000012                        ADD     #12,SP                  ;                       5816
```

E12


```
001060  012716  000000G                 9$:     MOV     #CRLF,(SP)                      ;                                       5843
001064  012746  000001                          MOV     #1,-(SP)
001070  010600                                  MOV     SP,R0                           ; SP,*
001072  104416                                  TRAP    16
001074  004737  000000V                         JSR     PC,EMS.TIM                      ;                                       5844
001100  062706  000020                          ADD     #20,SP                          ;                                       5741
001104  000207                                  RTS     PC

; Routine Size:  291 words,     Routine Base:  $CODE$ + 0536
; Maximum stack depth per invocation:  40 words



                                                .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000536'                 L$RPT:: JSR     PC,LRPT                         ;                                       5844
000004  104425                                  TRAP    25
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 1644
; Maximum stack depth per invocation:  2 words


;   5847  1
```

F12


```
;  5848  1     %sbttl 'INITIALIZE SECTION'
;  5849  1
;  5850  2     BGNINIT;
;  5851  2
;  5852  2     local
;  5853  2         DELAY_MULT : word,
;  5854  2         FLAG : byte,
;  5855  2         TEMP : word,
;  5856  2         HWPT_REF : ref block [HWPT_LEN, word] field (HWP_FIELDS),
;  5857  2         CLEAR_TABLES : byte,
;  5858  2         SMALLEST_DRIVE : byte,
;  5859  2         BLANKS : WORD INITIAL (%o'020040'),                          !ZZZ
;  5860  2         HWPT_ADDRESS : vector [MAX_UNITS, word];
;  5861  2
;  5862  2     SETPRI (PRI07);                                                  ! NO INTERRUPTS ALLOWED DURING INIT
;  5863  2
;  5864  2     if READEF (EF_NEW)                                               ! IS THIS A NEW PASS?
;  5865  2     then
;  5866  3         begin
;  5867  3         ENTRY_REASON = NEW_PASS;
;  5868  3
;  5869  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;  5870  3         then
;  5871  3             CLEAR_TABLES = FALSE
;  5872  3         else
;  5873  3             CLEAR_TABLES = TRUE;
;  5874  3
;  5875  2         end;
;  5876  2
;  5877  2     if READEF (EF_START)                                             ! IS THIS A START?
;  5878  2     then
;  5879  3         begin
;  5880  3         BRESET;
;  5881  3         ENTRY_REASON = START;
;  5882  3         CLEAR_TABLES = TRUE;
;  5883  3         ADDR_VECT_OK = FALSE;
;  5884  3         INIT_OCCURED = FALSE;
;  5885  3         end;
;  5886  2
;  5887  2     if READEF (EF_RESTART)                                           ! IS THIS A RESTART?
;  5888  2     then
;  5889  3         begin
;  5890  3         ENTRY_REASON = RESTART;
;  5891  3         CLEAR_TABLES = TRUE;
;  5892  3         end;
;  5893  2
;  5894  2     if READEF (EF_CONTINUE)                                          ! IS THIS A CONTINUE?
;  5895  2     then
;  5896  3         begin
;  5897  3         ENTRY_REASON = CONT;
;  5898  3
;  5899  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;  5900  3         then
```

```
;  5901  3            CLEAR_TABLES = FALSE
;  5902  3        else
;  5903  3            CLEAR_TABLES = TRUE;
;  5904  3
;  5905  2        end;
;  5906  2
;  5907  2        if READEF (EF_PWR)                                          ! ARE WE HERE BECAUSE OF POWER FAIL
;  5908  2        then
;  5909  3            begin
;  5910  3            ENTRY_REASON = PWR_FAIL;
;  5911  3            ADDR_VECT_OK = FALSE;
;  5912  3            INIT_OCCURED = FALSE;
;  5913  3            CLEAR_TABLES = TRUE;
;  5914  3            PRINTF (MSG_01);                                        ! "POWER DELAY - WAITING"
;  5915  3
;  5916  3            incr COUNT from 0 to 60 do                              ! WAIT APPROX. 60 SECONDS
;  5917  4                begin
;  5918  4                DELAY_MULT = 333;
;  5919  4                DELAY (.DELAY_MULT);
;  5920  4                BREAK;                                              ! BREAK FOR ACT
;  5921  3                end;
;  5922  3
;  5923  2            end;
;  5924  2
;  5925  2        !SETVEC (O_TVEC, O_BRK, PRI07);                             ! SET ODT TRAP VECTOR
;  5926  2
;  5927  2        !+
;  5928  2        !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
;  5929  2        !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
;  5930  2        !       (DIAGNOSTIC IS ABORTED).
;  5931  2        !-
;  5932  2
;  5933  2        if .L$UNIT gtru MAX_UNITS
;  5934  2        then
;  5935  3            begin
;  5936  3            ERRSF (1, EGS_01, EMS_01);
;  5937  3            DOCLN;
;  5938  3            end;
;  5939  2
;  5940  2        !+
;  5941  2        !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
;  5942  2        !       ALL RUN-TIME CONTROLLER STATUS TABLES (CSTs) ARE CLEARED TO 0, THEN
;  5943  2        !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
;  5944  2        !-
;  5945  2
;  5946  2        if .ENTRY_REASON neq NEW_PASS
;  5947  2        then
;  5948  3            begin
;  5949  3            SMALLEST_DRIVE = 255;                                   ! LARGEST DISK NO. ALLOWED BY MSCP
;  5950  3
;  5951  3            incr COUNT from 0 to ((MAX_CTLR * CST_LEN * 2) - 2) by 2 do
;  5952  3                (CST + .COUNT) = 0;
;  5953  3
```

```
;  5954  3          incr UNIT from 0 to (.L$UNIT - 1) do                                    ! LOOP THROUGH ALL UNITS
;  5955  3
;  5956  3              if (HWPT_ADDRESS [.UNIT] = GPHARD (.UNIT, HWPT_REF)) neqa 0         ! IF HWP TABLE FOUND
;  5957  3              then
;  5958  3
;  5959  3                  if .HWPT_REF [HWP_DISK_NUM] lssu .SMALLEST_DRIVE                ! FIND OUT THE SMALLEST DISK NUMBER
;  5960  3                  then
;  5961  3                      SMALLEST_DRIVE = .HWPT_REF [HWP_DISK_NUM];
;  5962  3
;  5963  3          incr UNIT from 0 to (.L$UNIT - 1) do                                    ! LOOP THROUGH ALL UNITS
;  5964  3
;  5965  3              if .HWPT_ADDRESS [.UNIT] neqa 0                                     ! IF HWP TABLE FOUND
;  5966  3              then
;  5967  4                  begin
;  5968  4                  FLAG = NOT_FOUND;
;  5969  4                  HWPT_REF = .HWPT_ADDRESS [.UNIT];
;  5970  4
;  5971  4                  incr CTLR from 0 to (MAX_CTLR - 1) do                           ! LOOP THROUGH ALL CSTs
;  5972  4
;  5973  4                      if .CST [.CTLR, IP_ADDR] eqla .HWPT_REF [HWP_IP_ADDR]
;  5974  4                      then
;  5975  4
;  5976  4                          if .CST [.CTLR, (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE
;  5977  4                                  + OF_UN + OF_DATA, D_PRES] eql NOT_PRESENT
;  5978  4                          then
;  5979  5                              begin                                           ! IF EMPTY SLOT FOUND
;  5980  5                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;  5981  5                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;  5982  5                                                                              ! COPY DISK ADDR AND PROT BIT
;  5983  5                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;  5984  5                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;  5985  5                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;  5986  5
;  5987  5                              IF .HWPT_REF [HWP_ENTIRE] EQL TRUE              !ZZZ IF DEFAULT TEST RANGE,
;  5988  5                              THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;        !ZZZ MAKE HI ADDR ALL ONES
;  5989  5
;  5990  5                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =
;  5991  5                                      .HWPT_REF [HWP_BEG_TRK];                !ZZZ
;  5992  5                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =          !ZZZ
;  5993  5                                      .HWPT_REF [HWP_BEG_TRK1];               !ZZZ
;  5994  5                              CST [.CTLR, .TEMP + OF_END, D_END0] =           !ZZZ
;  5995  5                                      .HWPT_REF [HWP_END_TRK];                !ZZZ
;  5996  5                              CST [.CTLR, .TEMP + OF_END1, D_END1] =          !ZZZ
;  5997  5                                      .HWPT_REF [HWP_END_TRK1];               !ZZZ
;  5998  5
;  5999  5                              CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;  6000  5                              CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;        !ZZZ BLANK NAME
;  6001  5
;  6002  5
;  6003  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;  6004  5                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
;  6005  5                                      NOT (.HWPT_REF [HWP_DISK_DUPEX]);       !ZZZ
;  6006  5
```

```
;  6007  5                        CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =     !ZZZ
;  6008  5                              (.HWPT_REF [HWP_DISK_DUPWT]);              !ZZZ
;  6009  5                        CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;      !ZZZ
;  6010  5                        FLAG = FOUND;
;  6011  5                        exitloop;
;  6012  5                        end
;  6013  4                    else
;  6014  5                        begin                                            ! DUPLICATE UNIT
;  6015  5                        PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR]);
;  6016  5                                                                         ! "DUPLICATE UNIT: X AT IP: XXXXXX"
;  6017  5                        DUR [.UNIT] = DU_CONF;                           ! CONFIGURATION ERROR
;  6018  5                        DODU (.UNIT);                                    ! DROP UNIT
;  6019  5                        FLAG = FOUND;
;  6020  5                        exitloop;
;  6021  4                        end;
;  6022  4
;  6023  4                IF .FLAG eql NOT_FOUND                                   ! IF NO IP MATCH TO EXISTING CST
;  6024  4                THEN
;  6025  5                    begin
;  6026  5
;  6027  5                    incr CTLR from 0 to (MAX_CTLR - 1) do                ! LOOP THROUGH EACH CST
;  6028  5
;  6029  5                        if .CST [.CTLR, IP_ADDR] eqla 0                  ! IF EMPTY CST FOUND
;  6030  5                        then
;  6031  6                            begin
;  6032  6                            CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;  6033  6                            CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;  6034  6                            CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;  6035  6                            TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;  6036  6                            CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;  6037  6                                                                         ! COPY DISK ADDR AND PROT BIT
;  6038  6                            CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;  6039  6                            CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;  6040  6                            CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;  6041  6
;  6042  6                            IF .HWPT_REF [HWP_ENTIRE] EQL TRUE           !ZZZ IF DEFAULT TEST RANGE,
;  6043  6                            THEN HWPT_REF [HWP_END_TRK1] = ALL_ONES;     !ZZZ MAKE HI ADDR ALL ONES
;  6044  6
;  6045  6                            CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =        !ZZZ
;  6046  6                                  .HWPT_REF [HWP_BEG_TRK];               !ZZZ
;  6047  6                            CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =       !ZZZ
;  6048  6                                  .HWPT_REF [HWP_BEG_TRK1];              !ZZZ
;  6049  6                            CST [.CTLR, .TEMP + OF_END, D_END0] =        !ZZZ
;  6050  6                                  .HWPT_REF [HWP_END_TRK];               !ZZZ
;  6051  6                            CST [.CTLR, .TEMP + OF_END1, D_END1] =       !ZZZ
;  6052  6                                  .HWPT_REF [HWP_END_TRK1];              !ZZZ
;  6053  6
;  6054  6                            CST [.CTLR, .TEMP + OF_NAME_0, D_ALL] = .BLANKS;     !ZZZ BLANK NAME
;  6055  6                            CST [.CTLR, .TEMP + OF_NAME_2, D_ALL] = .BLANKS;     !ZZZ BLANK NAME
;  6056  6
;  6057  6
;  6058  6                            CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;     !ZZZ
;  6059  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
```

K12


```
;  6113  3                CTLR_CNT = .CTLR_CNT + 1;                                  ! INCREMENT CONTROLLER COUNT
;  6114  3
;  6115  3            MEMORY (FREE_MEM_ADDR);                                        ! GET START OF FREE MEMORY
;  6116  3
;  6117  2            end;                                                           ! END OF "START" INITIALIZATION
;  6118  2
;  6119  2            !+
;  6120  2            !       CLEAR STATISTICS TABLES
;  6121  2            !-
;  6122  2
;  6123  2            incr UNITS from 0 to MAX_UNITS - 1 do                          ! CLEAR CURRENT STATISTICS
;  6124  2                incr COUNT from 0 to TALLY_CLEAR - 1 do
;  6125  2                    TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;
;  6126  2
;  6127  2            if .CLEAR_TABLES                                               ! IF CLEAR TABLES ON EVERY PASS
;  6128  2            then
;  6129  2                incr UNITS from 0 to MAX_UNITS - 1 do
;  6130  2                    incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do        ! INITIALIZE TOTALS
;  6131  2                        TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;           !
;  6132  2
;  6133  2            if .CLEAR_TABLES
;  6134  2            then
;  6135  2                incr CTLR from 0 to MAX_CTLR - 1 do
;  6136  3                    begin
;  6137  3                    C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;                      ! INITALIZE CONTROLLER ERRORS
;  6138  3                    C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;                      !
;  6139  2                    end;
;  6140  2
;  6141  2            !+
;  6142  2            !       MISCELLANEOUS INITIALIZATON
;  6143  2            !-
;  6144  2
;  6145  2            incr CTLR from 0 to (MAX_CTLR - 1) do                          ! INIT NO. OF OUTSTANDING QIOs
;  6146  2                QIO [.CTLR] = 0;
;  6147  2
;  6148  2            incr COUNT from 0 to (RP_CNT - 1) do                           ! INITIALIZE RETURN PACKET POOL
;  6149  2                RP_USE [.COUNT] = -1;
;  6150  2
;  6151  2            if .CLK_PRESENT                                                ! STOP CLOCK IF PRESENT
;  6152  2            then
;  6153  2                LINE_CLOCK = 0;
;  6154  2
;  6155  2            IODQ_IN = IODQ_OUT = 0;                                        ! INIT I/O DONE QUEUE POINTERS
;  6156  2            CRN_LOW = CRN_HIGH = 0;                                        ! INIT COMMAND REFERENCE NUMBER
;  6157  2            SETPRI (PRI00);                                                ! SET PROGRAM PRIORITY TO 0
;  6158  2
;  6159  2
;  6160  1            ENDINIT;

                                        .GLOBL  L$DLY
```

L12


```
                                        .SBTTL  LINIT INITIALIZE SECTION
000000 004137 000000G          LINIT:   JSR     R1,$SAVE5                       ;                                       5846
000004 162706 000030                    SUB     #30,SP
000010 012746 020040                    MOV     #20040,-(SP)                    ; *,BLANKS
000014 012700 000340                    MOV     #340,R0                         ;                                       5862
000020 104441                           TRAP    41
000022 012700 000035                    MOV     #35,R0                          ;                                       5864
000026 104447                           TRAP    47
000030 103014                           BHIS    2$
000032 112737 000005 000000G            MOVB    #5,ENTRY.REASON                 ;                                       5867
000040 105737 000000G                   TSTB    SWP.FLAGS                       ;                                       5869
000044 100403                           BMI     1$
000046 105066 000012                    CLRB    12(SP)                          ; CLEAR.TABLES                          5871
000052 000403                           BR      2$                              ;                                       5869
000054 112766 000001 000012    1$:      MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5873
000062 012700 000040           2$:      MOV     #40,R0                          ;                                       5877
000066 104447                           TRAP    47
000070 103013                           BHIS    3$
000072 104433                           TRAP    33
000074 112737 000001 000000G            MOVB    #1,ENTRY.REASON                 ;                                       5879
000102 112766 000001 000012             MOVB    #1,12(SP)                       ;                                       5881
000110 105037 000000G                   CLRB    ADDR.VECT.OK                    ; *,CLEAR.TABLES                        5882
000114 105037 000000G                   CLRB    INIT.OCCURED                    ;                                       5883
000120 012700 000037           3$:      MOV     #37,R0                          ;                                       5884
000124 104447                           TRAP    47                              ;                                       5887
000126 103006                           BHIS    4$
000130 112737 000002 000000G            MOVB    #2,ENTRY.REASON                 ;                                       5890
000136 112766 000001 000012             MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5891
000144 012700 000036           4$:      MOV     #36,R0                          ;                                       5894
000150 104447                           TRAP    47
000152 103014                           BHIS    6$
000154 112737 000003 000000G            MOVB    #3,ENTRY.REASON                 ;                                       5897
000162 105737 000000G                   TSTB    SWP.FLAGS                       ;                                       5899
000166 100403                           BMI     5$
000170 105066 000012                    CLRB    12(SP)                          ; CLEAR.TABLES                          5901
000174 000403                           BR      6$                              ;                                       5899
000176 112766 000001 000012    5$:      MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5903
000204 012700 000034           6$:      MOV     #34,R0                          ;                                       5907
000210 104447                           TRAP    47
000212 103043                           BHIS    12$
000214 112737 000004 000000G            MOVB    #4,ENTRY.REASON                 ;                                       5910
000222 105037 000000G                   CLRB    ADDR.VECT.OK                    ;                                       5911
000226 105037 000000G                   CLRB    INIT.OCCURED                    ;                                       5912
000232 112766 000001 000012             MOVB    #1,12(SP)                       ; *,CLEAR.TABLES                        5913
000240 012746 000000G                   MOV     #MSG.01,-(SP)                   ;                                       5914
000244 012746 000001                    MOV     #1,-(SP)
000250 010600                           MOV     SP,R0                           ; SP,*
000252 104417                           TRAP    17
000254 012702 000075                    MOV     #75,R2                          ; *,COUNT                               5916
000260 012703 000515           7$:      MOV     #515,R3                         ; *,DELAY.MULT                          5918
000264 010301                           MOV     R3,R1                           ; DELAY.MULT,$$TMP2                     5919
000266 001411                  8$:      BEQ     11$
000270 013700 000000G                   MOV     L$DLY,R0                        ; *,$$TMP1
```

M12


```
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$:     TRAP    22
000314  005302                          DEC     R2                      ; COUNT                         5916
000316  001360                          BNE     7$
000320  022626                          CMP     (SP)+,(SP)+             ;                               5909
000322  023727  000000G 000004 12$:     CMP     L$UNIT,#4               ;                               5933
000330  101405                          BLOS    13$
000332  104454                          TRAP    54                      ;                               5936
000334  000001                          .WORD   1
000336  000000G                         .WORD   EGS.01
000340  000000V                         .WORD   EMS.01
000342  104444                          TRAP    44
000344  123727  000000G 000005 13$:     CMPB    ENTRY.REASON,#5         ;                               5946
000352  001002                          BNE     14$
000354  000137  003754'                 JMP     43$
000360  112766  000377  000010 14$:     MOVB    #377,10(SP)             ; *,SMALLEST.DRIVE              5949
000366  005000                          CLR     R0                      ; COUNT                         5951
000370  005060  000000G        15$:     CLR     CST(R0)                 ; *(COUNT)                      5952
000374  062700  000002                   ADD     #2,R0                   ; *,COUNT                       5951
000400  020027  000124                   CMP     R0,#124                 ; COUNT,*
000404  003771                          BLE     15$
000406  013704  000000G                 MOV     L$UNIT,R4               ;                               5954
000412  005003                          CLR     R3                      ; UNIT
000414  000435                          BR      18$
000416  010302                 16$:     MOV     R3,R2                   ; UNIT,*                        5956
000420  006302                          ASL     R2
000422  012700  000022                   MOV     #22,R0
000426  060600                          ADD     SP,R0                   ; HWPT.ADDRESS,*
000430  060002                          ADD     R0,R2
000432  010300                          MOV     R3,R0                   ; UNIT,*
000434  104442                          TRAP    42
000436  010001                          MOV     R0,R1                   ; *,HWPT.REF
000440  010112                          MOV     R1,(R2)                 ; HWPT.REF,*
000442  001421                          BEQ     17$
000444  005002                          CLR     R2                      ;                               5959
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),*                 5961
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$:     INC     R3                      ; UNIT                          5954
000510  020304                 18$:     CMP     R3,R4                   ; UNIT,*
000512  002741                          BLT     16$
000514  013766  000000G 000016          MOV     L$UNIT,16(SP)           ;                               5963
```

N12


```
000522  005004                          CLR     R4                              ; UNIT
000524  000137  003732'                  JMP     41$
000530  010400                  19$:     MOV     R4,R0                           ; UNIT,*                        5965
000532  006300                          ASL     R0
000534  012703  000022                    MOV     #22,R3
000540  060603                          ADD     SP,R3                           ; HWPT.ADDRESS,*
000542  060300                          ADD     R3,R0
000544  005710                          TST     (R0)
000546  001002                          BNE     20$
000550  000137  003730'                  JMP     40$
000554  105066  000006          20$:     CLRB    6(SP)                           ; FLAG                          5968
000560  011001                          MOV     (R0),R1                         ; *,HWPT.REF                    5969
000562  005066  000002                    CLR     2(SP)                           ; CTLR                          5971
000566  016646  000002          21$:     MOV     2(SP),-(SP)                     ; CTLR,*                        5973
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  003152'                  JMP     28$
000616  012766  000001  000014  22$:     MOV     #1,14(SP)                       ;                               6010
000624  112766  000001  000006          MOVB    #1,6(SP)                        ; *,FLAG
000632  012705  000006                    MOV     #6,R5                           ;                               5976
000636  060105                          ADD     R1,R5                           ; HWPT.REF,*
000640  111546                          MOVB    (R5),-(SP)
000642  042716  177760                    BIC     #177760,(SP)
000646  005000                          CLR     R0
000650  156600  000012                    BISB    12(SP),R0                       ; SMALLEST.DRIVE,*
000654  160016                          SUB     R0,(SP)
000656  012746  000012                    MOV     #12,-(SP)
000662  004737  000000G                   JSR     PC,BL$MUL
000666  010066  000010                    MOV     R0,10(SP)
000672  005726                          TST     (SP)+
000674  016616  000004                    MOV     4(SP),(SP)                      ; CTLR,*                        5977
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                        5980
000736  062702  000003                    ADD     #3,R2                           ; *,TEMP
000742  010300                          MOV     R3,R0                           ;                               5981
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,*                        5983
000760  000316                          SWAB    (SP)
000762  042716  170377                    BIC     #170377,(SP)
000766  042710  007400                    BIC     #7400,(R0)
```

B13


```
000772  052610                          BIS     (SP)+,(R0)
000774  042710  010000                  BIC     #10000,(R0)
001000  052710  040000                  BIS     #40000,(R0)
001004  105715                          TSTB    (R5)                    ;                                       5984
001006  100003                          BPL     23$                     ;                                       5985
001010  012761  177777  000016          MOV     #-1,16(R1)              ;                                       5987
001016  010300                  23$:    MOV     R3,R0                   ; *,*(HWPT.REF)                         5988
001020  060200                          ADD     R2,R0                   ;                                       5990
001022  006300                          ASL     R0                      ; TEMP,*
001024  016160  000010  000002G         MOV     10(R1),CST+2(R0)        ; *(HWPT.REF),*
001032  010300                          MOV     R3,R0                   ;                                       5992
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                   ;                                       5994
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                   ;                                       5996
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                   ;                                       5999
001100  060200                          ADD     R2,R0                   ; TEMP,*
001102  006300                          ASL     R0
001104  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,*
001110  010300                          MOV     R3,R0                   ;                                       6000
001112  060200                          ADD     R2,R0                   ; TEMP,*
001114  006300                          ASL     R0
001116  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,*
001122  010300                          MOV     R3,R0                   ;                                       6003
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)              ;                                       6005
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$:    TST     (SP)+
001176  111516                          MOVB    (R5),(SP)               ;                                       6007
001200  042710  010000                  BIC     #10000,(R0)
001204  032726  000100                  BIT     #100,(SP)+
001210  001402                          BEQ     26$
001212  052710  010000                  BIS     #10000,(R0)
001216  010300                  26$:    MOV     R3,R0                   ;                                       6009
```

```
001220  060200                          ADD     R2,R0                           ; TEMP,*
001222  006300                          ASL     R0
001224  005060  000022G                 CLR     CST+22(R0)
001230  000430                          BR      29$                             ;                       5979
001232  011146                  27$:    MOV     (R1),-(SP)                      ; HWPT.REF,*            6015
001234  111546                          MOVB    (R5),-(SP)
001236  042716  177760                  BIC     #177760,(SP)
001242  012746  000000G                 MOV     #CER.01,-(SP)
001246  012746  000003                  MOV     #3,-(SP)
001252  010600                          MOV     SP,R0                           ; SP,*
001254  104417                          TRAP    17
001256  062706  000010                  ADD     #10,SP
001262  112764  000001  000000G         MOVB    #1,DUR(R4)                      ; *,*(UNIT)             6017
001270  010400                          MOV     R4,R0                           ; UNIT,*                6018
001272  104451                          TRAP    51
001274  000406                          BR      29$                             ;                       6014
001276  005266  000002          28$:    INC     2(SP)                           ; CTLR                  5971
001302  000243                          .WORD CLV!CLC

001304  003002                          BGT     29$
001306  000137  002442'                JMP     21$
001312  105766  000006          29$:    TSTB    6(SP)                           ; FLAG                  6023
001316  001402                          BEQ     30$
001320  000137  003730'                JMP     40$
001324  005066  000014          30$:    CLR     14(SP)                          ; CTLR                  6027
001330  016646  000014          31$:    MOV     14(SP),-(SP)                    ; CTLR,*                6029
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  003650'                JMP     37$
001360  011160  000000G          32$:    MOV     (R1),CST(R0)                    ; HWPT.REF,*            6032
001364  016103  000002                  MOV     2(R1),R3                        ; *(HWPT.REF),*         6033
001370  042703  177000                  BIC     #177000,R3
001374  042760  000777  000002G         BIC     #777,CST+2(R0)
001402  050360  000002G                 BIS     R3,CST+2(R0)
001406  116160  000004  000004G         MOVB    4(R1),CST+4(R0)                 ; *(HWPT.REF),*         6034
001414  012705  000006                  MOV     #6,R5                           ;                       6035
001420  060105                          ADD     R1,R5                           ; HWPT.REF,*
001422  111546                          MOVB    (R5),-(SP)
001424  042716  177760                  BIC     #177760,(SP)
001430  005000                          CLR     R0
001432  156600  000012                  BISB    12(SP),R0                       ; SMALLEST.DRIVE,*
001436  160016                          SUB     R0,(SP)
001440  012746  000012                  MOV     #12,-(SP)
001444  004737  000000G                 JSR     PC,BL$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,*                6036
001464  012746  000053                  MOV     #53,-(SP)
001470  004737  000000G                 JSR     PC,BL$MUL
```

D13


```
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,*                        6038
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)             ;                               6039
001534  052710  040000                  BIS     #40000,(R0)             ;                               6040
001540  105715                          TSTB    (R5)                    ;                               6042
001542  100003                          BPL     33$
001544  012761  177777  000016          MOV     #-1,16(R1)              ; *,*(HWPT.REF)                 6043
001552  010300                  33$:    MOV     R3,R0                   ;                               6045
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                   ;                               6047
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                   ;                               6049
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                   ;                               6051
001620  060200                          ADD     R2,R0                   ; TEMP,*
001622  006300                          ASL     R0
001624  016160  000016  000010G         MOV     16(R1),CST+10(R0)       ; *(HWPT.REF),*
001632  010300                          MOV     R3,R0                   ;                               6054
001634  060200                          ADD     R2,R0                   ; TEMP,*
001636  006300                          ASL     R0
001640  011660  000012G                 MOV     (SP),CST+12(R0)         ; BLANKS,*
001644  010300                          MOV     R3,R0                   ;                               6055
001646  060200                          ADD     R2,R0                   ; TEMP,*
001650  006300                          ASL     R0
001652  011660  000014G                 MOV     (SP),CST+14(R0)         ; BLANKS,*
001656  010300                          MOV     R3,R0                   ;                               6058
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)              ;                               6060
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)
```

E13


```
001720  006026                          ROR     (SP)+
001722  103002                          BCC     35$
001724  052710  100000                  BIS     #100000,(R0)
001730  005726                  35$:    TST     (SP)+
001732  111516                          MOVB    (R5),(SP)                       ;                               6061
001734  042710  010000                  BIC     #10000,(R0)
001740  032726  000100                  BIT     #100,(SP)+
001744  001402                          BEQ     36$
001746  052710  010000                  BIS     #10000,(R0)
001752  010300                  36$:    MOV     R3,R0                           ;                               6063
001754  060200                          ADD     R2,R0                           ; TEMP,*
001756  006300                          ASL     R0
001760  005060  000022G                 CLR     CST+22(R0)
001764  112766  000001  000006          MOVB    #1,6(SP)                        ; *,FLAG                        6064
001772  000410                          BR      39$                             ;                               6031
001774  005266  000014          37$:    INC     14(SP)                          ; CTLR                          6027
002000  000243                          .WORD CLV!CLC

002002  003002                          BGT     38$
002004  000137  003204'                JMP     31$
002010  105766  000006          38$:    TSTB    6(SP)                           ; FLAG                          6068
002014  001017                  39$:    BNE     40$
002016  012746  000001                  MOV     #1,-(SP)                        ;                               6071
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)                     6072
002044  010400                          MOV     R4,R0                           ; UNIT,*                        6073
002046  104451                          TRAP    51
002050  062706  000006                  ADD     #6,SP                           ;                               6070
002054  005204                  40$:    INC     R4                              ; UNIT                          5963
002056  020466  000016          41$:    CMP     R4,16(SP)                       ; UNIT,*
002062  002002                          BGE     42$
002064  000137  002404'                JMP     19$
002070  123727  000000G 000005  42$:    CMPB    ENTRY.REASON,#5                 ;                               6086
002076  001051                          BNE     48$
002100  013703  000000G         43$:    MOV     L$UNIT,R3                       ;                               6090
002104  005004                          CLR     R4                              ; UNIT
002106  000404                          BR      45$
002110  010400                  44$:    MOV     R4,R0                           ; UNIT,*                        6091
002112  104442                          TRAP    42
002114  010001                          MOV     R0,R1                           ; *,HWPT.REF
002116  005204                          INC     R4                              ; UNIT                          6090
002120  020403                  45$:    CMP     R4,R3                           ; UNIT,*
002122  002772                          BLT     44$
002124  005003                          CLR     R3                              ; CTLR                          6093
002126  010346                  46$:    MOV     R3,-(SP)                        ; CTLR,*                        6095
002130  012746  000126                  MOV     #126,-(SP)
002134  004737  000000G                 JSR     PC,BL$MUL
002140  105060  000005G                 CLRB    CST+5(R0)
002144  010316                          MOV     R3,(SP)                         ; CTLR,*                        6098
002146  012746  000053                  MOV     #53,-(SP)
```

F13


```
002152  004737  000000G                        JSR     PC,BL$MUL
002156  012701  000003                         MOV     #3,R1                   ; *,OFFSET                      6097
002162  010002                          47$:    MOV     R0,R2                   ;                               6098
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                      6097
002202  020127  000041                         CMP     R1,#41                  ; OFFSET,*
002206  003765                                  BLE     47$
002210  062706  000006                         ADD     #6,SP                   ;                               6094
002214  005203                                  INC     R3                      ; CTLR                          6093
002216  000243                                  .WORD CLV!CLC

002220  003742                                  BLE     46$
002222  123727  000000G 000001         48$:    CMPB    ENTRY.REASON,#1         ;                               6104
002230  001017                                  BNE     51$
002232  005037  000000G                        CLR     CTLR.CNT                ;                               6107
002236  005000                                  CLR     R0                      ; CTLR                          6109
002240  005760  000000G                 49$:    TST     CST(R0)                 ; *(CTLR)                       6111
002244  001402                                  BEQ     50$
002246  005237  000000G                        INC     CTLR.CNT                ;                               6113
002252  062700  000126                 50$:    ADD     #126,R0                 ; *,CTLR                        6109
002256  000243                                  .WORD CLV!CLC

002260  003767                                  BLE     49$
002262  104431                                  TRAP    31
002264  010037  000000G                        MOV     R0,FREE.MEM.ADDR        ;                               6115
002270  005001                          51$:    CLR     R1                      ; UNITS                         6123
002272  005003                          52$:    CLR     R3                      ; COUNT                         6124
002274  010300                          53$:    MOV     R3,R0                   ; COUNT,*                       6125
002276  060100                                  ADD     R1,R0                   ; UNITS,*
002300  006300                                  ASL     R0
002302  005060  000000G                        CLR     TALLY(R0)
002306  005203                                  INC     R3                      ; COUNT                         6124
002310  020327  000006                         CMP     R3,#6                   ; COUNT,*
002314  003767                                  BLE     53$
002316  062701  000033                         ADD     #33,R1                  ; *,UNITS                       6123
002322  020127  000121                         CMP     R1,#121                 ; UNITS,*
002326  003761                                  BLE     52$
002330  032766  000001  000012                 BIT     #1,12(SP)               ; *,CLEAR.TABLES                6127
002336  001436                                  BEQ     57$
002340  005001                                  CLR     R1                      ; UNITS                         6129
002342  012703  000007                 54$:    MOV     #7,R3                   ; *,COUNT                       6130
002346  010300                          55$:    MOV     R3,R0                   ; COUNT,*                       6131
002350  060100                                  ADD     R1,R0                   ; UNITS,*
002352  006300                                  ASL     R0
002354  005060  000000G                        CLR     TALLY(R0)
002360  005203                                  INC     R3                      ; COUNT                         6130
002362  020327  000032                         CMP     R3,#32                  ; COUNT,*
002366  003767                                  BLE     55$
002370  062701  000033                         ADD     #33,R1                  ; *,UNITS                       6129
002374  020127  000121                         CMP     R1,#121                 ; UNITS,*
002400  003760                                  BLE     54$
```

```
002402  032766  000001  000012                  BIT     #1,12(SP)                       ; *,CLEAR.TABLES                        6133
002410  001411                                  BEQ     57$
002412  005000                                  CLR     R0                              ; CTLR                                  6135
002414  105060  000000G                 56$:    CLRB    C.ERR.TBL(R0)                   ; *(CTLR)                               6137
002420  105060  000001G                         CLRB    C.ERR.TBL+1(R0)                 ; *(CTLR)                               6138
002424  062700  000002                          ADD     #2,R0                           ; *,CTLR                                6135
002430  000243                                  .WORD CLV!CLC

002432  003770                                  BLE     56$
002434  005000                          57$:    CLR     R0                              ; CTLR                                  6145
002436  105060  000000G                 58$:    CLRB    QIO(R0)                         ; *(CTLR)                               6146
002442  005200                                  INC     R0                              ; CTLR                                  6145
002444  000243                                  .WORD CLV!CLC

002446  003773                                  BLE     58$
002450  005000                                  CLR     R0                              ; COUNT                                 6148
002452  112760  000377  000000G         59$:    MOVB    #377,RP.USE(R0)                 ; *,*(COUNT)                            6149
002460  005200                                  INC     R0                              ; COUNT                                 6148
002462  020027  000007                          CMP     R0,#7                           ; COUNT,*
002466  003771                                  BLE     59$
002470  132737  000001  000000G                 BITB    #1,CLK.PRESENT                  ;                                       6151
002476  001402                                  BEQ     60$
002500  005037  177546                          CLR     @#177546                        ;                                       6153
002504  005037  000000G                 60$:    CLR     IODQ.OUT                        ;                                       6155
002510  005037  000000G                         CLR     IODQ.IN
002514  005037  000000G                         CLR     CRN.HIGH                        ;                                       6156
002520  005037  000000G                         CLR     CRN.LOW
002524  005000                                  CLR     R0                              ;                                       6157
002526  104441                                  TRAP    41
002530  062706  000032                          ADD     #32,SP                          ;                                       5846
002534  000207                                  RTS     PC

; Routine Size:  687 words,    Routine Base:  $CODE$ + 1654
; Maximum stack depth per invocation:  25 words



                                                .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001654'         L$INIT::JSR     PC,LINIT                        ;                                       6157
000004  104411                                  TRAP    11
000006  000207                                  RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 4412
; Maximum stack depth per invocation:  2 words
```

H13


```
;   6161  1     %sbttl 'AUTODROP SECTION'
;   6162  1
;   6163  1     !+
;   6164  1     ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;   6165  1     ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;   6166  1     ! SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;   6167  1     ! DROPPED FROM TESTING.
;   6168  1     !-
;   6169  1
;   6170  2     BGNAUTO;
;   6171  2
;   6172  2     !if BIT_TST (SWP_FLAGS, SWF_TRC)
;   6173  2     !then
;   6174  2     !   PRINTF (DBM3);
;   6175  2
;   6176  2     return;
;   6177  2
;   6178  1     ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                              ;  6160

; Routine Size:  1 word,        Routine Base:  $CODE$ + 4422
; Maximum stack depth per invocation:  0 words



                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  004422'         L$AUTO::JSR     PC,LAUTO                        ;  6176
000004  104461                          TRAP    61
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4424
; Maximum stack depth per invocation:  2 words
```

```
;   6179  1     %sbttl 'CLEANUP CODING SECTION'
;   6180  1
;   6181  1     !+
;   6182  1     ! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;   6183  1     ! AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;   6184  1     !-
;   6185  1
;   6186  2     BGNCLN;
;   6187  2
;   6188  2     LABEL
;   6189  2         LZ1;                                                                    !ZZZ
;   6190  2                                                                                 !ZZZ
;   6191  2     DORPT;
;   6192  2
;   6193  2     !CLRVEC (0_TVEC);                                   ! RETURN ODT TRAP TO DIAGNOSTIC SUPERVISER
;   6194  2
;   6195  2     if .CLK_PRESENT
;   6196  2     then
;   6197  3         begin
;   6198  3         LINE_CLOCK = 0;                                 ! STOP THE LINE-CLOCK
;   6199  3     !   CLRVEC (%o'100');                               ! RETURN LINE-CLOCK'S VECTOR TO SUPERVISOR
;   6200  2         end;
;   6201  2
;   6202  2     incr CTLR from 0 to (MAX_CTLR - 1) do               ! FOR EACH CONTROLLER
;   6203  2
;   6204  2         if (RDRX_ADDR = .CST [.CTLR, IP_ADDR]) neqa 0   ! IF CONTROLLER EXISTS
;   6205  2         then
;   6206  3             begin
;   6207  3
;   6208  3             if .ADDR_VECT_OK
;   6209  3             then
;   6210  4     LZ1:        begin                                                           !ZZZ
;   6211  4
;   6212  4                 if .DCT [.CTLR, STAT] eql ONLINE        ! IF CONTROLLER ALIVE
;   6213  4                 then
;   6214  4
;   6215  4     !MMM                    incr COUNT from 1 to 10000 do
;   6216  4                             incr COUNT from 1 to 200 do                         !MMM
;   6217  5                         begin
;   6218  5                         DELAY (1);
;   6219  5                         BREAK;                                                  !MMM
;   6220  5                         if .DCT [.CTLR, CRING_CNT] eql 0    ! WAIT TILL OUTSTANDING COMMANDS FINISHED
;   6221  5                         then
;   6222  5
;   6223  5                             INCR Z  FROM 0 TO 3  DO                             !ZZZ
;   6224  6                                 BEGIN                                           !ZZZ
;   6225  6                                 TEMP1 =  (.DCT [.CTLR, RR_BEG]) + 4 * .Z;       !DESCRIPTOR ADDRESS   !ZZZ
;   6226  6                                 TEMP2 = ..TEMP1;                                !PACKET ADDRESS       !ZZZ
;   6227  6                                 IF ..TEMP2  EQL CRN_LOW                         !CRN                  !ZZZ
;   6228  6                                 THEN                                            !IF THE LAST CRN IS BACK, !ZZZ
;   6229  6                                     (WRT_RDRX (RCIP, RC_ALL, ALL_ONES); LEAVE LZ1); !THEN STOP WAITING !ZZZ
;   6230  5                                 END;                                            !ZZZ
;   6231  4                         end;
```

J13


```
;   6232  4
;   6233  4                WRT_RDRX (RCIP, RC_ALL, ALL_ONES);          ! WRITE IP TO STOP DEVICE
;   6234  3                end;
;   6235  3
;   6236  3            CLRVEC (.CST[.CTLR, VEC_ADDR]);                 ! RETURN CONTROLLER'S TRAP VECTOR TO SUPERVISOR
;   6237  2            end;
;   6238  2
;   6239  1     ENDCLN;


                                        .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  004137  000000G                 LCLEAN: JSR     R1,$SAVE5                                       ;                       6178
000004  005746                                  TST     -(SP)
000006  104424                                  TRAP    24                                              ;                       6189
000010  132737  000001  000000G                BITB    #1,CLK.PRESENT                                  ;                       6195
000016  001402                                  BEQ     1$
000020  005037  177546                          CLR     @#177546                                        ;                       6198
000024  005005                          1$:     CLR     R5                                              ; CTLR                  6202
000026  010546                          2$:     MOV     R5,-(SP)                                        ; CTLR,*                6204
000030  012746  000126                          MOV     #126,-(SP)
000034  004737  000000G                         JSR     PC,BL$MUL
000040  010003                                  MOV     R0,R3
000042  022626                                  CMP     (SP)+,(SP)+
000044  016337  000000G 000000G                MOV     CST(R3),RDRX.ADDR
000052  001501                                  BEQ     13$
000054  132737  000001  000000G                BITB    #1,ADDR.VECT.OK                                 ;                       6208
000062  001470                                  BEQ     12$
000064  010546                                  MOV     R5,-(SP)                                        ; CTLR,*                6212
000066  012746  000022                          MOV     #22,-(SP)
000072  004737  000000G                         JSR     PC,BL$MUL
000076  010001                                  MOV     R0,R1
000100  022626                                  CMP     (SP)+,(SP)+
000102  005761  000000G                         TST     DCT(R1)
000106  100052                                  BPL     11$
000110  012704  000310                          MOV     #310,R4                                         ; *,COUNT               6216
000114  012702  000001                  3$:     MOV     #1,R2                                           ; *,$$TMP2              6218
000120  001410                          4$:     BEQ     7$
000122  013700  000000G                         MOV     L$DLY,R0                                        ; *,$$TMP1
000126  001403                                  BEQ     6$
000130  005016                          5$:     CLR     (SP)                                            ; $$TMP
000132  005300                                  DEC     R0                                              ; $$TMP1
000134  001375                                  BNE     5$
000136  005302                          6$:     DEC     R2                                              ; $$TMP2
000140  000767                                  BR      4$
000142  104422                          7$:     TRAP    22
000144  105761  000000G                         TSTB    DCT(R1)                                         ;                       6220
000150  001027                                  BNE     10$
000152  005000                                  CLR     R0                                              ; Z                     6223
000154  016137  000004G 000000G          8$:     MOV     DCT+4(R1),TEMP1                                 ;                       6225
000162  060037  000000G                         ADD     R0,TEMP1                                        ; Z,*
000166  017737  000000G 000000G                MOV     @TEMP1,TEMP2                                    ;                       6226
000174  027727  000000G 000000G                CMP     @TEMP2,#CRN.LOW                                 ;                       6227
000202  001005                                  BNE     9$
```

K13


```
000204  012702  177777                    MOV     #-1,R2                  ; *,RC.REG                      6229
000210  010277  000000G                   MOV     R2,@RDRX.ADDR           ; RC.REG,*
000214  000413                            BR      12$
000216  062700  000004          9$:       ADD     #4,R0                   ; *,Z                           6223
000222  020027  000014                    CMP     R0,#14                  ; Z,*
000226  003752                            BLE     8$
000230  005304                  10$:      DEC     R4                      ; COUNT                         6216
000232  001330                            BNE     3$
000234  012700  177777          11$:      MOV     #-1,R0                  ; *,RC.REG                      6233
000240  010077  000000G                   MOV     R0,@RDRX.ADDR           ; RC.REG,*
000244  016300  000002G         12$:      MOV     CSI+2(R3),R0            ;                               6236
000250  042700  177000                    BIC     #177000,R0
000254  104436                            TRAP    36
000256  005205                  13$:      INC     R5                      ; CTLR                          6202
000260  000243                            .WORD CLV!CLC

000262  003661                            BLE     2$
000264  005726                            TST     (SP)+                   ;                               6178
000266  000207                            RTS     PC

; Routine Size:  92 words,      Routine Base:  $CODE$ + 4434
; Maximum stack depth per invocation:  10 words



                                          .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  004434'          L$CLEAN::
                                          JSR     PC,LCLEAN               ;                               6237
000004  104412                            TRAP    12
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 4724
; Maximum stack depth per invocation:  2 words
```

```
;    6240  1     %sbttl 'DROP UNIT SECTION'
;    6241  1
;    6242  1     !+
;    6243  1     ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    6244  1     ! TO NO LONGER BE TESTED.
;    6245  1     !-
;    6246  1
;    6247  2     BGNDU;
;    6248  2
;    6249  2     local
;    6250  2         UNIT : word,
;    6251  2         PRINT : byte initial (byte (FALSE));                          ! UNIT NUMBER
;    6252  2                                                                       ! NO PRINTING
;    6253  2     label
;    6254  2         SEARCH;
;    6255  2
;    6256  3     begin
;    6257  3
;    6258  3     register
;    6259  3         INPUT = 0;                                                    ! R0 = UNIT NO.
;    6260  3
;    6261  3     UNIT = .INPUT;                                                    ! GET UNIT NUMBER
;    6262  2     end;                                                              ! UNDECLARE R0
;    6263  2
;    6264  2     !ZZZif BIT_TST (SWP_FLAGS, SWF_TRC)
;    6265  2     !ZZZthen
;    6266  2     !ZZZ    PRINTF (DBM5, .UNIT);
;    6267  2
;    6268  2     SEARCH :                                                          ! SEARCH BLOCK
;    6269  3     begin
;    6270  3
;    6271  3     incr CTLR from 0 to (MAX_CTLR - 1) do                             ! FOR EACH CNTR
;    6272  3
;    6273  3         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    6274  3
;    6275  3             if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and    ! IF UNIT MATCHES
;    6276  4                (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    6277  3             then
;    6278  4                 begin
;    6279  4
;    6280  4                 if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or    ! IF UNIT ALIVE
;    6281  4                    (.DUR [.UNIT] eql DU_ONLINE) or
;    6282  5                    (.DUR [.UNIT] eql DU_PROTECT)
;    6283  4                 then
;    6284  5                     begin
;    6285  5                     PRINT = TRUE;                                     ! O.K. TO PRINT
;    6286  5
;    6287  5                     if (.CST [.CTLR, U_CNT] gtru 0) and
;    6288  6                        (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    6289  5                     then
;    6290  5                         CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1; ! DECREMENT COUNT
;    6291  5
;    6292  5                     if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;  6293  6                        (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;  6294  5                    then
;  6295  5                        EOP_FLAG = TRUE;                                    ! ALL UNITS OFFLINE
;  6296  5
;  6297  5                        CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;   ! MARK UNIT OFFLINE
;  6298  4                        end;                                                ! IF UNIT ALIVE
;  6299  4
;  6300  4                    leave SEARCH;                                           ! EXIT SEARCH BLOCK
;  6301  3                    end;                                                    ! IF UNIT FOUND
;  6302  3
;  6303  2            end;
;  6304  2
;  6305  2            if .PRINT or                                                    ! IF OK TO PRINT
;  6306  2               (.DUR [.UNIT] eql DU_CONF) or
;  6307  2               (.DUR [.UNIT] eql DU_INIT) or
;  6308  2               (.DUR [.UNIT] eql DU_ONLINE) or
;  6309  3               (.DUR [.UNIT] eql DU_PROTECT)
;  6310  2            then
;  6311  3                begin
;  6312  3                PRINTF (DU_MSG, .UNIT);
;  6313  3                PRINTF (.DU_RSN [.DUR [.UNIT]]);                            ! "UNIT XX DROPPED"
;  6314  2                end;                                                        ! REASON
;  6315  2
;  6316  1            ENDDU;


                                .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G   LDU:  JSR     R1,$SAVE5                       ;               6239
000004  024646              CMP     -(SP),-(SP)
000006  105066  000002          CLRB    2(SP)                           ; PRINT
000012  010001              MOV     R0,R1                           ; INPUT,UNIT    6261
000014  005005              CLR     R5                              ; CTLR          6271
000016  010546          1$: MOV     R5,-(SP)                        ; CTLR,*        6275
000020  012746  000053      MOV     #53,-(SP)
000024  004737  000000G     JSR     PC,BL$MUL
000030  010066  000004      MOV     R0,4(SP)
000034  012703  000003      MOV     #3,R3
000040  010300          2$: MOV     R3,R0                           ; *,OFFSET      6273
000042  066600  000004      ADD     4(SP),R0                        ; OFFSET,*      6275
000046  006300              ASL     R0
000050  012702  000000G     MOV     #CST,R2
000054  060002              ADD     R0,R2
000056  010104              MOV     R1,R4
000060  011200              MOV     (R2),R0                         ; UNIT,*
000062  000300              SWAB    R0
000064  042700  177760      BIC     #177760,R0
000070  020004              CMP     R0,R4
000072  001055              BNE     8$
000074  032712  040000      BIT     #40000,(R2)                     ;               6276
000100  001452              BEQ     8$
000102  005004              CLR     R4                              ;               6280
000104  032712  020000      BIT     #20000,(R2)
000110  001402              BEQ     3$
```

N13


```
000112  005204                          INC     R4
000114  000410                          BR      4$
000116  126127  000000G 000007   3$:     CMPB    DUR(R1),#7              ; *(UNIT),*             6281
000124  001404                          BEQ     4$
000126  126127  000000G 000011           CMPB    DUR(R1),#11             ; *(UNIT),*             6282
000134  001032                          BNE     7$
000136  112766  000001  000006   4$:     MOVB    #1,6(SP)                ; *,PRINT               6285
000144  010516                          MOV     R5,(SP)                 ; CTLR,*                6287
000146  012746  000126                   MOV     #126,-(SP)
000152  004737  000000G                  JSR     PC,BL$MUL
000156  005726                          TST     (SP)+
000160  062700  000004G                  ADD     #CST+4,R0
000164  105760  000001                   TSTB    1(R0)
000170  001404                          BEQ     5$
000172  006004                          ROR     R4                      ;                       6288
000174  105660  000001                   SBCB    1(R0)                   ;                       6290
000200  001006                          BNE     6$                      ;                       6292
000202  032712  020000           5$:     BIT     #20000,(R2)             ;                       6293
000206  001403                          BEQ     6$
000210  112737  000001  000000G          MOVB    #1,EOP.FLAG             ;                       6295
000216  042712  020000           6$:     BIC     #20000,(R2)             ;                       6297
000222  022626                   7$:     CMP     (SP)+,(SP)+             ;                       6278
000224  000411                          BR      9$
000226  062703  000012           8$:     ADD     #12,R3                  ; *,OFFSET              6273
000232  020327  000041                   CMP     R3,#41                  ; OFFSET,*
000236  003700                          BLE     2$
000240  022626                          CMP     (SP)+,(SP)+
000242  005205                          INC     R5                      ; CTLR                  6271
000244  000243                          .WORD CLV!CLC

000246  003663                          BLE     1$
000250  032766  000001  000002   9$:     BIT     #1,2(SP)                ; *,PRINT               6305
000256  001020                          BNE     10$
000260  126127  000000G 000001           CMPB    DUR(R1),#1              ; *(UNIT),*             6306
000266  001414                          BEQ     10$
000270  126127  000000G 000002           CMPB    DUR(R1),#2              ; *(UNIT),*             6307
000276  001410                          BEQ     10$
000300  126127  000000G 000007           CMPB    DUR(R1),#7              ; *(UNIT),*             6308
000306  001404                          BEQ     10$
000310  126127  000000G 000011           CMPB    DUR(R1),#11             ; *(UNIT),*             6309
000316  001024                          BNE     11$
000320  010146                  10$:     MOV     R1,-(SP)                ; UNIT,*                6312
000322  012746  000000G                  MOV     #DU.MSG,-(SP)
000326  012746  000002                   MOV     #2,-(SP)
000332  010600                          MOV     SP,R0                   ; SP,*
000334  104417                          TRAP    17
000336  116101  000000G                  MOVB    DUR(R1),R1              ; *(UNIT),*             6313
000342  042701  177400                   BIC     #177400,R1
000346  006301                          ASL     R1
000350  016116  000000G                  MOV     DU.RSN(R1),(SP)
000354  012746  000001                   MOV     #1,-(SP)
000360  010600                          MOV     SP,R0                   ; SP,*
000362  104417                          TRAP    17
```

B14


```
000364  062706  000010                          ADD     #10,SP                          ;                                       6311
000370  022626                          11$:    CMP     (SP)+,(SP)+                     ;                                       6239
000372  000207                                  RTS     PC

; Routine Size:  126 words,     Routine Base:  $CODE$ + 4734
; Maximum stack depth per invocation:  14 words


                                                .SBTTL  L$DU DROP UNIT SECTION
000000  004737  004734'                 L$DU::  JSR     PC,LDU                          ;                                       6314
000004  104453                                  TRAP    53
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5330
; Maximum stack depth per invocation:  2 words
```

```
;    6317  1     %sbttl 'ADD UNIT SECTION'
;    6318  1
;    6319  1     !+
;    6320  1     ! THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    6321  1     !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    6322  1     !  TO THE TEST CYCLE.
;    6323  1     !-
;    6324  1
;    6325  2     BGNAU;
;    6326  2
;    6327  2     local
;    6328  2         STINDX : word,
;    6329  2         ENDIDX : word;
;    6330  2
;    6331  2     register
;    6332  2         UNIT = 0;                                  ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;    6333  2
;    6334  3     if BIT_TST (SWP_FLAGS, SWF_CST)
;    6335  3     then
;    6336  3         begin                                      ! IF CLEAR STAT. TABLES TRUE....
;    6337  3         STINDX = .UNIT * TALLY_LEN;                ! ZERO OUT
;    6338  3         ENDIDX = .STINDX + TALLY_LEN - 1;          !    ADDED
;    6339  3
;    6340  3         incr COUNT from .STINDX to .ENDIDX do      !    UNIT'S
;    6341  3             TALLY [.COUNT] = 0;                    !    STATISTICS
;    6342  3
;    6343  2         end;
;    6344  2
;    6345  1     ENDAU;


                                .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G    LAU: JSR     R1,$SAVE2                  ;                          6316
000004  105737  000000G         TSTB    SWP.FLAGS                  ;                          6334
000010  100023                 BPL     3$
000012  010046                 MOV     R0,-(SP)                   ; UNIT,*                   6337
000014  012746  000033         MOV     #33,-(SP)
000020  004737  000000G         JSR     PC,BL$MUL
000024  010002                 MOV     R0,R2
000026  062702  000032         ADD     #32,R2                     ; STINDX,ENDIDX            6338
000032  010001                 MOV     R0,R1                      ; *,ENDIDX
000034  005301                 DEC     R1                         ; STINDX,COUNT             6340
000036  000404                 BR      2$                         ; COUNT
000040  010100             1$: MOV     R1,R0                      ; COUNT,*                  6341
000042  006300                 ASL     R0
000044  005060  000000G        CLR     TALLY(R0)
000050  005201             2$: INC     R1                         ; COUNT                    6340
000052  020102                 CMP     R1,R2                      ; COUNT,ENDIDX
000054  003771                 BLE     1$
000056  022626                 CMP     (SP)+,(SP)+                ;                          6336
000060  000207             3$: RTS     PC                         ;                          6316

; Routine Size:  25 words,    Routine Base:  $CODE$ + 5340
```

D14


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



                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  005340'          L$AU::  JSR     PC,LAU                  ;       6343
000004  104452                          TRAP    52
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 5422
; Maximum stack depth per invocation:  2 words
```

E14


```
;   6346  1     %sbttl 'NON-EXISTENT MEMORY TRAP HANDLER'
;   6347  1
;   6348  1     !+
;   6349  1     !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
;   6350  1     !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON-EXISTENT
;   6351  1     !       MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RDRX
;   6352  1     !       REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
;   6353  1     !-
;   6354  1
;   6355  2     BGNSRV (NEX_TRAP);
;   6356  2
;   6357  2     NEX = TRUE;                                 ! NEX TRAP OCCURRED
;   6358  2
;   6359  1     ENDSRV;


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

; Routine Size:  4 words,      Routine Base:  $CODE$ + 5432
; Maximum stack depth per invocation:  0 words
```

6357
6355

F14


```
;   6360  1     %sbttl 'HOST MEMORY PARITY TRAP HANDLER'                        !MMM
;   6361  1
;   6362  1     !+
;   6363  1     !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 114 FOR ALL HOST MEMORY PARITY
;   6364  1     !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO READ A MEMORY LOCATION WITH
;   6365  1     !       PARITY/UNCORRECTABLE ECC ERROR DETECTED BY THE MEMORY'S PARITY OR ECC LOGIC.
;   6366  1     !-
;   6367  1
;   6368  2     BGNSRV (PARITY);
;   6369  2
;   6370  2     builtin
;   6371  2           HALT;
;   6372  2
;   6373  2     PRINTF (DBM125, ..CSR_ADD);
;   6374  2
;   6375  2     .CSR_ADD = %o'40000';
;   6376  2
;   6377  2     PRINTF (DBM126, ..CSR_ADD);
;   6378  2
;   6379  2     HALT ();
;   6380  2
;   6381  1     ENDSRV;


                                        .SBTTL  PARITY HOST MEMORY PARITY TRAP HANDLER
000000  010046                  PARITY::MOV     R0,-(SP)                        ;                                       6368
000002  017746  000000G                 MOV     @CSR.ADD,-(SP)                  ;                                       6373
000006  012746  000000G                 MOV     #DBM125,-(SP)
000012  012746  000002                  MOV     #2,-(SP)
000016  010600                          MOV     SP,R0                           ; SP,*
000020  104417                          TRAP    17
000022  012777  040000  000000G         MOV     #40000,@CSR.ADD                 ;                                       6375
000030  012716  040000                  MOV     #40000,(SP)                     ;                                       6377
000034  012746  000000G                 MOV     #DBM126,-(SP)
000040  012746  000002                  MOV     #2,-(SP)
000044  010600                          MOV     SP,R0                           ; SP,*
000046  104417                          TRAP    17
000050  000000                          HALT                                    ;                                       6379
000052  062706  000012                  ADD     #12,SP                          ;                                       6368
000056  012600                          MOV     (SP)+,R0
000060  000002                          RTI

; Routine Size:  25 words,      Routine Base:  $CODE$ + 5442
; Maximum stack depth per invocation:  8 words

;   6382  1
```

```
;   6383  1     %sbttl 'TIME OF DAY'
;   6384  1
;   6385  1     !+
;   6386  1     !       THIS INTERRUPT SERVICE ROUTINE KEEPS TRACK OF THE TIME-OF-DAY
;   6387  1     !-
;   6388  1
;   6389  2     BGNSRV (TIME);
;   6390  2
;   6391  2     CLK_TICKS = .CLK_TICKS + 1;             ! INCREMENT CLOCK-TICKS
;   6392  2
;   6393  2     if .CLK_TICKS gequ 3600
;   6394  2     then
;   6395  3         begin
;   6396  3         MINUTES = .MINUTES + 1;             ! UPDATE MINUTE COUNT
;   6397  3         CLK_TICKS = 0;
;   6398  2         end;
;   6399  2
;   6400  2     if .MINUTES gequ 60
;   6401  2     then
;   6402  3         begin
;   6403  3         HOURS = .HOURS + 1;                 ! UPDATE HOUR COUNT
;   6404  3         MINUTES = 0;
;   6405  2         end;
;   6406  2
;   6407  2     if .HOURS gequ 24
;   6408  2     then
;   6409  2         HOURS = 0;                          ! RATIONALIZE HOURS
;   6410  2
;   6411  1     ENDSRV;


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

; Routine Size:  25 words,      Routine Base:  $CODE$ + 5524
; Maximum stack depth per invocation:  0 words
```

```
;   6412  1     %sbttl 'GLOBAL ROUTINES'
;   6413  1
;   6414  1     global routine SET_CPAR (CTLR) : novalue =
;   6415  1
;   6416  1     !+
;   6417  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;   6418  1     !       FOR THE GIVEN CONTROLLER NUMBER.
;   6419  1     !
;   6420  1     !       INPUTS:
;   6421  1     !               CTLR - CONTROLLER NUMBER
;   6422  1     !
;   6423  1     !       IMPLICIT OUTPUTS:
;   6424  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;   6425  1     !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;   6426  1     !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;   6427  1     !               RDRX_ADDR - ADDRESS OF CONTROLLER'S IP REGISTER
;   6428  1     !-
;   6429  1
;   6430  2         begin
;   6431  2         CCTLR = .CTLR;                          ! SET CURRENT CONTROLLER NUMBER
;   6432  2         CST_ADDR = CST + (.CTLR * CST_LEN * 2); ! CALCULATE ADDRESS OF CONTROLLER'S CST
;   6433  2         DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2); ! CALCULATE ADDRESS OF CONTROLLER'S DCT
;   6434  2         RDRX_ADDR = .CST_ADDR [IP_ADDR];        ! GET CONTROLLER'S DEVICE ADDRESS
;   6435  1         end;


                                        .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                 SET.CPAR::
                                        MOV     R1,-(SP)                        ;
000002  016601  000004                  MOV     4(SP),R1                        ; CTLR,*                6414
000006  010137  000000G                 MOV     R1,CCTLR                                                6431
000012  010146                          MOV     R1,-(SP)                        ;                       6432
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  062700  000000G                 ADD     #CST,R0
000030  010037  000000G                 MOV     R0,CST.ADDR
000034  010116                          MOV     R1,(SP)                         ;                       6433
000036  012746  000022                  MOV     #22,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  062700  000000G                 ADD     #DCT,R0
000052  010037  000000G                 MOV     R0,DCT.ADDR
000056  017737  000000G 000000G         MOV     @CST.ADDR,RDRX.ADDR             ;                       6434
000064  062706  000006                  ADD     #6,SP                           ;                       6430
000070  012601                          MOV     (SP)+,R1                        ;                       6414
000072  000207                          RTS     PC

; Routine Size:  30 words,    Routine Base:  $CODE$ + 5606
; Maximum stack depth per invocation:  5 words
```

```
;    6436  1     global routine SET_UPAR (OFFSET) : novalue =
;    6437  1     !+
;    6438  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
;    6439  1     !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
;    6440  1     !
;    6441  1     !       INPUTS:
;    6442  1     !               OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
;    6443  1     !                        DESCRIBES A UNIT
;    6444  1     !
;    6445  1     !       IMPLICIT INPUTS:
;    6446  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    6447  1     !
;    6448  1     !       IMPLICIT OUTPUTS:
;    6449  1     !               CUOFF - CURRENT UNIT'S CST OFFSET
;    6450  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    6451  1     !               L$LUN - CURRENT UNIT NUMBER (DRS UNIT NUMBER)
;    6452  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    6453  1     !-
;    6454  2         begin
;    6455  2         CUOFF = .OFFSET;
;    6456  2         CDISK = .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM];
;    6457  2         L$LUN = .CST_ADDR [.OFFSET + OF_DATA, D_UNIT];
;    6458  2         T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
;    6459  1         end;


                                        .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                 SET.UPAR::
                                        MOV     R1,-(SP)                ;                               6436
000002  016637  000004  000000G        MOV     4(SP),CUOFF             ; OFFSET,*                      6455
000010  016600  000004                 MOV     4(SP),R0                ; CUOFF,*                       6456
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                 ;                               6457
000036  000301                         SWAB    R1
000040  042701  177760                 BIC     #177760,R1
000044  010137  000000G                MOV     R1,L$LUN
000050  010146                         MOV     R1,-(SP)                ; L$LUN,*                       6458
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)+             ;                               6454
000074  012601                         MOV     (SP)+,R1                ;                               6436
000076  000207                         RTS     PC

; Routine Size:  32 words,    Routine Base:  $CODE$ + 5702
; Maximum stack depth per invocation:  4 words
```

```
;  6460  1
;  6461  1     global routine GET_PKT (CTLR) =
;  6462  1
;  6463  1     !+
;  6464  1     !       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
;  6465  1     !       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;  6466  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
;  6467  1     !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;  6468  1     !
;  6469  1     !       INPUTS:
;  6470  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;  6471  1     !-
;  6472  1
;  6473  2         begin
;  6474  2
;  6475  2         local
;  6476  2             index : signed word initial (-1),
;  6477  2             RING_ADDR : word,
;  6478  2             PACKET_OWNED : byte,
;  6479  2             NEXT_PACKET : byte;
;  6480  2
;  6481  2
;  6482  2         NEXT_PACKET = .NEXT_PKT_USE;                    ! NEXT PACKET TO TRY
;  6483  2
;  6484  2         incr COUNT from 0 to (PKT_CNT - 1) do           ! FOR EACH ENTRY IN ALLOCATION TABLE
;  6485  3             begin
;  6486  3             PACKET_OWNED = FALSE;
;  6487  3
;  6488  3             if .PKT_USE [.NEXT_PACKET] lss 0            ! IF ENTRY INDICATES FREE PACKET
;  6489  3
;  6490  3             then
;  6491  4                 begin
;  6492  4                 RING_ADDR = .DCT_ADDR [RR_BEG];         ! FIRST RESPONSE PACKET'S ADDRESS
;  6493  4
;  6494  4                 incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS
;  6495  4
;  6496  4                     if (..RING_ADDR eql MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
;  6497  4 !MMM                    (((.RING_ADDR + 1) and ED_OWN) eql ED_OWN)
;  6498  5                         (((.(.RING_ADDR + 2)) and ED_OWN) eql ED_OWN)                    !MMM
;  6499  5
;  6500  4                     then
;  6501  5                         begin                           ! CHECK ADDRESS AND OWNERSHIP
;  6502  5                         PACKET_OWNED = TRUE;            ! PACKET OWNED BY CONTROLLER
;  6503  5                         exitloop;
;  6504  5                         end
;  6505  4                     else
;  6506  4                         RING_ADDR = .RING_ADDR + 4;     ! ADDRESS OF NEXT PACKET IN RING
;  6507  4
;  6508  4                 if not .PACKET_OWNED                    ! IF NOT ALREADY USED
;  6509  4
;  6510  4                 then
;  6511  5                     begin
;  6512  5                     PKT_USE [.NEXT_PACKET] = .CTLR;     ! ALLOCATE PACKET TO CONTROLLER
```

K14


```
;    6513  5                    index = .NEXT_PACKET;
;    6514  5
;    6515  5                    incr J from 2 to (PKT_LEN - 1) do       ! ZERO OUT PACKET
;    6516  5                        MSCP_PKT [.NEXT_PACKET, .J, 0, 16, 0] = 0;
;    6517  5
;    6518  5                    exitloop;                               ! DONE
;    6519  5
;    6520  4                    end;
;    6521  4
;    6522  3                end;
;    6523  3
;    6524  3            NEXT_PACKET = .NEXT_PACKET + 1;                 ! TRY NEXT PACKET IN RING
;    6525  3
;    6526  3            if .NEXT_PACKET gequ PKT_CNT
;    6527  3
;    6528  3            then
;    6529  3                NEXT_PACKET = 0;                            ! IF BEYOND ALL PACKETS, START AT THE TOP
;    6530  3
;    6531  3            end;
;    6532  2
;    6533  2            if (.index geq 0) and                           ! IF PACKET FOUND
;    6534  3               (.PKT_USE [.index] geq 0)
;    6535  2            then
;    6536  2
;    6537  3                begin
;    6538  3                MSCP_PKT [.index, MSGLEN] = SZ_GEN;         ! PACKET SIZE - ONLY ONLINE AND SCC CHANGE IT
;    6539  3                MSCP_PKT [.index, CREDITS] = 1;             ! CREDIT SIZE
;    6540  3                NEXT_PKT_USE = .NEXT_PACKET + 1;            ! NEXT PACKET TO ALLOCATE
;    6541  3
;    6542  3                if.NEXT_PKT_USE gequ PKT_CNT
;    6543  3                then
;    6544  3                    NEXT_PKT_USE = 0;                       ! CYCLE BACK TO BEGINNING IF AT END
;    6545  3
;    6546  2                end;
;    6547  2
;    6548  2            return .index;
;    6549  2
;    6550  1            end;


                                .SBTTL  GET.PKT GLOBAL ROUTINES
000000  004137  000000G    GET.PKT::
                            JSR     R1,$SAVE5                       ;                       6461
000004  162706  000006       SUB     #6,SP
000010  012704  177777       MOV     #-1,R4                          ; *,INDEX               6473
000014  113766  000000G 000004  MOVB  NEXT.PKT.USE,4(SP)             ; *,NEXT.PACKET         6482
000022  012766  000014  000002  MOV   #14,2(SP)                      ; *,COUNT               6484
000030  105016             1$:  CLRB  (SP)                          ; PACKET.OWNED          6486
000032  005001                  CLR   R1                            ;                       6488
000034  156601  000004          BISB  4(SP),R1                      ; NEXT.PACKET,*
000040  105761  000000G        TSTB  PKT.USE(R1)
000044  002071                  BGE   7$
000046  013700  000000G        MOV   DCT.ADDR,R0                   ;                       6492
```

L14


```
000052  016005  000004                  MOV     4(R0),R5                ; *,RING.ADDR                   6496
000056  010146                          MOV     R1,-(SP)                ;
000060  012746  000106                  MOV     #106,-(SP)
000064  004737  000000G                 JSR     PC,BL$MUL
000070  012702  000010                  MOV     #10,R2                  ; *,I                           6494
000074  021560  000000G         2$:     CMP     (R5),MSCP.PKT(R0)       ; RING.ADDR,*                   6496
000100  001013                          BNE     3$
000102  016503  000002                  MOV     2(R5),R3                ; *(RING.ADDR),*                6498
000106  042703  077777                  BIC     #77777,R3
000112  020327  100000                  CMP     R3,#-100000
000116  001004                          BNE     3$
000120  112766  000001  000004          MOVB    #1,4(SP)                ; *,PACKET.OWNED                6502
000126  000404                          BR      4$                      ;                               6501
000130  062705  000004          3$:     ADD     #4,R5                   ; *,RING.ADDR                   6506
000134  005302                          DEC     R2                      ; I                             6494
000136  001356                          BNE     2$
000140  032766  000001  000004  4$:     BIT     #1,4(SP)                ; *,PACKET.OWNED                6508
000146  001027                          BNE     6$
000150  116661  000030  000000G         MOVB    30(SP),PKT.USE(R1)      ; CTLR,*                        6512
000156  010104                          MOV     R1,R4                   ; *,INDEX                       6513
000160  010116                          MOV     R1,(SP)                 ;                               6516
000162  012746  000043                  MOV     #43,-(SP)
000166  004737  000000G                 JSR     PC,BL$MUL
000172  005726                          TST     (SP)+
000174  012702  000002                  MOV     #2,R2                   ; *,J                           6515
000200  010003          5$:     MOV     R0,R3                   ;                               6516
000202  060203                  ADD     R2,R3                   ; J,*
000204  006303                  ASL     R3
000206  005063  000000G         CLR     MSCP.PKT(R3)
000212  005202                  INC     R2                      ; J                             6515
000214  020227  000042          CMP     R2,#42                  ; J,*
000220  003767                  BLE     5$
000222  022626                  CMP     (SP)+,(SP)+             ;                               6511
000224  000414                  BR      9$
000226  022626          6$:     CMP     (SP)+,(SP)+             ;                               6491
000230  105266  000004  7$:     INCB    4(SP)                   ; NEXT.PACKET                   6524
000234  126627  000004  000014  CMPB    4(SP),#14               ; NEXT.PACKET,*                 6526
000242  103402                  BLO     8$
000244  105066  000004                  CLRB    4(SP)                   ; NEXT.PACKET                   6529
000250  005366  000002          8$:     DEC     2(SP)                   ; COUNT                         6484
000254  001265                          BNE     1$
000256  005704          9$:     TST     R4                      ; INDEX                         6533
000260  002435                  BLT     11$
000262  105764  000000G         TSTB    PKT.USE(R4)             ; *(INDEX)                      6534
000266  002432                  BLT     11$
000270  010446                  MOV     R4,-(SP)                ; INDEX,*                       6538
000272  012746  000106                  MOV     #106,-(SP)
000276  004737  000000G                 JSR     PC,BL$MUL
000302  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000310  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)     ;                               6539
000316  152760  000001  000010G         BISB    #1,MSCP.PKT+10(R0)
000324  005000                          CLR     R0                      ;                               6540
000326  156600  000010                  BISB    10(SP),R0               ; NEXT.PACKET,*
```

M14


```
000332  005200                          INC     R0
000334  110037  000000G                 MOVB    R0,NEXT.PKT.USE
000340  120027  000014                  CMPB    R0,#14                  ; NEXT.PKT.USE,*                6542
000344  103402                          BLO     10$
000346  105037  000000G                 CLRB    NEXT.PKT.USE            ;                               6544
000352  022626                  10$:    CMP     (SP)+,(SP)+             ;                               6537
000354  010400                  11$:    MOV     R4,R0                   ; INDEX,*                       6473
000356  062706  000006                  ADD     #6,SP                   ;                               6461
000362  000207                          RTS     PC

; Routine Size:  122 words,    Routine Base:  $CODE$ + 6002
; Maximum stack depth per invocation:  13 words


;   6551  1
;   6552  1
```

```
;    6553  1
;    6554  1
;    6555  1     global routine PUT_PKT (index) : novalue =
;    6556  1
;    6557  1     !+
;    6558  1     !       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    6559  1     !       ROUTINE.
;    6560  1     !-
;    6561  1
;    6562  1
;    6563  2         begin
;    6564  2
;    6565  2
;    6566  2         local
;    6567  2             RING_ADDR : word,
;    6568  2             OWNER : word;
;    6569  2
;    6570  2         RING_ADDR = .DCT_ADDR [RR_BEG];                                 ! ADDRESS IN FIRST RESPONSE RING
;    6571  2
;    6572  2         incr COUNT from 1 to (RRING_LEN + CRING_LEN) do                 ! FOR EACH ADDRESS IN THE RINGS
;    6573  3             begin
;    6574  3
;    6575  3             if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR              ! IF ADDRESS MATCHES
;    6576  3
;    6577  3             then
;    6578  4                 begin
;    6579  4                 OWNER = .RING_ADDR + 2;                                 ! ADDRESS OF OWNERSHIP WORD
;    6580  4                 .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;   ! GIVE OWNERSHIP TO HOST
;    6581  3                 end;
;    6582  3
;    6583  3
;    6584  3             RING_ADDR = .RING_ADDR + 4;                                 ! LOOK AT NEXT PACKET ADDRESS IN RING
;    6585  2             end;
;    6586  2
;    6587  2         PKT_USE [.index] = -1;
;    6588  2
;    6589  2
;    6590  1         end;


                                .SBTTL  PUT.PKT GLOBAL ROUTINES
000000  004137  000000G   PUT.PKT::
                            JSR     R1,$SAVE4
000004  013700  000000G            MOV     DCT.ADDR,R0                             ;                                   6555
000010  016001  000004              MOV     4(R0),R1                                ;                                   6570
000014  016602  000014              MOV     14(SP),R2                               ; *,RING.ADDR
000020  010246                      MOV     R2,-(SP)                                ; INDEX,*                           6575
000022  012746  000106              MOV     #106,-(SP)
000026  004737  000000G             JSR     PC,BL$MUL
000032  012704  000010              MOV     #10,R4                                  ; *,COUNT                           6572
000036  026011  000000G     1$:     CMP     MSCP.PKT(R0),(R1)                       ; *,RING.ADDR                       6575
000042  001005                      BNE     2$
000044  012703  000002              MOV     #2,R3                                   ; *,OWNER                           6579
```

B15


```
000050  060103                          ADD     R1,R3                           ; RING.ADDR,OWNER
000052  042713  140000                  BIC     #140000,(R3)                    ; *,OWNER                       6580
000056  062701  000004          2$:     ADD     #4,R1                           ; *,RING.ADDR                   6584
000062  005304                          DEC     R4                              ; COUNT                         6572
000064  001364                          BNE     1$
000066  112762  000377  000000G         MOVB    #377,PKT.USE(R2)                ;                               6588
000074  022626                          CMP     (SP)+,(SP)+                     ;                               6563
000076  000207                          RTS     PC                              ;                               6555

; Routine Size:  32 words,    Routine Base:  $CODE$ + 6366
; Maximum stack depth per invocation:  8 words

;   6591  1
;   6592  1
```

```
;   6593  1     routine PUTA_PKT (CTLR) : novalue =
;   6594  1
;   6595  1     !+
;   6596  1     !       THIS ROUTINE DEALLOCATES ALL MSCP PACKETS WHICH HAVE BEEN ALLOCATED
;   6597  1     !       TO A PARTICULAR CONTROLLER.
;   6598  1     !
;   6599  1     !       INPUTS:
;   6600  1     !               CTLR - CONTROLLER NUMBER
;   6601  1     !-
;   6602  1
;   6603  1         incr COUNT from 0 to (PKT_CNT - 1) do       ! FOR EACH ENTRY IN ALLOCATION TABLE
;   6604  1
;   6605  1             if .PKT_USE [.COUNT] eql .CTLR          ! IF PACKET IS ALLOCATED TO GIVEN CONTROLLER
;   6606  1             then
;   6607  1                 PKT_USE [.COUNT] = -1;              ! DEALLOCATE IT


                                        .SBTTL  PUTA.PKT GLOBAL ROUTINES
000000  010146                 PUTA.PKT:
                                        MOV     R1,-(SP)                        ;                       6593
000002  005000                         CLR     R0                              ; COUNT                 6603
000004  116001  000000G        1$:     MOVB    PKT.USE(R0),R1                  ; *(COUNT),*            6605
000010  020166  000004                  CMP     R1,4(SP)                        ; *,CTLR
000014  001003                          BNE     2$
000016  112760  000377  000000G        MOVB    #377,PKT.USE(R0)                ; *,*(COUNT)            6607
000024  005200                 2$:     INC     R0                              ; COUNT                 6603
000026  020027  000013                  CMP     R0,#13                          ; COUNT,*
000032  003764                          BLE     1$
000034  012601                          MOV     (SP)+,R1                        ;                       6593
000036  000207                          RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 6466
; Maximum stack depth per invocation:  2 words
```

```
;    6608  1     global routine GET_RETPKT (CTLR) =
;    6609  1
;    6610  1     !+
;    6611  1     !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;    6612  1     !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    6613  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;    6614  1     !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    6615  1     !
;    6616  1     !       INPUTS:
;    6617  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    6618  1     !-
;    6619  1
;    6620  2         begin
;    6621  2
;    6622  2         local
;    6623  2             index : signed word initial (-1);       ! ASSUME NONE AVAILABLE
;    6624  2
;    6625  2         incr COUNT from 0 to (RP_CNT - 1) do        ! FOR EACH ENTRY IN TABLE
;    6626  2
;    6627  2             if .RP_USE [.COUNT] lss 0               ! IF FREE RETPKT IS FOUND
;    6628  2             then
;    6629  3                 begin
;    6630  3                 RP_USE [.COUNT] = .CTLR;            ! ALLOCATE RETURN PACKET TO CONTROLLER
;    6631  3                 index = .COUNT;
;    6632  3
;    6633  3                 incr J from 0 to (RP_LEN - 1) do    ! ZERO OUT RETPKT
;    6634  3                     RETPKT [.COUNT, .J, 0, 16, 0] = 0;
;    6635  3
;    6636  3                 exitloop;                           ! DONE
;    6637  2                 end;
;    6638  2
;    6639  2         return .index;                              ! RETURN PACKET INDEX (OR -1) TO CALLER
;    6640  1         end;


                                .SBTTL  GET.RETPKT GLOBAL ROUTINES
000000  004137  000000G   GET.RETPKT::
                            JSR     R1,$SAVE4                   ;                               6608
000004  012703  177777      MOV     #-1,R3                      ; *,INDEX                       6620
000010  005001              CLR     R1                          ; COUNT                         6625
000012  105761  000000G 1$: TSTB    RP.USE(R1)                  ; *(COUNT)                      6627
000016  002025              BGE     3$
000020  116661  000014  000000G    MOVB    14(SP),RP.USE(R1)   ; CTLR,*(COUNT)                 6630
000026  010103              MOV     R1,R3                       ; COUNT,INDEX                   6631
000030  010146              MOV     R1,-(SP)                    ; COUNT,*                       6634
000032  012746  000026      MOV     #26,-(SP)
000036  004737  000000G      JSR     PC,BL$MUL
000042  022626              CMP     (SP)+,(SP)+
000044  005002              CLR     R2                          ; J                             6633
000046  010004          2$: MOV     R0,R4                       ;                               6634
000050  060004              ADD     R2,R4                       ; J,*
000052  006304              ASL     R4
000054  005064  000000G      CLR     RETPKT(R4)
```

```
000060  005202                          INC     R2                      ; J                             6633
000062  020227  000025                  CMP     R2,#25                  ; J,*
000066  003767                          BLE     2$
000070  000404                          BR      4$
000072  005201                  3$:     INC     R1                      ;                               6629
000074  020127  000007                  CMP     R1,#7                   ; COUNT                         6625
000100  003744                          BLE     1$                      ; COUNT,*
000102  010300                  4$:     MOV     R3,R0                   ; INDEX,*                       6620
000104  000207                          RTS     PC                      ;                               6608

; Routine Size:  35 words,      Routine Base:  $CODE$ + 6526
; Maximum stack depth per invocation:  8 words
```

F15


```
;   6641  1     global routine PUT_RETPKT (index) : novalue =
;   6642  1
;   6643  1     !+
;   6644  1     !       THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;   6645  1     !       ROUTINE.
;   6646  1     !-
;   6647  1
;   6648  1         RP_USE [.index] = -1;


                                        .SBTTL  PUT.RETPKT GLOBAL ROUTINES
000000  016600  000002                 PUT.RETPKT::
                                        MOV     2(SP),R0                        ; INDEX,*                       6648
000004  112760  000377  000000G        MOVB    #377,RP.USE(R0)
000012  000207                          RTS     PC                              ;                               6641

; Routine Size:  6 words,      Routine Base:  $CODE$ + 6634
; Maximum stack depth per invocation:  0 words
```

```
1
1
1     global routine GET_IO_BUFF (ADDR) : novalue =
1
1
1     !+
1     !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
1     !       POOL.
1     !
1     !       INPUTS:
1     !               ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
1     !
1     !       IMPLICIT INPUTS:
1     !               CCTLR - CURRENT CONTROLLER NUMBER
1     !
1     !       OUTPUTS:
1     !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
1     !               WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A ZERO IS RETURNED
1     !               AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
1     !-
1
1
1
2         begin
2         .ADDR = 0;                                                  ! ASSUME FAILURE
2
2         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
2
2             if .BUFF_OWN [.COUNT] lss 0                             ! IF BUFFER IS FREE
2
2             then
2
3                 begin
3                 BUFF_OWN [.COUNT] = .CCTLR;                         ! ALLOCATE BUFFER TO CONTROLLER
3                 .ADDR = .BUFF_ADDR [.COUNT];                        ! RETURN BUFFER DESCRIPTOR
3                 exitloop;                                           ! DONE
2                 end;
2
2
1         end;                                                        ! ROUTINE GET_IO_BUFF


                                        .SBTTL  GET.IO.BUFF GLOBAL ROUTINES
000000  010146                  GET.IO.BUFF::
                                        MOV     R1,-(SP)                        ;                                       6651
000002  005076  000004                  CLR     @4(SP)                          ; ADDR                                  6673
000006  005001                          CLR     R1                              ; COUNT                                 6675
000010  105761  000000G         1$:     TSTB    BUFF.OWN(R1)                    ; *(COUNT)                              6677
000014  002011                          BGE     2$
000016  113761  000000G 000000G         MOVB    CCTLR,BUFF.OWN(R1)              ; *,*(COUNT)                            6682
000024  010100                          MOV     R1,R0                           ; COUNT,*                               6683
000026  006300                          ASL     R0
000030  016076  000000G 000004          MOV     BUFF.ADDR(R0),@4(SP)            ; *,ADDR
```

H15


```
000036  000404                          BR      3$                      ;                       6681
000040  005201                  2$:     INC     R1                      ; COUNT                 6675
000042  020127  000007                  CMP     R1,#7                   ; COUNT,*
000046  003760                          BLE     1$
000050  012601                  3$:     MOV     (SP)+,R1                ;                       6651
000052  000207                          RTS     PC

; Routine Size:  22 words,    Routine Base:  $CODE$ + 6650
; Maximum stack depth per invocation:  2 words


;   6689  1
;   6690  1
```

```
;   6691  1     global routine PUT_IO_BUFF (ADDR) : novalue =
;   6692  1
;   6693  1     !+
;   6694  1     !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;   6695  1     !       TO THE BUFFER POOL.
;   6696  1     !
;   6697  1     !       INPUTS:
;   6698  1     !               ADDR - ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
;   6699  1     !                       DEALLOCATED
;   6700  1     !-
;   6701  1
;   6702  1         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;   6703  1
;   6704  1             if .BUFF_ADDR [.COUNT] eqla ..ADDR                      ! IF THIS IS THE BUFFER'S ENTRY
;   6705  1             then
;   6706  2                 begin
;   6707  2                 BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE BUFFER
;   6708  2                 exitloop;                                           ! DONE
;   6709  1                 end;


                                        .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  010146                 PUT.IO.BUFF::
                                        MOV     R1,-(SP)                    ;                                   6691
000002  005001                          CLR     R1                          ; COUNT                             6702
000004  010100                  1$:     MOV     R1,R0                       ; COUNT,*                           6704
000006  006300                          ASL     R0
000010  026076  000000G 000004          CMP     BUFF.ADDR(R0),@4(SP)        ; *,ADDR
000016  001004                          BNE     2$
000020  112761  000377  000000G         MOVB    #377,BUFF.OWN(R1)           ; *,*(COUNT)                        6707
000026  000404                          BR      3$                          ;                                   6706
000030  005201                  2$:     INC     R1                          ; COUNT                             6702
000032  020127  000007                  CMP     R1,#7                       ; COUNT,*
000036  003762                          BLE     1$
000040  012601                  3$:     MOV     (SP)+,R1                    ;                                   6691
000042  000207                          RTS     PC

; Routine Size:  18 words,      Routine Base:  $CODE$ + 6724
; Maximum stack depth per invocation:  2 words
```

J15


```
;   6710  1     global routine PUTA_BUFF : novalue =
;   6711  1
;   6712  1     !+
;   6713  1     !       THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
;   6714  1     !       THE CURRENT CONTROLLER (CCTLR).
;   6715  1     !-
;   6716  1
;   6717  1         incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;   6718  1
;   6719  1             if .BUFF_OWN [.COUNT] eql .CCTLR                        ! IF THIS BUFFER ALLOCATED TO CURRENT CONTROLLER
;   6720  1             then
;   6721  1                 BUFF_OWN [.COUNT] = -1;                             ! DEALLOCATE IT


                                        .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                         PUTA.BUFF::
                                        MOV     R1,-(SP)                        ;                                       6710
000002  005000                         CLR     R0                              ; COUNT                                 6717
000004  116001  000000G                1$:     MOVB    BUFF.OWN(R0),R1         ; *(COUNT),*                            6719
000010  020137  000000G                        CMP     R1,CCTLR
000014  001003                         BNE     2$
000016  112760  000377  000000G        MOVB    #377,BUFF.OWN(R0)               ; *,*(COUNT)                            6721
000024  005200                         2$:     INC     R0                      ; COUNT                                 6717
000026  020027  000007                 CMP     R0,#7                           ; COUNT,*
000032  003764                         BLE     1$
000034  012601                         MOV     (SP)+,R1                        ;                                       6710
000036  000207                         RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 6770
; Maximum stack depth per invocation:  2 words
```

K15


```
;   6722  1     global routine OUT_IODQ =
;   6723  1
;   6724  1     !+
;   6725  1     !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;   6726  1     !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;   6727  1     !       QUEUE IS ALSO UPDATED.
;   6728  1     !
;   6729  1     !       INPUTS:
;   6730  1     !               NONE
;   6731  1     !
;   6732  1     !       OUTPUTS:
;   6733  1     !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;   6734  1     !-
;   6735  1
;   6736  2         begin
;   6737  2
;   6738  2         local
;   6739  2             index : word;
;   6740  2
;   6741  2         index = .IODQ [.IODQ_OUT];              ! GET NEXT RETPKT INDEX
;   6742  2         IODQ_OUT = .IODQ_OUT + 1;               ! ADVANCE "OUT" POINTER
;   6743  2
;   6744  2         if .IODQ_OUT gequ IODQ_LEN              ! IF BEYOND END OF QUEUE
;   6745  2         then
;   6746  2             IODQ_OUT = 0;                       ! SET POINTER TO BEGINNING OF QUEUE
;   6747  2
;   6748  2         return .index;                          ! RETURN INDEX TO CALLER
;   6749  1         end;


                                        .SBTTL  OUT.IODQ GLOBAL ROUTINES
000000  013700  000000G          OUT.IODQ::
                                        MOV     IODQ.OUT,R0                     ;                       6741
000004  116000  000000G                 MOVB    IODQ(R0),R0                     ; *,INDEX
000010  042700  177400                  BIC     #177400,R0                      ; *,INDEX
000014  005237  000000G                 INC     IODQ.OUT                        ;                       6742
000020  023727  000000G 000010          CMP     IODQ.OUT,#10                    ;                       6744
000026  103402                          BLO     1$
000030  005037  000000G                 CLR     IODQ.OUT                        ;                       6746
000034  000207                  1$:     RTS     PC                              ;                       6722

; Routine Size:  15 words,      Routine Base:  $CODE$ + 7030
; Maximum stack depth per invocation:  0 words
```

```
;   6750  1     global routine IN_IODQ (index) : novalue =
;   6751  1
;   6752  1     !+
;   6753  1     !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;   6754  1     !       UPDATES THE IODQ_IN POINTER.
;   6755  1     !-
;   6756  1
;   6757  1         if ((.IODQ_IN + 1) eql .IODQ_OUT) or
;   6758  2            (.IODQ_IN - (IODQ_LEN - 1) eql .IODQ_OUT)
;   6759  2         then
;   6760  1             return
;   6761  1         else
;   6762  2             begin
;   6763  2             IODQ [.IODQ_IN] = .index;              ! LOAD INDEX INTO QUEUE
;   6764  2             IODQ_IN = .IODQ_IN + 1;                ! ADVANCE "IN" POINTER
;   6765  2
;   6766  2             if .IODQ_IN gequ IODQ_LEN              ! IF BEYOND END OF QUEUE
;   6767  2             then
;   6768  2                 IODQ_IN = 0;                       ! CYCLE BACK TO BEGINNING OF QUEUE
;   6769  2
;   6770  1             end;                                   ! IF IODQ IS NOT FULL


                                .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                IN.IODQ::
                                MOV     R1,-(SP)                ;                       6750
000002  013701  000000G         MOV     IODQ.IN,R1              ;                       6757
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
000022  162700  000007          SUB     #7,R0                   ;                       6758
000026  020037  000000G         CMP     R0,IODQ.OUT
000032  001413                  BEQ     1$
000034  116661  000004  000000G MOVB    4(SP),IODQ(R1)          ;                       6760
                                                                ; INDEX,*               6763
000042  005237  000000G         INC     IODQ.IN                 ;                       6764
000046  023727  000000G 000010  CMP     IODQ.IN,#10             ;                       6766
000054  103402                  BLO     1$
000056  005037  000000G         CLR     IODQ.IN                 ;                       6768
000062  012601                1$:       MOV     (SP)+,R1        ;                       6750
000064  000207                  RTS     PC

; Routine Size:  27 words,      Routine Base:  $CODE$ + 7066
; Maximum stack depth per invocation:  2 words
```

```
;   6771  1
;   6772  1
;   6773  1     global routine DROP_CTLR (CTLR, REASON) : novalue =
;   6774  1
;   6775  1
;   6776  1
;   6777  1     !+
;   6778  1     !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;   6779  1     !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;   6780  1     !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;   6781  1     !       SECTION.
;   6782  1     !-
;   6783  1
;   6784  1
;   6785  2         begin
;   6786  2
;   6787  2         local
;   6788  2             UNIT;
;   6789  2
;   6790  2         incr N from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do       ! FOR EACH UNIT
;   6791  2
;   6792  2             if .CST [.CTLR, .N + OF_DATA, D_PRES] eql PRESENT                                       ! IF CONFIGURED
;   6793  2             then
;   6794  3                 begin
;   6795  3                 UNIT = .CST [.CTLR, .N + OF_DATA, D_UNIT];                                          ! DRS UNIT NUMBER
;   6796  3                 DUR [.UNIT] = .REASON;                                                              ! DROP REASON
;   6797  3                 DODU (.UNIT);                                                                       ! DROP UNIT
;   6798  2                 end;
;   6799  2
;   6800  1         end;


                        .SBTTL  DROP.CTLR GLOBAL ROUTINES
000000  004137  000000G DROP.CTLR::
                        JSR     R1,$SAVE3                       ;                                       6773
000004  016646  000014          MOV     14(SP),-(SP)            ; CTLR,*                                6792
000010  012746  000053          MOV     #53,-(SP)
000014  004737  000000G         JSR     PC,BL$MUL
000020  010003                  MOV     R0,R3
000022  012702  000003          MOV     #3,R2                   ; *,N                                   6790
000026  010300          1$:     MOV     R3,R0                   ;                                       6792
000030  060200                  ADD     R2,R0                   ; N,*
000032  006300                  ASL     R0
000034  032760  040000  000000G BIT     #40000,CST(R0)
000042  001412                  BEQ     2$
000044  016001  000000G         MOV     CST(R0),R1              ; *,UNIT                                6795
000050  000301                  SWAB    R1                      ; UNIT
000052  042701  177760          BIC     #177760,R1              ; *,UNIT
000056  116661  000016  000000G MOVB    16(SP),DUR(R1)          ; REASON,*(UNIT)                        6796
000064  010100                  MOV     R1,R0                   ; UNIT,*                                6797
000066  104451                  TRAP    51
000070  062702  000012  2$:     ADD     #12,R2                  ; *,N                                   6790
000074  020227  000041          CMP     R2,#41                  ; N,*
```

N15


```
000100  003752                          BLE     1$
000102  022626                          CMP     (SP)+,(SP)+                     ;                       6785
000104  000207                          RTS     PC                              ;                       6773

; Routine Size:  35 words,      Routine Base:  $CODE$ + 7154
; Maximum stack depth per invocation:  8 words


;   6801  1
;   6802  1
```

```
;   6803  1     global routine DRV_CTLERR (CTLR) : novalue =
;   6804  1
;   6805  1     !+
;   6806  1     !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;   6807  1     !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
;   6808  1     !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;   6809  1     !       PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;   6810  1     !       C-RING COUNT, AND DEALLOCATING MSCP PACKETS DESCRIBED IN THE RESPONSE
;   6811  1     !       RING.
;   6812  1     !
;   6813  1     !       INPUTS:
;   6814  1     !               CTLR - DYING CONTROLLER NUMBER
;   6815  1     !-
;   6816  1
;   6817  2         begin
;   6818  2
;   6819  2         local
;   6820  2             D_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS);  ! CONTROLLER'S DCT ADDRESS
;   6821  2
;   6822  2         D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                       ! GET CONTROLLER'S DCT ADDR
;   6823  2         D_ADDR [WORD0] = OFFLINE;                                   ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;   6824  2         PUTA_PKT (.CTLR);                                           ! RELEASE ALL PACKETS ALLOCATED TO CONTROLLER
;   6825  2         DROP_CTLR (.CTLR, DU_CFATAL);                               ! DROP ALL UNITS ON THE CONTROLLER
;   6826  1         end;                                                        ! ROUTINE DRV_CTLERR

                                .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  010146         DRV.CTLERR::
                                MOV     R1,-(SP)                            ;
000002  016601  000004          MOV     4(SP),R1                            ; CTLR,*                6803
000006  010146                  MOV     R1,-(SP)                                                    6822
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                6823
000026  010116                  MOV     R1,(SP)                             ;                       6824
000030  004737  006466'          JSR     PC,PUTA.PKT
000034  010116                  MOV     R1,(SP)                             ;                       6825
000036  012746  000006          MOV     #6,-(SP)
000042  004737  007154'          JSR     PC,DROP.CTLR
000046  062706  000006          ADD     #6,SP                               ;                       6817
000052  012601                  MOV     (SP)+,R1                            ;                       6803
000054  000207                  RTS     PC

; Routine Size:  23 words,    Routine Base:  $CODE$ + 7262
; Maximum stack depth per invocation:  5 words
```

```
;  6827  1     global routine SEND (index) =
;  6828  1
;  6829  1     !+
;  6830  1     !       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;  6831  1     !       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
;  6832  1     !       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
;  6833  1     !       DEVICE'S IP REGISTER.  IF THE
;  6834  1     !       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;  6835  1     !       THE CALLER, AND NO ACTION IS TAKEN.
;  6836  1     !
;  6837  1     !       INPUTS:
;  6838  1     !               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
;  6839  1     !                       BE SENT
;  6840  1     !
;  6841  1     !       IMPLICIT INPUTS:
;  6842  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  6843  1     !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;  6844  1     !-
;  6845  1
;  6846  2         begin
;  6847  2
;  6848  2         local
;  6849  2             SLOT_ADDR,
;  6850  2             TEMP : word,
;  6851  2             CUR_PRIORITY : word;
;  6852  2
;  6853  2         if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                   ! IF CRING IS NOT FULL AND
;  6854  3            ((.DCT_ADDR [STAT] eql ONLINE) or                            !   IF DEVICE IS ONLINE OR
;  6855  3             (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                    !   IT IS A SET-CTRL-CHAR COMMAND
;  6856  2         then
;  6857  2
;  6858  4             if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
;  6859  4                      (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
;  6860  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                                                  !ZZZ
;  6861  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                                                  !ZZZ
;  6862  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                                                  !ZZZ
;  6863  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                                                  !ZZZ
;  6864  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                                                  !ZZZ
;  6865  4             (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ESP) OR                                                  !ZZZ
;  6866  3                      (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
;  6867  2             then
;  6868  3                 begin
;  6869  3                 PRINTF (DBM107, .MSCP_PKT [.index, OPCODE]);
;  6870  3                 return FAILURE;
;  6871  3                 end
;  6872  2             else
;  6873  3                 begin
;  6874  3
;  6875  3                 do
;  6876  4                     BREAK                                              ! LOOP TILL CREDIT BALANCE POSITIVE
;  6877  4                 until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
;  6878  4                        (.CREDIT_BAL gequ 1)) or
;  6879  3                       (.CREDIT_BAL gtru 1);
```

D16


```
;  6880  3
;  6881  3                MSCP_PKT [.index, CRN_LO] = (CRN_LOW = .CRN_LOW + 1);       ! ASSIGN CMD REF NUM
;  6882  3
;  6883  3                if .CRN_LOW eql 0
;  6884  3                then
;  6885  3                    CRN_HIGH = .CRN_HIGH + 1;                               ! CMD REF NUM (HIGH ORDER)
;  6886  3
;  6887  3                MSCP_PKT [.index, CRN_HI] = .CRN_HIGH;                      !
;  6888  3                SLOT_ADDR = .DCT_ADDR [CR_NEXT];                            ! ADDR OF NEXT COMMAND SLOT
;  6889  3
;  6890  3                do                                                          ! WAIT TILL NEXT SLOT HOST OWNED
;  6891  3                    BREAK                                                   !
;  6892  3                until ((.(.SLOT_ADDR + 2) and ED_OWN) eql 0);               !
;  6893  3
;  6894  3                GETPRI (CUR_PRIORITY);                                      ! NO INTERRUPTS WHILE POINTERS UPDATED
;  6895  3    !ZZZ        SETPRI (PRI04);                                             !
;  6896  3                SETPRI (.BRLEVEL);                                          !         ZZZ
;  6897  3
;  6898  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_LO];                    ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
;  6899  3                SLOT_ADDR = .SLOT_ADDR + 2;                                 ! ADVANCE TO NEXT WORD
;  6900  3                .SLOT_ADDR = .MSCP_PKT [.index, PKT_HI];                    ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
;  6901  3                .SLOT_ADDR = ..SLOT_ADDR and (not (ED_FLAG));               ! CLEAR INTERRUPT FLAG IN CASE SET
;  6902  3                .SLOT_ADDR = ..SLOT_ADDR or ED_OWN;                         ! GIVE OWNERSHIP TO CONTROLLER
;  6903  3                SLOT_ADDR = .SLOT_ADDR + 2;                                 ! ADVANCE TO NEXT COMMAND SLOT
;  6904  3
;  6905  3                if .SLOT_ADDR gtra .DCT_ADDR [CR_END]                       ! IF BEYOND END OF CRING
;  6906  3                then
;  6907  3                    SLOT_ADDR = .DCT_ADDR [CR_BEG];                         ! CYCLE BACK TO BEGINNING
;  6908  3
;  6909  3                DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                            ! RESTORE CR NEXT POINTER IN DCT
;  6910  3                DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;           ! INCR # OF COMMANDS IN CRING
;  6911  4                IF (.MSCP_PKT [.INDEX, CONNID] EQL CID_MSCP)                !IF MSCP COMMAND                ZZZ
;  6912  3                THEN (CREDIT_BAL = .CREDIT_BAL - 1);                        !DECR CREDIT BALANCE            ZZZ
;  6913  3                TEMP = .RDRX_ADDR [RCIP, RC_ALL];                           ! READ IP TO FORCE PORT TO POLL
;  6914  3                SETPRI (.CUR_PRIORITY);                                     ! LOWER PRIORITY
;  6915  3                return SUCCESS;
;  6916  3                end
;  6917  3
;  6918  2            else
;  6919  2                return FAILURE;                                             ! IF DEVICE IS NOT ONLINE
;  6920  2
;  6921  1            end;                                                            ! ROUTINE SEND


                                        .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G           SEND::JSR     R1,$SAVE3                           ;                               6827
000004  005746                        TST     -(SP)
000006  127727  000000G 000004        CMPB    @DCT.ADDR,#4                        ;                               6853
000014  103100                        BHIS    2$
000016  005777  000000G               TST     @DCT.ADDR                           ;                               6854
000022  100413                        BMI     1$
000024  016646  000014                MOV     14(SP),-(SP)                        ; INDEX,*                       6855
000030  012746  000106                MOV     #106,-(SP)
```

E16


```
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$:         MOV     14(SP),-(SP)            ; INDEX,*                       6858
000056  012746  000106                          MOV     #106,-(SP)
000062  004737  000000G                        JSR     PC,BL$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                  ;                               6859
000120  001437                                  BEQ     3$
000122  020027  000004                          CMP     R0,#4
000126  001434                                  BEQ     3$
000130  020027  000005                          CMP     R0,#5                   ;                               6861
000134  001431                                  BEQ     3$
000136  020027  000001                          CMP     R0,#1                   ;                               6862
000142  001426                                  BEQ     3$
000144  020027  000003                          CMP     R0,#3                   ;                               6863
000150  001423                                  BEQ     3$
000152  020027  000006                          CMP     R0,#6                   ;                               6864
000156  001420                                  BEQ     3$
000160  020027  000002                          CMP     R0,#2                   ;                               6865
000164  001415                                  BEQ     3$
000166  020027  000042                          CMP     R0,#42                  ;                               6866
000172  001412                                  BEQ     3$
000174  010046                                  MOV     R0,-(SP)                ;                               6869
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                   ;                               6868
000216  000504                      2$:         BR      10$                     ;                               6858
000220  104422                      3$:         TRAP    22                      ;                               6875
000222  105762  000004G                        TSTB    MSCP.PKT+4(R2)          ;                               6877
000226  001003                                  BNE     4$
000230  005737  000000G                        TST     CREDIT.BAL              ;                               6878
000234  001004                                  BNE     5$
000236  023727  000000G 000001      4$:         CMP     CREDIT.BAL,#1           ;                               6879
000244  101765                                  BLOS    3$
000246  013700  000000G              5$:         MOV     CRN.LOW,R0              ;                               6881
000252  005200                                  INC     R0
000254  010037  000000G                        MOV     R0,CRN.LOW
000260  010062  000012G                        MOV     R0,MSCP.PKT+12(R2)
000264  001002                                  BNE     6$                      ;                               6883
000266  005237  000000G                        INC     CRN.HIGH                ;                               6885
000272  013762  000000G 000014G      6$:         MOV     CRN.HIGH,MSCP.PKT+14(R2) ;                              6887
000300  013700  000000G                        MOV     DCT.ADDR,R0             ;                               6888
```

F16


```
000304  016001  000020                    MOV     20(R0),R1                       ; *,SLOT.ADDR
000310  104422                      7$:   TRAP    22                              ;                       6890
000312  032761  100000  000002            BIT     #-100000,2(R1)                  ; *,*(SLOT.ADDR)        6892
000320  001373                            BNE     7$
000322  104440                            TRAP    40                              ;                       6894
000324  010003                            MOV     R0,R3                           ; *,CUR.PRIORITY
000326  013700  000000G                   MOV     BRLEVEL,R0                      ;                       6896
000332  104441                            TRAP    41
000334  016221  000000G                   MOV     MSCP.PKT(R2),(R1)+              ; *,SLOT.ADDR           6898
000340  016211  000002G                   MOV     MSCP.PKT+2(R2),(R1)             ; *,SLOT.ADDR           6900
000344  042711  040000                    BIC     #40000,(R1)                     ; *,SLOT.ADDR           6901
000350  052721  100000                    BIS     #100000,(R1)+                   ; *,SLOT.ADDR           6902
000354  013700  000000G                   MOV     DCT.ADDR,R0                     ;                       6905
000360  020160  000012                    CMP     R1,12(R0)                       ; SLOT.ADDR,*
000364  101402                            BLOS    8$
000366  016001  000010                    MOV     10(R0),R1                       ; *,SLOT.ADDR           6907
000372  010160  000020              8$:   MOV     R1,20(R0)                       ; SLOT.ADDR,*           6909
000376  105210                            INCB    (R0)                            ;                       6910
000400  105762  000011G                   TSTB    MSCP.PKT+11(R2)                 ;                       6911
000404  001002                            BNE     9$
000406  005337  000000G                   DEC     CREDIT.BAL                      ;                       6912
000412  017716  000000G              9$:   MOV     @RDRX.ADDR,(SP)                 ; *,RC.REG              6913
000416  010300                            MOV     R3,R0                           ; CUR.PRIORITY,*        6914
000420  104441                            TRAP    41
000422  012700  000001                    MOV     #1,R0                           ;                       6858
000426  000401                            BR      11$                             ;                       6919
000430  005000                     10$:   CLR     R0
000432  005726                     11$:   TST     (SP)+                           ;                       6827
000434  000207                            RTS     PC

; Routine Size:  143 words,    Routine Base:  $CODE$ + 7340
; Maximum stack depth per invocation:  10 words
```

```
;   6922  1     global routine WAIT : novalue =
;   6923  1
;   6924  1     !+
;   6925  1     !       THE PURPOSE OF THIS ROUTINE IS TO KILL TIME UNTIL AN RDRX INTERRUPT
;   6926  1     !       RESULTS IN A RETURN PACKET INDEX BEING DEPOSITED INTO THE I/O DONE
;   6927  1     !       QUEUE (IODQ).
;   6928  1     !-
;   6929  1
;   6930  1         do
;   6931  1             BREAK                                   ! BREAK FOR ACT
;   6932  1         until .IODQ_IN neq .IODQ_OUT;


                                        .SBTTL  WAIT GLOBAL ROUTINES
000000  104422                         WAIT::
000000                                  1$:     TRAP    22
000002  023737  000000G 000000G                 CMP     IODQ.IN,IODQ.OUT                ;                       6930
000010  001773                                  BEQ     1$                              ;                       6932
000012  000207                                  RTS     PC                              ;                       6922

; Routine Size:  6 words,      Routine Base:  $CODE$ + 7776
; Maximum stack depth per invocation:  2 words


;   6933  1
```

H16


```
;    6934  1
;    6935  1     GLOBAL ROUTINE MODULAS (LO_LIMIT, HI_LIMIT) =                                  !ZZZ
;    6936  1                                                                                    !ZZZ
;    6937  1     !+ THE PURPOSE OF THIS ROUTINE IS TO GET A RANDOM NUMBER BETWEEN               !ZZZ
;    6938  1     !  THE LOW AND HIGH LIMITS. THIS SHOULD WORK FOR A 16 BIT WORD.               !ZZZ
;    6939  1     !- THE "MOD" FUNC ONLY WORKS ON 15 BITS.                                       !ZZZ
;    6940  1                                                                                    !ZZZ
;    6941  2     BEGIN                                                                          !ZZZ
;    6942  2     OWN     X : WORD;                   !VARIABLE FOR RANDOM WD TABLE              !ZZZ
;    6943  2     LOCAL   ANSWER : UNSIGNED WORD,     !FINAL ANSWER                              !ZZZ
;    6944  2             SAVESZ : UNSIGNED WORD,     !SAVES SIZE OF WINDOW                      !ZZZ
;    6945  2             SIZE : UNSIGNED WORD;       !SIZE OF WINDOW                            !ZZZ
;    6946  2                                                                                    !ZZZ
;    6947  2                                                                                    !ZZZ
;    6948  2     X = .X + 1;                                                                    !ZZZ
;    6949  2     IF .X GEQ RDM_LEN                                                              !ZZZ
;    6950  2     THEN X = 0;                         !KEEP ROTATING RANDOM NUMBERS USED         !ZZZ
;    6951  2                         !ZZZ
;    6952  2     SIZE = .HI_LIMIT - .LO_LIMIT;                                                  !ZZZ
;    6953  2     SAVESZ = .HI_LIMIT - .LO_LIMIT;                                                !ZZZ
;    6954  3     IF (.SIZE LEQU %O'077777')          !IF BIT 15 NOT SET                         !ZZZ
;    6955  3     THEN ANSWER = ((.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1))                  !ZZZ
;    6956  3                                         !ONLY 15 BIT WD, SO TAKE RANDOM SAMPLE     !ZZZ
;    6957  2     ELSE                                !16 BIT WD                                 !ZZZ
;    6958  3             BEGIN                                                                  !ZZZ
;    6959  3             SIZE = .SIZE + -1;          !MAKES SIZE A 15 BIT LENGTH, OR DIV BY 2   !ZZZ
;    6960  3             ANSWER = (.RANDOM [.X] AND %O'077777') MOD (.SIZE + 1);                !ZZZ
;    6961  3                                         !GIVES 15 BIT RANDOM NUMBER                !ZZZ
;    6962  3             ANSWER = .ANSWER + 1;       !BUILD UP TO REGULAR SIZE                  !ZZZ
;    6963  3             ANSWER = .ANSWER + (.RANDOM [.X + 1] AND 1);                           !ZZZ
;    6964  3                                         !RANDOMLY FILL BIT 0                       !ZZZ
;    6965  4             IF (.ANSWER GTRU SAVESZ) !ITS POSSIBLE TO BE 1 LARGER THAN SIZE        !ZZZ
;    6966  3             THEN ANSWER = .SAVESZ;  !SO CHECK.                                     !ZZZ
;    6967  2             END;                                                                   !ZZZ
;    6968  2     RETURN .ANSWER;                                                                !ZZZ
;    6969  1     END;                                !END MODULAS ROUTINE                       !ZZZ

010012                              X:      .BLKW   1


                                            .SBTTL  MODULAS GLOBAL ROUTINES
000000  004137  000000G             MODULAS::
                                            JSR     R1,$SAVE2                   ;                   6935
000004  005746                              TST     -(SP)
000006  005237  010012'                      INC     X                           ;                   6948
000012  023727  010012' 000020              CMP     X,#20                       ;                   6949
000020  002402                              BLT     1$
000022  005037  010012'                      CLR     X                           ;                   6950
000026  016600  000012              1$:     MOV     12(SP),R0                   ; HI.LIMIT,*        6952
000032  166600  000014                      SUB     14(SP),R0                   ; LO.LIMIT,*
000036  010001                              MOV     R0,R1                       ; *,SIZE
000040  010016                              MOV     R0,(SP)                     ; *,SAVESZ          6953
```

I16


```
000042  013700  010012'                    MOV     X,R0                            ;                               6955
000046  006300                              ASL     R0
000050  020127  077777                      CMP     R1,#77777                       ; SIZE,*                        6954
000054  101011                              BHI     2$
000056  016046  000000G                     MOV     RANDOM(R0),-(SP)                ;                               6955
000062  042716  100000                      BIC     #100000,(SP)
000066  010146                              MOV     R1,-(SP)                        ; SIZE,*
000070  005216                              INC     (SP)
000072  004737  000000G                     JSR     PC,BL$MOD
000076  000431                              BR      3$
000100  006201                      2$:     ASR     R1                              ;                               6954
000102  016046  000000G                     MOV     RANDOM(R0),-(SP)                ; SIZE                          6959
000106  042716  100000                      BIC     #100000,(SP)                    ;                               6960
000112  010146                              MOV     R1,-(SP)                        ; SIZE,*
000114  005216                              INC     (SP)
000116  004737  000000G                     JSR     PC,BL$MOD
000122  006300                              ASL     R0
000124  013701  010012'                    MOV     X,R1                            ; ANSWER                        6962
000130  006301                              ASL     R1                              ;                               6963
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                           ;
000150  060601                              ADD     SP,R1                           ; SAVESZ,*                      6965
000152  020001                              CMP     R0,R1                           ; ANSWER,*
000154  101402                              BLOS    3$
000156  016600  000004                      MOV     4(SP),R0                        ; SAVESZ,ANSWER                 6966
000162  062706  000006              3$:     ADD     #6,SP                           ;                               6935
000166  000207                              RTS     PC

; Routine Size:  60 words,    Routine Base:  $CODE$ + 10014
; Maximum stack depth per invocation:  7 words
```

```
;    6970  1     %sbttl 'ERROR MESSAGE SUBROUTINES'
;    6971  1
;    6972  1     routine EMS_SA : novalue =
;    6973  1
;    6974  1     !+
;    6975  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;    6976  1     !       THE CONTENTS OF THE SA REGISTER.
;    6977  1     !-
;    6978  1
;    6979  2         begin
;    6980  2
;    6981  2         if .SA_REG eql %o'177777'                                   ! IF CONTROLLER TIME-OUT
;    6982  2         then
;    6983  3             begin
;    6984  3             PRINTX (CRLF);
;    6985  3             PRINTX (ASTERISK);
;    6986  3             PRINTX (.CNTR_ERR [0]);
;    6987  3             end
;    6988  2         else
;    6989  2
;    6990  2             if (.SA_REG and %o'003777') lequ 22                     ! IF GENERIC CONTROLLER ERROR
;    6991  2             then
;    6992  3                 begin
;    6993  3                 PRINTX (CRLF);
;    6994  3                 PRINTX (ASTERISK);
;    6995  3                 PRINTX (.CNTR_ERR [.SA_REG and %o'003777']);
;    6996  3                 end
;    6997  2             else
;    6998  2
;    6999  2                 if ((.SA_REG and %o'003777') - 400) lequ 6          ! IF RDRX SPECIFIC CONTROLLER ERROR
;    7000  2                 then
;    7001  3                     begin
;    7002  3                     PRINTX (CRLF);
;    7003  3                     PRINTX (ASTERISK);
;    7004  3                     PRINTX (.RDRX_ERR [(.SA_REG and %o'003777') - 400]);
;    7005  3                     end
;    7006  2                 else
;    7007  2                     PRINTX (EX_SA, .SA_REG);                        ! JUST PRINT CONTENTS OF SA
;    7008  2
;    7009  2         EMS_TIM ();                                                 ! TIME
;    7010  1         end;


                                .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  010146          EMS.SA: MOV     R1,-(SP)                                ;                                       6972
000002  013701  000000G          MOV     SA.REG,R1                               ;                                       6981
000006  020127  177777          CMP     R1,#-1
000012  001023                  BNE     1$
000014  012746  000000G          MOV     #CRLF,-(SP)                             ;                                       6984
000020  012746  000001          MOV     #1,-(SP)
000024  010600                  MOV     SP,R0                                   ; SP,*
000026  104415                  TRAP    15
000030  012716  000000G          MOV     #ASTERISK,(SP)                          ;                                       6985
```

K16


```
000034  012746  000001                        MOV     #1,-(SP)
000040  010600                                MOV     SP,R0
000042  104415                                TRAP    15                      ; SP,*
000044  013716  000000G                       MOV     CNTR.ERR,(SP)           ;                               6986
000050  012746  000001                        MOV     #1,-(SP)
000054  010600                                MOV     SP,R0
000056  104415                                TRAP    15                      ; SP,*
000060  000475                                BR      3$                      ;                               6983
000062  010100                        1$:     MOV     R1,R0                   ;                               6990
000064  042700  174000                        BIC     #174000,R0
000070  020027  000026                        CMP     R0,#26
000074  101030                                BHI     2$
000076  012746  000000G                       MOV     #CRLF,-(SP)             ;                               6993
000102  012746  000001                        MOV     #1,-(SP)
000106  010600                                MOV     SP,R0
000110  104415                                TRAP    15                      ; SP,*
000112  012716  000000G                       MOV     #ASTERISK,(SP)          ;                               6994
000116  012746  000001                        MOV     #1,-(SP)
000122  010600                                MOV     SP,R0
000124  104415                                TRAP    15                      ; SP,*
000126  013700  000000G                       MOV     SA.REG,R0               ;                               6995
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
000152  104415                                TRAP    15                      ; SP,*
000154  000437                                BR      3$                      ;                               6992
000156  010100                        2$:     MOV     R1,R0                   ;                               6999
000160  042700  174000                        BIC     #174000,R0
000164  162700  000620                        SUB     #620,R0
000170  020027  000006                        CMP     R0,#6
000174  101031                                BHI     4$
000176  012746  000000G                       MOV     #CRLF,-(SP)             ;                               7002
000202  012746  000001                        MOV     #1,-(SP)
000206  010600                                MOV     SP,R0
000210  104415                                TRAP    15                      ; SP,*
000212  012716  000000G                       MOV     #ASTERISK,(SP)          ;                               7003
000216  012746  000001                        MOV     #1,-(SP)
000222  010600                                MOV     SP,R0
000224  104415                                TRAP    15                      ; SP,*
000226  013700  000000G                       MOV     SA.REG,R0               ;                               7004
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
000252  104415                                TRAP    15                      ; SP,*
000254  005726                        3$:     TST     (SP)+                   ;                               7001
000256  000407                                BR      5$                      ;                               6999
000260  010146                        4$:     MOV     R1,-(SP)                ;                               7007
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              ;                       7009
000302  062706  000006                  ADD     #6,SP                   ;                       6979
000306  012601                          MOV     (SP)+,R1                ;                       6972
000310  000207                          RTS     PC

; Routine Size:  101 words,    Routine Base:  $CODE$ + 10204
; Maximum stack depth per invocation:  7 words

;   7011  1
```

M16


```
;  7012  1     routine EMS_SBC1 (addr): novalue =                                          !MMM
;  7013  1
;  7014  1     !*
;  7015  1     !       THIS ROUTINE PRINTS (EXTENDED) THE  "SB_CODE" (SUB-CODE) IF
;  7016  1     !       EITHER THE STATUS CODE (ST_CODE) OR THE SUB-CODE IS NON-ZERO. (A
;  7017  1     !       NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
;  7018  1     !       HAS MEANING WITH A NON-ZERO STATUS CODE).
;  7019  1     !-
;  7020  1
;  7021  2         begin
;  7022  2
;  7023  2         local
;  7024  2
;  7025  2             ST_CODE : word,                                                 ! MMM
;  7026  2             SB_CODE : word,                                                 ! MMM
;  7027  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);           ! RETURN PACKET ADDRESS  MMM
;  7028  2
;  7029  2
;  7030  2         RP_ADDR = .addr;                                    ! MMM
;  7031  2         ST_CODE = .RP_ADDR [STSCOD];                        ! MMM
;  7032  2         SB_CODE = .RP_ADDR [SUBCOD];                        ! MMM
;  7033  2
;  7034  2         if (.ST_CODE or .SB_CODE) neq 0                                 ! PRINT SUB-CODE ONLY ON ERROR
;  7035  2         then
;  7036  3             begin
;  7037  3             PRINTX (EX_SB);                                             ! SUB-CODE :
;  7038  3
;  7039  3             selectoneu  .ST_CODE of                                     !MMM  ADDITIONAL SUBCODES
;  7040  3                 set                                                     !MMM
;  7041  3
;  7042  4                 [ST_SUC]:           begin                               ! SUCCESS SUB-CODES  MMM
;  7043  4                                     if .SB_CODE lequ 16                 ! SUCCESS SUB-CODES MMM
;  7044  4                                     then
;  7045  4                                     PRINTX (.TBL_SUC [.SB_CODE]);
;  7046  4                                     if .SB_CODE eqlu 32
;  7047  4                                     then
;  7048  4                                     PRINTX (.TBL_SUC [17]);
;  7049  4                                     if .SB_CODE eqlu 64
;  7050  4                                     then
;  7051  4                                     PRINTX (.TBL_SUC [18]);             !MMM
;  7052  3                                     end;                                !MMM
;  7053  3
;  7054  3                 [ST_CMD]:           PRINTX (EX_SBO, .SB_CODE / 8);      ! INVALID COMMAND
;  7055  3
;  7056  3                 [ST_ABO]:           ;                                   ! COMMAND ABORTED
;  7057  3
;  7058  3                 [ST_OFL]:           if .SB_CODE lequ 8                  ! UNIT OFFLINE
;  7059  3                                     then
;  7060  3                                         PRINTX (.TBL_OFL [.SB_CODE]);
;  7061  3
;  7062  3                 [ST_AVL]:           ;                                   ! UNIT AVAILABLE
;  7063  3
;  7064  3                 [ST_MFE]:           if .SB_CODE lequ 10                 ! MEDIA FORMAT ERROR
```

```
;  7065  3                                then
;  7066  3                                    PRINTX (.TBL_MFE [.SB_CODE]);
;  7067  3
;  7068  4                    [ST_WPT]:       begin                                   !MMM
;  7069  4                                    if .SB_CODE eqlu 8                      !MMM
;  7070  4                                    then
;  7071  4                                        PRINTX (.TBL_WPT [3]);               !MMM
;  7072  4                                    if (.SB_CODE / 128) lequ 2                ! WRITE PROTECTED
;  7073  4                                    then
;  7074  4                                        PRINTX (.TBL_WPT [(.SB_CODE / 128)]); !MMM
;  7075  3                                    end;                                    !MMM
;  7076  3
;  7077  3                    [ST_CMP]:       ;                                       ! COMPARE ERROR
;  7078  3
;  7079  3                    [ST_DAT]:       if .SB_CODE lequ 15                     ! DATA ERROR
;  7080  3                                    then
;  7081  3                                        PRINTX (.TBL_DAT [.SB_CODE]);
;  7082  3
;  7083  3                    [ST_HST]:       if .SB_CODE lequ 4                      ! HOST ACCESS ERROR
;  7084  3                                    then
;  7085  3                                        PRINTX (.TBL_HST [.SB_CODE]);
;  7086  3
;  7087  3                    [ST_CNT]:       if .SB_CODE lequ 3                      ! CONTROLLER ERROR
;  7088  3                                    then
;  7089  3                                        PRINTX (.TBL_CNT [.SB_CODE]);
;  7090  3
;  7091  3                    [ST_DRV]:       if .SB_CODE lequ 8                      ! DRIVE ERROR
;  7092  3                                    then
;  7093  3                                        PRINTX (.TBL_DRV [.SB_CODE]);
;  7094  3
;  7095  3                    [otherwise]:    PRINTX (EX_SBO, .SB_CODE);              ! JUST PRINT SUB-CODE IF NO MATCH
;  7096  3                    tes;
;  7097  3
;  7098  3
;  7099  3        !MMM    case .ST_CODE from ST_SUC to ST_DRV of
;  7100  3        !MMM        set
;  7101  3
;  7102  3        !MMM        [ST_SUC]:       if .SB_CODE lequ 16                     ! SUCCESS SUB-CODES
;  7103  3        !MMM                        then
;  7104  3        !MMM                            PRINTX (.TBL_SUC [.SB_CODE]);
;  7105  3
;  7106  3        !MMM        [ST_CMD]:       PRINTX (EX_SBO, .SB_CODE / 8);          ! INVALID COMMAND
;  7107  3
;  7108  3        !MMM        [ST_ABO]:       ;                                       ! COMMAND ABORTED
;  7109  3
;  7110  3        !MMM        [ST_OFL]:       if .SB_CODE lequ 8                      ! UNIT OFFLINE
;  7111  3        !MMM                        then
;  7112  3        !MMM                            PRINTX (.TBL_OFL [.SB_CODE]);
;  7113  3
;  7114  3        !MMM        [ST_AVL]:       ;                                       ! UNIT AVAILABLE
;  7115  3
;  7116  3        !MMM        [ST_MFE]:       if .SB_CODE lequ 10                     ! MEDIA FORMAT ERROR
;  7117  3        !MMM                        then
```

```
;   7118  3     !MMM                                PRINTX (.TBL_MFE [.SB_CODE]);
;   7119  3
;   7120  3     !MMM        [ST_WPT]:           if (.SB_CODE / 128) lequ 2                  ! WRITE PROTECTED
;   7121  3     !MMM                            then
;   7122  3     !MMM                                PRINTX (.TBL_WPT [(.SB_CODE / 128)]);
;   7123  3
;   7124  3     !MMM        [ST_CMP]:           ;                                           ! COMPARE ERROR
;   7125  3
;   7126  3     !MMM        [ST_DAT]:           if .SB_CODE lequ 15                         ! DATA ERROR
;   7127  3     !MMM                            then
;   7128  3     !MMM                                PRINTX (.TBL_DAT [.SB_CODE]);
;   7129  3
;   7130  3     !MMM        [ST_HST]:           if .SB_CODE lequ 4                          ! HOST ACCESS ERROR
;   7131  3     !MMM                            then
;   7132  3     !MMM                                PRINTX (.TBL_HST [.SB_CODE]);
;   7133  3
;   7134  3     !MMM        [ST_CNT]:           if .SB_CODE lequ 3                          ! CONTROLLER ERROR
;   7135  3     !MMM                            then
;   7136  3     !MMM                                PRINTX (.TBL_CNT [.SB_CODE]);
;   7137  3
;   7138  3     !MMM        [ST_DRV]:           if .SB_CODE lequ 8                          ! DRIVE ERROR
;   7139  3     !MMM                            then
;   7140  3     !MMM                                PRINTX (.TBL_DRV [.SB_CODE]);
;   7141  3
;   7142  3     !MMM        [outrange]:         PRINTX (EX_SBO, .SB_CODE);                  ! JUST PRINT SUB-CODE IF NO MATCH
;   7143  3     !MMM        tes;
;   7144  3
;   7145  2             end;
;   7146  2
;   7147  1         end;


                                        .SBTTL  EMS.SBC1 ERROR MESSAGE SUBROUTINES
000000  004137  000000G                 EMS.SBC1:
                                        JSR     R1,$SAVE2                   ;                                   7012
000004  016600  000010                  MOV     10(SP),R0                   ; ADDR,RP.ADDR                      7030
000010  116002  000016                  MOVB    16(R0),R2                   ; *(RP.ADDR),ST.CODE                7031
000014  042702  177740                  BIC     #177740,R2                  ; *,ST.CODE
000020  016001  000016                  MOV     16(R0),R1                   ; *(RP.ADDR),SB.CODE                7032
000024  006201                          ASR     R1                          ; SB.CODE
000026  006201                          ASR     R1                          ; SB.CODE
000030  006201                          ASR     R1                          ; SB.CODE
000032  006201                          ASR     R1                          ; SB.CODE
000034  006201                          ASR     R1                          ; SB.CODE
000036  042701  174000                  BIC     #174000,R1                  ; *,SB.CODE
000042  010100                          MOV     R1,R0                       ; SB.CODE,*                         7034
000044  050200                          BIS     R2,R0                       ; ST.CODE,*
000046  001001                          BNE     1$
000050  000207                          RTS     PC
000052  012746  000000G                 1$:     MOV     #EX.SB,-(SP)                ;                                   7037
000056  012746  000001                  MOV     #1,-(SP)
000062  010600                          MOV     SP,R0                       ; SP,*
000064  104415                          TRAP    15
```

```
000066  005702                          TST     R2                              ; ST.CODE                       7042
000070  001040                          BNE     4$
000072  020127  000020                  CMP     R1,#20                          ; SB.CODE,*                     7043
000076  101011                          BHI     2$
000100  010100                          MOV     R1,R0                           ; SB.CODE,*                     7045
000102  006300                          ASL     R0
000104  016016  000000'                  MOV     TBL.SUC(R0),(SP)
000110  012746  000001                  MOV     #1,-(SP)
000114  010600                          MOV     SP,R0                           ; SP,*
000116  104415                          TRAP    15
000120  005726                          TST     (SP)+
000122  020127  000040          2$:     CMP     R1,#40                          ; SB.CODE,*                     7046
000126  001007                          BNE     3$
000130  013716  000042'                  MOV     TBL.SUC+42,(SP)                 ;                               7048
000134  012746  000001                  MOV     #1,-(SP)
000140  010600                          MOV     SP,R0                           ; SP,*
000142  104415                          TRAP    15
000144  005726                          TST     (SP)+
000146  020127  000100          3$:     CMP     R1,#100                         ; SB.CODE,*                     7049
000152  001030                          BNE     5$
000154  013716  000044'                  MOV     TBL.SUC+44,(SP)                 ;                               7051
000160  012746  000001                  MOV     #1,-(SP)
000164  010600                          MOV     SP,R0                           ; SP,*
000166  104415                          TRAP    15
000170  000565                          BR      12$
000172  020227  000001          4$:     CMP     R2,#1                           ; ST.CODE,*                     7054
000176  001020                          BNE     6$
000200  010116                          MOV     R1,(SP)                         ; SB.CODE,*
000202  012746  000010                  MOV     #10,-(SP)
000206  004737  000000G                 JSR     PC,BL$DIV
000212  010016                          MOV     R0,(SP)
000214  012746  000000G                 MOV     #EX.SBO,-(SP)
000220  012746  000002                  MOV     #2,-(SP)
000224  010600                          MOV     SP,R0                           ; SP,*
000226  104415                          TRAP    15
000230  062706  000006                  ADD     #6,SP
000234  000137  011402'          5$:     JMP     17$                             ;                               7039
000240  020227  000002          6$:     CMP     R2,#2                           ; ST.CODE,*                     7056
000244  001773                          BEQ     5$                              ;                               7039
000246  020227  000003                  CMP     R2,#3                           ; ST.CODE,*                     7058
000252  001014                          BNE     7$
000254  020127  000010                  CMP     R1,#10                          ; SB.CODE,*
000260  101365                          BHI     5$
000262  010100                          MOV     R1,R0                           ; SB.CODE,*                     7060
000264  006300                          ASL     R0
000266  016016  000046'                  MOV     TBL.OFL(R0),(SP)
000272  012746  000001                  MOV     #1,-(SP)
000276  010600                          MOV     SP,R0                           ; SP,*
000300  104415                          TRAP    15
000302  000556                          BR      15$
000304  020227  000004          7$:     CMP     R2,#4                           ; ST.CODE,*                     7062
000310  001565                          BEQ     17$                             ;                               7039
000312  020227  000005                  CMP     R2,#5                           ; ST.CODE,*                     7064
```

E1


```
000316  001014                          BNE     8$
000320  020127  000012                  CMP     R1,#12                          ; SB.CODE,*
000324  101157                          BHI     17$
000326  010100                          MOV     R1,R0                           ; SB.CODE,*                     7066
000330  006300                          ASL     R0
000332  016016  000070'                  MOV     TBL.MFE(R0),(SP)
000336  012746  000001                  MOV     #1,-(SP)
000342  010600                          MOV     SP,R0                           ; SP,*
000344  104415                          TRAP    15
000346  000534                          BR      15$
000350  020227  000006          8$:     CMP     R2,#6                           ; ST.CODE,*                     7068
000354  001033                          BNE     10$
000356  020127  000010                  CMP     R1,#10                          ; SB.CODE,*                     7069
000362  001007                          BNE     9$
000364  013716  000124'                  MOV     TBL.WPT+6,(SP)                  ;                               7071
000370  012746  000001                  MOV     #1,-(SP)
000374  010600                          MOV     SP,R0                           ; SP,*
000376  104415                          TRAP    15
000400  005726                          TST     (SP)+
000402  010116          9$:             MOV     R1,(SP)                         ; SB.CODE,*                     7072
000404  012746  000200                  MOV     #200,-(SP)
000410  004737  000000G                 JSR     PC,BL$DIV
000414  005726                          TST     (SP)+
000416  020027  000002                  CMP     R0,#2
000422  101120                          BHI     17$
000424  006300                          ASL     R0
000426  016016  000116'                  MOV     TBL.WPT(R0),(SP)                ;                               7074
000432  012746  000001                  MOV     #1,-(SP)
000436  010600                          MOV     SP,R0                           ; SP,*
000440  104415                          TRAP    15
000442  000476                          BR      15$
000444  020227  000007          10$:    CMP     R2,#7                           ; ST.CODE,*                     7077
000450  001505                          BEQ     17$                             ;                               7039
000452  020227  000010                  CMP     R2,#10                          ; ST.CODE,*                     7079
000456  001014                          BNE     11$
000460  020127  000017                  CMP     R1,#17                          ; SB.CODE,*
000464  101077                          BHI     17$
000466  010100                          MOV     R1,R0                           ; SB.CODE,*                     7081
000470  006300                          ASL     R0
000472  016016  000126'                  MOV     TBL.DAT(R0),(SP)
000476  012746  000001                  MOV     #1,-(SP)
000502  010600                          MOV     SP,R0                           ; SP,*
000504  104415                          TRAP    15
000506  000454                          BR      15$
000510  020227  000011          11$:    CMP     R2,#11                          ; ST.CODE,*                     7083
000514  001014                          BNE     13$
000516  020127  000004                  CMP     R1,#4                           ; SB.CODE,*
000522  101060                          BHI     17$
000524  010100                          MOV     R1,R0                           ; SB.CODE,*                     7085
000526  006300                          ASL     R0
000530  016016  000166'                  MOV     TBL.HST(R0),(SP)
000534  012746  000001                  MOV     #1,-(SP)
000540  010600                          MOV     SP,R0                           ; SP,*
```

F1


```
000542  104415                          TRAP    15
000544  000435                  12$:    BR      15$
000546  020227  000012          13$:    CMP     R2,#12                  ; ST.CODE,*                     7087
000552  001014                          BNE     14$
000554  020127  000003                  CMP     R1,#3                   ; SB.CODE,*
000560  101041                          BHI     17$
000562  010100                          MOV     R1,R0                   ; SB.CODE,*                     7089
000564  006300                          ASL     R0
000566  016016  000200'                  MOV     TBL.CNT(R0),(SP)
000572  012746  000001                  MOV     #1,-(SP)
000576  010600                          MOV     SP,R0                   ; SP,*
000600  104415                          TRAP    15
000602  000416                          BR      15$
000604  020227  000013          14$:    CMP     R2,#13                  ; ST.CODE,*                     7091
000610  001015                          BNE     16$
000612  020127  000010                  CMP     R1,#10                  ; SB.CODE,*
000616  101022                          BHI     17$
000620  010100                          MOV     R1,R0                   ; SB.CODE,*                     7093
000622  006300                          ASL     R0
000624  016016  000210'                  MOV     TBL.DRV(R0),(SP)
000630  012746  000001                  MOV     #1,-(SP)
000634  010600                          MOV     SP,R0                   ; SP,*
000636  104415                          TRAP    15
000640  005726                  15$:    TST     (SP)+
000642  000410                          BR      17$                     ;                               7039
000644  010116                  16$:    MOV     R1,(SP)                 ; SB.CODE,*                     7095
000646  012746  000000G                 MOV     #EX.SB0,-(SP)
000652  012746  000002                  MOV     #2,-(SP)
000656  010600                          MOV     SP,R0                   ; SP,*
000660  104415                          TRAP    15
000662  022626                          CMP     (SP)+,(SP)+
000664  022626                  17$:    CMP     (SP)+,(SP)+             ;                               7036
000666  000207                          RTS     PC                      ;                               7012

; Routine Size:  220 words,    Routine Base:  $CODE$ + 10516
; Maximum stack depth per invocation:  10 words


;   7148  1
;   7149  1
```

```
;  7150  1
;  7151  1     routine EMS_CMD1 (addr) : novalue =                                  ! MMM
;  7152  1
;  7153  1     !+
;  7154  1     !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;  7155  1     !       PRESENT) OF THE RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;  7156  1     !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;  7157  1     !
;  7158  1     !-
;  7159  1
;  7160  2         begin
;  7161  2
;  7162  2         local
;  7163  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);          ! RETURN PACKET ADDRESS  MMM
;  7164  2
;  7165  2         RP_ADDR = .addr;                                                   ! MMM
;  7166  2         PRINTX (EX_CMD);                                                   ! "COMMAND: "
;  7167  2
;  7168  2         selectoneu (.RP_ADDR [ENDCOD] and OP_MSK) of
;  7169  2             set
;  7170  2
;  7171  2             [OP_ONL]:       PRINTX (EX_ONL);                           ! ONLINE
;  7172  2
;  7173  2             [OP_ACC]:       PRINTX (EX_ACC);                           ! ACCESS
;  7174  2
;  7175  3             [OP_RD]:        begin
;  7176  3                             PRINTX (EX_RD);                            ! READ
;  7177  3
;  7178  3     !MMM                    if .RP_ADDR [CMDMOD] neq 0
;  7179  3                             if (.RP_ADDR [CMDMOD] and MD_CMP) neq 0 !MMM
;  7180  3                             then
;  7181  3                                 PRINTX (EX_CMP);                       ! COMPARE
;  7182  3
;  7183  2                             end;
;  7184  2
;  7185  3                 [OP_WRT]:   begin
;  7186  3                             PRINTX (EX_WRT);                           ! WRITE
;  7187  3
;  7188  3     !MMM                    if .RP_ADDR [CMDMOD] neq 0
;  7189  3                             if (.RP_ADDR [CMDMOD] and MD_CMP) neq 0   !MMM
;  7190  3                             then
;  7191  3                                 PRINTX (EX_CMP);                       ! COMPARE
;  7192  3
;  7193  2                             end;
;  7194  2
;  7195  2                 [otherwise]: PRINTX (EX_OP, .RP_ADDR [ENDCOD]);        ! ENDCODE VALUE IF NO MATCH
;  7196  2                 tes;
;  7197  2
;  7198  1         end;                                                           ! ROUTINE EMS_CMD1


                                        .SBTTL  EMS.CMD1 ERROR MESSAGE SUBROUTINES
000000  004137  000000G                EMS.CMD1:
```

```
000004 016601 000010                    JSR     R1,$SAVE2                       ;
000010 012746 000000G                   MOV     10(SP),R1                       ; ADDR,RP.ADDR                  7151
000014 012746 000001                    MOV     #EX.CMD,-(SP)                   ;                               7165
000020 010600                           MOV     #1,-(SP)                        ;                               7166
000022 104415                           MOV     SP,R0                           ; SP,*
000024 116102 000014                    TRAP    15
000030 042702 177600                    MOVB    14(R1),R2                       ; *(RP.ADDR),*                  7168
000034 020227 000011                    BIC     #177600,R2
000040 001007                           CMP     R2,#11                          ;                               7171
000042 012716 000000G                   BNE     1$
000046 012746 000001                    MOV     #EX.ONL,(SP)
000052 010600                           MOV     #1,-(SP)
000054 104415                           MOV     SP,R0                           ; SP,*
000056 000462                           TRAP    15
000060 020227 000020            1$:     BR      5$
000064 001007                           CMP     R2,#20                          ;                               7173
000066 012716 000000G                   BNE     2$
000072 012746 000001                    MOV     #EX.ACC,(SP)
000076 010600                           MOV     #1,-(SP)
000100 104415                           MOV     SP,R0                           ; SP,*
000102 000450                           TRAP    15
000104 020227 000041            2$:     BR      5$
000110 001021                           CMP     R2,#41                          ;                               7175
000112 012716 000000G                   BNE     3$
000116 012746 000001                    MOV     #EX.RD,(SP)                     ;                               7176
000122 010600                           MOV     #1,-(SP)
000124 104415                           MOV     SP,R0                           ; SP,*
000126 032761 040000 000012             TRAP    15
000134 001433                           BIT     #40000,12(R1)                   ; *,*(RP.ADDR)                  7179
000136 012716 000000G                   BEQ     5$
000142 012746 000001                    MOV     #EX.CMP,(SP)                    ;                               7181
000146 010600                           MOV     #1,-(SP)
000150 104415                           MOV     SP,R0                           ; SP,*
000152 000423                           TRAP    15
000154 020227 000042            3$:     BR      4$
000160 001023                           CMP     R2,#42                          ;                               7185
000162 012716 000000G                   BNE     6$
000166 012746 000001                    MOV     #EX.WRT,(SP)                    ;                               7186
000172 010600                           MOV     #1,-(SP)
000174 104415                           MOV     SP,R0                           ; SP,*
000176 032761 040000 000012             TRAP    15
000204 001407                           BIT     #40000,12(R1)                   ; *,*(RP.ADDR)                  7189
000206 012716 000000G                   BEQ     5$
000212 012746 000001                    MOV     #EX.CMP,(SP)                    ;                               7191
000216 010600                           MOV     #1,-(SP)
000220 104415                           MOV     SP,R0                           ; SP,*
000222 005726                   4$:     TRAP    15
000224 005726                   5$:     TST     (SP)+
000226 000412                           TST     (SP)+                           ;                               7185
000230 005016                           BR      7$                              ;                               7168
000232 116116 000014            6$:     CLR     (SP)                            ;                               7195
000236 012746 000000G                   MOVB    14(R1),(SP)                     ; *(RP.ADDR),*
                                        MOV     #EX.OP,-(SP)
```

I1


```
000242  012746  000002                        MOV     #2,-(SP)
000246  010600                                MOV     SP,R0                           ; SP,*
000250  104415                                TRAP    15
000252  022626                                CMP     (SP)+,(SP)+
000254  022626                        7$:     CMP     (SP)+,(SP)+                     ;                                               7160
000256  000207                                RTS     PC                              ;                                               7151

; Routine Size:  88 words,      Routine Base:  $CODE$ + 11406
; Maximum stack depth per invocation:  9 words


;   7199  1
```

J1


```
;   7200  1     GLOBAL ROUTINE EMS_DBN : NOVALUE =                                                  !ZZZ
;   7201  1     !+                                                                                  !ZZZ
;   7202  1     !       THIS ROUTINE PRINTS THE PRESENT DBN                                         !ZZZ
;   7203  1     !                                                                                   !ZZZ
;   7204  1     !       IMPLICIT IMPUTS:                                                            !ZZZ
;   7205  1     !               CST_ADDR - ADDRESS OF CONTROLLER STATUS TABLE                       !ZZZ
;   7206  1     !-                                                                                  !ZZZ
;   7207  1                                                                                         !ZZZ
;   7208  2     BEGIN                                                                               !ZZZ
;   7209  2     PRINTB (XX13, .CDISK);                                  !"DISK XXX"                 !ZZZ
; P 7210  2     PRINTB (XX23, .CST_ADDR [.CUOFF + OF_DBN, D_DBN], .CST_ADDR                         !ZZZ
;   7211  2                            [.CUOFF + OF_DBN, D_DBN]);       !"DBN: XXXXXX."            !ZZZ
;   7212  2     PRINTB (XX32, .S_DUPPKT - 2);                           !PRINT BYTE COUNT           !ZZZ
;   7213  2     PRINTB (XX33, .S_PATTERN);                              !PRINT THE PATTERN          !ZZZ
;   7214  2     PRINTB (XX34, .(DUPPKT + .S_DUPPKT), .(DUPPKT + .S_DUPPKT));    !PRINT THE WORD READ   !ZZZ
;   7215  2     EMS_BLK (DUPPKT +2, 256);                               !PRINT WHOLE BLOCK READ     !ZZZ
;   7216  1     END;                                                    !IN OCTAL                   !ZZZ


                                .SBTTL  EMS.DBN ERROR MESSAGE SUBROUTINES
000000  013746  000000G   EMS.DBN::
                            MOV     CDISK,-(SP)                     ;                           7209
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                ;                           7211
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)           ;                           7212
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)          ;                           7213
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             ;                           7214
000132  016016  000000G             MOV     DUPPKT(R0),(SP)
000136  011646                      MOV     (SP),-(SP)
000140  012746  000000G             MOV     #XX34,-(SP)
000144  012746  000003              MOV     #3,-(SP)
```

K1


```
000150  010600                          MOV     SP,R0                           ; SP,*
000152  104414                          TRAP    14
000154  012716  000002G                 MOV     #DUPPKT+2,(SP)                  ;                       7215
000160  012746  000400                  MOV     #400,-(SP)
000164  004737  000000V                 JSR     PC,EMS.BLK
000170  062706  000034                  ADD     #34,SP                          ;                       7208
000174  000207                          RTS     PC                              ;                       7200

; Routine Size:  63 words,      Routine Base:  $CODE$ + 11666
; Maximum stack depth per invocation:  15 words


;   7217  1
;   7218  1
```

```
;    7219  1
;    7220  1     GLOBAL ROUTINE EMS_BLK (ADDR, LENGTH) : NOVALUE =                                !ZZZ
;    7221  1                                                                                     !ZZZ
;    7222  1     !+                                                                             !ZZZ
;    7223  1     !       THIS ROUTINE WILL PRINTX A BLOCK OF MEMORY, WHICH IS 'LENGTH'          !ZZZ
;    7224  1     !       WORDS LONG STARTING AT ADDRESS 'ADDR'. PRINTING IS DONE IN OCTAL       !ZZZ
;    7225  1     !       8 WDS TO A LINE.                                                       !ZZZ
;    7226  1     !-                                                                             !ZZZ
;    7227  1                                                                                    !ZZZ
;    7228  2     BEGIN                                                                          !ZZZ
;    7229  2     LITERAL                                                                        !ZZZ
;    7230  2             MASK = %o'7';                                                          !ZZZ
;    7231  2                                                                                    !ZZZ
;    7232  2     PRINTX (CRLF);                                                                 !ZZZ
;    7233  2             INCR COUNT FROM 1 TO .LENGTH DO          !FOR EACH WD TO PRINT         !ZZZ
;    7234  3                     BEGIN                                                          !ZZZ
;    7235  3                     IF ((.COUNT - 1) AND MASK) EQL 0 !IF START OF NEW LINE         !ZZZ
;    7236  3                     THEN                                                           !ZZZ
;    7237  3                             PRINTX (SPACE4);         !PRINT 4 BLANKS               !ZZZ
;    7238  3                                                                                    !ZZZ
;    7239  3                     PRINTX (EX_WRD, ..ADDR);         !PRINTX A WORD                !ZZZ
;    7240  3                     ADDR = .ADDR +2;                 !TO NEXT ADDRESS              !ZZZ
;    7241  3                                                                                    !ZZZ
;    7242  4                     IF (((.COUNT AND MASK) EQL 0) OR !END OF LINE OR               !ZZZ
;    7243  4                         (.COUNT EQL .LENGTH))        !WHEN DONE                    !ZZZ
;    7244  3                     THEN                                                           !ZZZ
;    7245  3                             PRINTX (CRLF);           !PRINT CR LF                  !ZZZ
;    7246  2                     END;                                                           !ZZZ
;    7247  1             END;                                                                   !ZZZ


                                        .SBTTL  EMS.BLK ERROR MESSAGE SUBROUTINES
000000  010146                EMS.BLK::
                                        MOV     R1,-(SP)
000002  012746  000000G                 MOV     #CRLF,-(SP)             ;                       7220
000006  012746  000001                  MOV     #1,-(SP)                ;                       7232
000012  010600                          MOV     SP,R0
000014  104415                          TRAP    15                      ; SP,*
000016  005001                          CLR     R1                      ; COUNT                 7233
000020  000445                          BR      5$
000022  010100                  1$:     MOV     R1,R0                   ; COUNT,*               7235
000024  005300                          DEC     R0
000026  032700  000007                  BIT     #7,R0
000032  001007                          BNE     2$
000034  012716  000000G                 MOV     #SPACE4,(SP)            ;                       7237
000040  012746  000001                  MOV     #1,-(SP)
000044  010600                          MOV     SP,R0
000046  104415                          TRAP    15                      ; SP,*
000050  005726                          TST     (SP)+
000052  017616  000012          2$:     MOV     @12(SP),(SP)            ; ADDR,*                7239
000056  012746  000000G                 MOV     #EX.WRD,-(SP)
000062  012746  000002                  MOV     #2,-(SP)
000066  010600                          MOV     SP,R0                   ; SP,*
```

M1


```
000070  104415                          TRAP    15
000072  062766  000002  000016          ADD     #2,16(SP)               ; *,ADDR                        7240
000100  032701  000007                  BIT     #7,R1                   ; *,COUNT                       7242
000104  001403                          BEQ     3$
000106  020166  000014                  CMP     R1,14(SP)               ; COUNT,LENGTH                  7243
000112  001007                          BNE     4$
000114  012716  000000G           3$:    MOV     #CRLF,(SP)              ;                               7245
000120  012746  000001                  MOV     #1,-(SP)
000124  010600                          MOV     SP,R0                   ; SP,*
000126  104415                          TRAP    15
000130  005726                          TST     (SP)+
000132  022626                  4$:     CMP     (SP)+,(SP)+             ;                               7234
000134  005201                  5$:     INC     R1                      ; COUNT                         7233
000136  020166  000010                  CMP     R1,10(SP)               ; COUNT,LENGTH
000142  003727                          BLE     1$
000144  022626                          CMP     (SP)+,(SP)+             ;                               7228
000146  012601                          MOV     (SP)+,R1                ;                               7220
000150  000207                          RTS     PC

; Routine Size:  53 words,      Routine Base:  $CODE$ + 12064
; Maximum stack depth per invocation:  8 words


;   7248  1
;   7249  1
```

```
;    7250  1
;    7251  1     routine EMS_LBN1 (addr) : novalue =                                    ! MMM
;    7252  1
;    7253  1     !+
;    7254  1     !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;    7255  1     !       THE RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;    7256  1     !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;    7257  1     !       PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE RETURN PACKET
;    7258  1     !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;    7259  1     !       PRINTED.
;    7260  1     !
;    7261  1     !-
;    7262  1
;    7263  2         begin
;    7264  2         local
;    7265  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);          ! RETURN PACKET ADDRESS  MMM
;    7266  2
;    7267  2         RP_ADDR = .addr;                                                   ! MMM
;    7268  2
;    7269  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                     ! IF NO BAD BLOCK FOUND
;    7270  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7271  2         then
;    7272  2             PRINTX (EX_LBN2, .RP_ADDR [LBN_HI], .RP_ADDR [LBN_LO]);                                    !ZZZ
;    7273  2
;    7274  2         if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                     ! IF BAD BLOCKS FOUND AND REPLACED
;    7275  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7276  2         then
;    7277  2             PRINTX (EX_BBU2, .RP_ADDR [BBLK_HI], .RP_ADDR [BBLK_LO]);                                  !ZZZ
;    7278  2
;    7279  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                         ! IF HOST REPLACEABLE BAD BLOCK FOUND
;    7280  3            (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7281  2         then
;    7282  2             PRINTX (EX_BB2, .RP_ADDR [BBLK_HI], .RP_ADDR [BBLK_LO]);                                   !ZZZ
;    7283  2
;    7284  2         if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                         ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;    7285  3            (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;    7286  2         then
;    7287  2             PRINTX (EX_BB12, .RP_ADDR [BBLK_HI], .RP_ADDR [BBLK_LO]);                                  !ZZZ
;    7288  1         end;


                                .SBTTL  EMS.LBN1 ERROR MESSAGE SUBROUTINES
000000  004137  000000G  EMS.LBN1:
                            JSR     R1,$SAVE3                       ;                       7251
000004  016601  000012          MOV     12(SP),R1                   ; ADDR,RP.ADDR          7267
000010  005002                  CLR     R2                          ;                       7269
000012  156102  000015          BISB    15(R1),R2                   ; *(RP.ADDR),*
000016  005003                  CLR     R3
000020  105702                  TSTB    R2
000022  100002                  BPL     1$
000024  005203                  INC     R3
000026  000417                  BR      2$
000030  032702  000100  1$:     BIT     #100,R2                     ;                       7270
```

B2


```
000034  001014                          BNE     2$
000036  016146  000050                  MOV     50(R1),-(SP)            ; *(RP.ADDR),*                  7272
000042  016146  000052                  MOV     52(R1),-(SP)            ; *(RP.ADDR),*
000046  012746  000000G                 MOV     #EX.LBN2,-(SP)
000052  012746  000003                  MOV     #3,-(SP)
000056  010600                          MOV     SP,R0                   ; SP,*
000060  104415                          TRAP    15
000062  062706  000010                  ADD     #10,SP
000066  032703  000001          2$:     BIT     #1,R3                   ;                               7274
000072  001022                          BNE     4$
000074  032702  000100                  BIT     #100,R2                 ;                               7275
000100  001414                          BEQ     3$
000102  016146  000040                  MOV     40(R1),-(SP)            ; *(RP.ADDR),*                  7277
000106  016146  000042                  MOV     42(R1),-(SP)            ; *(RP.ADDR),*
000112  012746  000000G                 MOV     #EX.BBU2,-(SP)
000116  012746  000003                  MOV     #3,-(SP)
000122  010600                          MOV     SP,R0                   ; SP,*
000124  104415                          TRAP    15
000126  062706  000010                  ADD     #10,SP
000132  032703  000001          3$:     BIT     #1,R3                   ;                               7279
000136  001417                          BEQ     5$
000140  032702  000100          4$:     BIT     #100,R2                 ;                               7280
000144  001014                          BNE     5$
000146  016146  000040                  MOV     40(R1),-(SP)            ; *(RP.ADDR),*                  7282
000152  016146  000042                  MOV     42(R1),-(SP)            ; *(RP.ADDR),*
000156  012746  000000G                 MOV     #EX.BB2,-(SP)
000162  012746  000003                  MOV     #3,-(SP)
000166  010600                          MOV     SP,R0                   ; SP,*
000170  104415                          TRAP    15
000172  062706  000010                  ADD     #10,SP
000176  006003                  5$:     ROR     R3                      ;                               7284
000200  103017                          BCC     6$
000202  032702  000100                  BIT     #100,R2                 ;                               7285
000206  001414                          BEQ     6$
000210  016146  000040                  MOV     40(R1),-(SP)            ; *(RP.ADDR),*                  7287
000214  016146  000042                  MOV     42(R1),-(SP)            ; *(RP.ADDR),*
000220  012746  000000G                 MOV     #EX.BB12,-(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                  6$:     RTS     PC                      ;                               7251

; Routine Size:  81 words,    Routine Base:  $CODE$ + 12236
; Maximum stack depth per invocation:  10 words

;   7289  1
;   7290  1
;   7291  1
;   7292  1
```

```
;   7293  1
;   7294  1     routine EMS_BC1 (addr) : novalue =                  ! MMM
;   7295  1
;   7296  1     !+
;   7297  1     !       THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE
;   7298  1     !       RETURN PACKET: THE BYTE COUNT FROM THE COMMAND PACKET AND THE
;   7299  1     !       ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE PACKET).
;   7300  1     !
;   7301  1     !-
;   7302  1
;   7303  2         begin
;   7304  2         local
;   7305  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);       ! RETURN PACKET ADDRESS  MMM
;   7306  2
;   7307  2         RP_ADDR = .addr;                                    ! MMM
;   7308  2         PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]);       ! "BYTE COUNT IN COMMAND: XXXXX."
;   7309  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);         ! "ACTUAL # OF BYTES TRANSFERRED: XXXXX."
;   7310  1         end;                                        ! ROUTINE EMS_BC1


                                .SBTTL  EMS.BC1 ERROR MESSAGE SUBROUTINES
000000  010146          EMS.BC1:MOV     R1,-(SP)                        ;                                       7294
000002  016601  000004          MOV     4(SP),R1                        ; ADDR.RP.ADDR                          7307
000006  016146  000044          MOV     44(R1),-(SP)                    ; *(RP.ADDR),*                          7308
000012  012746  000000G         MOV     #EX.CBC,-(SP)
000016  012746  000002          MOV     #2,-(SP)
000022  010600                  MOV     SP,R0                           ; SP,*
000024  104415                  TRAP    15
000026  016116  000020          MOV     20(R1),(SP)                     ; *(RP.ADDR),*                          7309
000032  012746  000000G         MOV     #EX.BC,-(SP)
000036  012746  000002          MOV     #2,-(SP)
000042  010600                  MOV     SP,R0                           ; SP,*
000044  104415                  TRAP    15
000046  062706  000012          ADD     #12,SP                          ;                                       7303
000052  012601                  MOV     (SP)+,R1                        ;                                       7294
000054  000207                  RTS     PC

; Routine Size:  23 words,    Routine Base:  $CODE$ + 12500
; Maximum stack depth per invocation:  8 words

;   7311  1
```

D2


```
;   7312  1
;   7313  1     routine EMS_BD1 (addr) : novalue =                                  ! MMM
;   7314  1
;   7315  1     !+
;   7316  1     !       THIS ROUTINE PRINTS (EXTENDED) THE TWO-WORD I/O BUFFER DESCRIPTOR
;   7317  1     !       APPEARING IN THE RETURN PACKET.
;   7318  1     !
;   7319  1     !-
;   7320  2         begin
;   7321  2
;   7322  2         local
;   7323  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);      ! RETURN PACKET ADDRESS  MMM
;   7324  2
;   7325  2         RP_ADDR = .addr;                                    ! MMM
;   7326  2         PRINTX (EX_BD, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! "I/O BUFFER DESCRIPTOR: XXXXXX XXXXXX"
;   7327  2
;   7328  1         end;


                                        .SBTTL  EMS.BD1 ERROR MESSAGE SUBROUTINES
000000  016600  000002          EMS.BD1:MOV     2(SP),R0                        ; ADDR,RP.ADDR                  7325
000004  016046  000024                  MOV     24(R0),-(SP)                    ; *(RP.ADDR),*                  7326
000010  016046  000026                  MOV     26(R0),-(SP)                    ; *(RP.ADDR),*
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                          ;                               7320
000034  000207                          RTS     PC                              ;                               7313

; Routine Size:  15 words,    Routine Base:  $CODE$ + 12556
; Maximum stack depth per invocation:  6 words

;   7329  1
```

E2


```
;    7330  1
;    7331  1      global routine EMS_R2 (addr) : novalue =                                    ! MMM
;    7332  1
;    7333  1
;    7334  1      !+
;    7335  1      !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;    7336  1      !       OF THE RETURN PACKET.
;    7337  1      !-
;    7338  1
;    7339  1
;    7340  2          begin
;    7341  2
;    7342  2          local
;    7343  2              RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);          ! RETURN PACKET ADDRESS  MMM
;    7344  2
;    7345  2
;    7346  2          RP_ADDR = .addr;                                                ! MMM
;    7347  2          EMS_SBC1 (.RP_ADDR);                                             ! MMM
;    7348  2          EMS_CMD1 (.RP_ADDR);                                            ! COMMAND (AND MODIFIER) MMM
;    7349  2
;    7350  2          if (.RP_ADDR [ENDCOD] and OP_MSK) neq OP_ONL
;    7351  2
;    7352  2          then
;    7353  2              EMS_LBN1 (.RP_ADDR);                                              ! LBN OR BAD BLOCK NUMBER  MMM
;    7354  2
;    7355  2          if ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD) or
;    7356  3             ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_WRT)
;    7357  3
;    7358  2          then
;    7359  3              begin
;    7360  3              EMS_BC1 (.RP_ADDR);                                           ! BYTE COUNTS MMM
;    7361  3              EMS_BD1 (.RP_ADDR);                                           ! I/O BUFFER DESCRIPTOR MMM
;    7362  2              end;
;    7363  2
;    7364  2          EMS_TIM ();                                                     ! TIME
;    7365  1          end;                                                            ! ROUTINE EMS_R2


                                .SBTTL  EMS.R2 ERROR MESSAGE SUBROUTINES
000000  010146          EMS.R2::MOV     R1,-(SP)                                ;                                       7331
000002  016601  000004          MOV     4(SP),R1                                ; ADDR,RP.ADDR                          7346
000006  010146                  MOV     R1,-(SP)                                ; RP.ADDR,*                             7347
000010  004737  010516'          JSR     PC,EMS.SBC1
000014  010116                  MOV     R1,(SP)                                 ; RP.ADDR,*                             7348
000016  004737  011406'          JSR     PC,EMS.CMD1
000022  116100  000014          MOVB    14(R1),R0                               ; *(RP.ADDR),*                          7350
000026  042700  177600          BIC     #177600,R0
000032  020027  000011          CMP     R0,#11
000036  001403                  BEQ     1$
000040  010116                  MOV     R1,(SP)                                 ; RP.ADDR,*                             7353
000042  004737  012236'          JSR     PC,EMS.LBN1
000046  116100  000014      1$: MOVB    14(R1),R0                               ; *(RP.ADDR),*                          7355
000052  042700  177600          BIC     #177600,R0
```

F2


```
000056  020027  000041                        CMP     R0,#41
000062  001407                                BEQ     2$
000064  116100  000014                        MOVB    14(R1),R0                       ; *(RP.ADDR),*                  7356
000070  042700  177600                        BIC     #177600,R0
000074  020027  000042                        CMP     R0,#42
000100  001006                                BNE     3$
000102  010116                        2$:     MOV     R1,(SP)                         ; RP.ADDR,*                     7360
000104  004737  012500'                        JSR     PC,EMS.BC1
000110  010116                                MOV     R1,(SP)                         ; RP.ADDR,*                     7361
000112  004737  012556'                        JSR     PC,EMS.BD1
000116  004737  000000V                3$:     JSR     PC,EMS.TIM                      ;                               7364
000122  005726                                TST     (SP)+                           ;                               7340
000124  012601                                MOV     (SP)+,R1                        ;                               7331
000126  000207                                RTS     PC

; Routine Size:  44 words,    Routine Base:  $CODE$ + 12614
; Maximum stack depth per invocation:  3 words

;   7366  1
```

```
;   7367  1     global routine EMS_R1 (addr) : novalue =
;   7368  1
;   7369  1     !+
;   7370  1     !       THIS  ROUTINE  IS  CALLED  TO  PRINT  THE  ENTIRE CONTENTS OF THE
;   7371  1     !       RETURN PACKET DESIGNATED.  HOWEVER, THE PRINTING WILL ONLY
;   7372  1     !       OCCUR IF EXTENDED ERROR PRINTING IS ENABLED.
;   7373  1     !-
;   7374  1
;   7375  2         begin
;   7376  2         local
;   7377  2             RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);          ! RETURN PACKET ADDRESS  MMM
;   7378  2
;   7379  2         RP_ADDR = .addr;                                ! MMM
;   7380  2         PRINTX (EX_RP);                                 ! "CONTENTS OF RETURN PACKET:"
;   7381  2         EMS_BLK (.RP_ADDR, RP_LEN);                     ! PRINT BLOCK OF WORDS
;   7382  1         end;


                                        .SBTTL  EMS.R1 ERROR MESSAGE SUBROUTINES
000000  010146                 EMS.R1::MOV     R1,-(SP)                        ;                                       7367
000002  016601  000004                  MOV     4(SP),R1                        ; ADDR,RP.ADDR                          7379
000006  012746  000000G                 MOV     #EX.RP,-(SP)                    ;                                       7380
000012  012746  000001                  MOV     #1,-(SP)
000016  010600                          MOV     SP,R0                           ; SP,*
000020  104415                          TRAP    15
000022  010116                          MOV     R1,(SP)                         ; RP.ADDR,*                             7381
000024  012746  000026                  MOV     #26,-(SP)
000030  004737  012064'                 JSR     PC,EMS.BLK
000034  062706  000006                  ADD     #6,SP
000040  012601                          MOV     (SP)+,R1                        ;                                       7375
000042  000207                          RTS     PC                              ;                                       7367

; Routine Size:  18 words,     Routine Base:  $CODE$ + 12744
; Maximum stack depth per invocation:  5 words


;   7383  1
```

H2


```
;   7384  1     global routine EMS_CMP (ADDR) : novalue =
;   7385  1
;   7386  1     !+
;   7387  1     ! THIS ROUTINE IS CALLED FROM 'HOST_WRT_CHK' AND PRINTS RELEVENT DATA ON A HOST
;   7388  1     ! COMPARE ERROR
;   7389  1     !-
;   7390  1
;   7391  2         begin
;   7392  2
;   7393  2         local
;   7394  2             ORIG_ADDR : ref block [RP_LEN, word] field (RP_FIELDS); ! MMM
;   7395  2
;   7396  2         ORIG_ADDR = .ADDR;                                          ! ADDRESS OF THE WRITE RETPKT
;   7397  2         PRINTB (ERR_00, .CDISK);                                    ! "DISK XXX"
;   7398  2         PRINTB (DASH);                                              !
;   7399  2         PRINTB (.ERR_COD [12]);                                     ! " - HOST COMPARE ERROR"
;   7400  2         PRINTX (EX_LBW2, .ORIG_ADDR [LBN_HI], .ORIG_ADDR [LBN_LO]); ! LBN (WRITTEN)                  ZZZ
;   7401  2         PRINTX (EX_LBR2, .RP_ADDR [LBN_HI], .RP_ADDR [LBN_LO]);     ! LBN (READ)                    ZZZ
;   7402  2         PRINTX (EX_CBW, .ORIG_ADDR [CBCNT_LO]);                     ! BYTE COUNT (WRITE)
;   7403  2         PRINTX (EX_BC, .ORIG_ADDR [BCNT_LO]);                       ! BYTE COUNT XMITTED (WRITE)
;   7404  2         PRINTX (EX_CBR, .RP_ADDR [CBCNT_LO]);                       ! BYTE COUNT (READ);
;   7405  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);                         ! BYTE COUNT XMITTED (READ)
;   7406  2         PRINTX (EX_BDW, .ORIG_ADDR [BUFF_1], .ORIG_ADDR [BUFF_0]);  ! BUFFER ADDRESS (WRITE)
;   7407  2         PRINTX (EX_BDR, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! BUFFER ADDRESS (READ)
;   7408  2         EMS_TIM ();                                                 ! TIME
;   7409  2         EMS_R2 (.ORIG_ADDR);                                        ! MMM
;   7410  2         EMS_R1 (.ORIG_ADDR);                                        ! MMM
;   7411  2         EMS_R2 (.RP_ADDR);                                          ! MMM
;   7412  2         EMS_R1 (.RP_ADDR);                                          ! MMM
;   7413  1         end;


                                        .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  010146                  EMS.CMP::
                                        MOV     R1,-(SP)                        ;                                7384
000002  016601  000004                  MOV     4(SP),R1                        ; ADDR,ORIG.ADDR                 7396
000006  013746  000000G                 MOV     CDISK,-(SP)                     ;                                7397
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)                      ;                                7398
000032  012746  000001                  MOV     #1,-(SP)
000036  010600                          MOV     SP,R0                           ; SP,*
000040  104414                          TRAP    14
000042  013716  000030G                 MOV     ERR.COD+30,(SP)                 ;                                7399
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),*                 7400
000062  016146  000052                  MOV     52(R1),-(SP)                    ; *(ORIG.ADDR),*
000066  012746  000000G                 MOV     #EX.LBW2,-(SP)
000072  012746  000003                  MOV     #3,-(SP)
```

```
000076  010600                  MOV     SP,R0                           ; SP,*
000100  104415                  TRAP    15
000102  013700  000000G         MOV     RP.ADDR,R0                      ;                                       7401
000106  016016  000050          MOV     50(R0),(SP)
000112  016046  000052          MOV     52(R0),-(SP)
000116  012746  000000G         MOV     #EX.LBR2,-(SP)
000122  012746  000003          MOV     #3,-(SP)
000126  010600                  MOV     SP,R0                           ; SP,*
000130  104415                  TRAP    15
000132  016116  000044          MOV     44(R1),(SP)                     ; *(ORIG.ADDR),*                        7402
000136  012746  000000G         MOV     #EX.CBW,-(SP)
000142  012746  000002          MOV     #2,-(SP)
000146  010600                  MOV     SP,R0                           ; SP,*
000150  104415                  TRAP    15
000152  016116  000020          MOV     20(R1),(SP)                     ; *(ORIG.ADDR),*                        7403
000156  012746  000000G         MOV     #EX.BC,-(SP)
000162  012746  000002          MOV     #2,-(SP)
000166  010600                  MOV     SP,R0                           ; SP,*
000170  104415                  TRAP    15
000172  013700  000000G         MOV     RP.ADDR,R0                      ;                                       7404
000176  016016  000044          MOV     44(R0),(SP)
000202  012746  000000G         MOV     #EX.CBR,-(SP)
000206  012746  000002          MOV     #2,-(SP)
000212  010600                  MOV     SP,R0                           ; SP,*
000214  104415                  TRAP    15
000216  013700  000000G         MOV     RP.ADDR,R0                      ;                                       7405
000222  016016  000020          MOV     20(R0),(SP)
000226  012746  000000G         MOV     #EX.BC,-(SP)
000232  012746  000002          MOV     #2,-(SP)
000236  010600                  MOV     SP,R0                           ; SP,*
000240  104415                  TRAP    15
000242  016116  000024          MOV     24(R1),(SP)                     ; *(ORIG.ADDR),*                        7406
000246  016146  000026          MOV     26(R1),-(SP)                    ; *(ORIG.ADDR),*
000252  012746  000000G         MOV     #EX.BDW,-(SP)
000256  012746  000003          MOV     #3,-(SP)
000262  010600                  MOV     SP,R0                           ; SP,*
000264  104415                  TRAP    15
000266  013700  000000G         MOV     RP.ADDR,R0                      ;                                       7407
000272  016016  000024          MOV     24(R0),(SP)
000276  016046  000026          MOV     26(R0),-(SP)
000302  012746  000000G         MOV     #EX.BDR,-(SP)
000306  012746  000003          MOV     #3,-(SP)
000312  010600                  MOV     SP,R0                           ; SP,*
000314  104415                  TRAP    15
000316  004737  000000V         JSR     PC,EMS.TIM                      ;                                       7408
000322  010116                  MOV     R1,(SP)                         ; ORIG.ADDR,*                           7409
000324  004737  012614'         JSR     PC,EMS.R2
000330  010116                  MOV     R1,(SP)                         ; ORIG.ADDR,*                           7410
000332  004737  012744'         JSR     PC,EMS.R1
000336  013716  000000G         MOV     RP.ADDR,(SP)                    ;                                       7411
000342  004737  012614'         JSR     PC,EMS.R2
000346  013716  000000G         MOV     RP.ADDR,(SP)                    ;                                       7412
000352  004737  012744'         JSR     PC,EMS.R1
```

J2


```
000356  062706  000062                          ADD     #62,SP                          ;                       7391
000362  012601                                  MOV     (SP)+,R1                        ;                       7384
000364  000207                                  RTS     PC

; Routine Size:  123 words,     Routine Base:  $CODE$ + 13010
; Maximum stack depth per invocation:  28 words
```

K2


```
;    7414  1     global routine EMS_ERR : novalue =
;    7415  1
;    7416  2         begin
;    7417  2
;    7418  2     ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;    7419  2     !
;    7420  2         PRINTB (ERR_00, .CDISK);                        ! "DISK XXX"
;    7421  2         PRINTB (DASH);                                  !
;    7422  2
;    7423  2         if (.ST_CODE gtru 0) and                        ! IF STATUS CODE IS WITHIN RANGE
;    7424  3            (.ST_CODE lequ 11)
;    7425  2         then
;    7426  3             PRINTB (.ERR_COD [.ST_CODE - 1])            ! PRINTB APPROPRIATE MESSAGE
;    7427  2         else
;    7428  2
;    7429  2             if .ST_CODE eql ST_DIA
;    7430  2             then
;    7431  3                 PRINTB (.ERR_COD [11])                  ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    7432  2             else
;    7433  2                 PRINTB (EX_SC, .ST_CODE);               ! JUST PRINT STATUS CODE WHEN NO MATCH
;    7434  2
;    7435  2         EMS_R2 (.RP_ADDR);                                      ! PRINTX OTHER RETPKT FIELDS
;    7436  2
;    7437  1         end;


                                .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G  EMS.ERR::
                        MOV     CDISK,-(SP)                     ;                       7420
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)              ;                       7421
000024  012746  000001          MOV     #1,-(SP)
000030  010600                  MOV     SP,R0                   ; SP,*
000032  104414                  TRAP    14
000034  013700  000000G         MOV     ST.CODE,R0              ;                       7423
000040  001413                  BEQ     1$
000042  020027  000013          CMP     R0,#13                  ;                       7424
000046  101010                  BHI     1$
000050  006300                  ASL     R0                      ;                       7426
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$                      ;                       7423
000070  020027  000037  1$:     CMP     R0,#37                  ;                       7429
000074  001007                  BNE     2$
000076  013716  000026G         MOV     ERR.COD+26,(SP)         ;                       7431
000102  012746  000001          MOV     #1,-(SP)
000106  010600                  MOV     SP,R0                   ; SP,*
000110  104414                  TRAP    14
```

L2


```
000112  000410                          BR      3$                              ;                               7429
000114  010016                  2$:     MOV     R0,(SP)                         ;                               7433
000116  012746  000000G                 MOV     #EX.SC,-(SP)
000122  012746  000002                  MOV     #2,-(SP)
000126  010600                          MOV     SP,R0                           ; SP,*
000130  104414                          TRAP    14
000132  005726                          TST     (SP)+
000134  013716  000000G          3$:     MOV     RP.ADDR,(SP)                    ;                               7435
000140  004737  012614'                 JSR     PC,EMS.R2
000144  062706  000012                  ADD     #12,SP                          ;                               7416
000150  000207                          RTS     PC                              ;                               7414

; Routine Size:  53 words,      Routine Base:  $CODE$ + 13376
; Maximum stack depth per invocation:  8 words
```

M2


```
;   7438  1     routine EMS_TIM : novalue =
;   7439  1
;   7440  1     !+
;   7441  1     ! THIS ROUTINE PRINTS THE TIME-OF-DAY MESSAGE
;   7442  1     !-
;   7443  1
;   7444  1     PRINTX (EX_TIM, .HOURS, .MINUTES);


                                        .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                  EMS.TIM:CLR     -(SP)                                   ;                       7444
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                                      ;                       7438

; Routine Size:  15 words,    Routine Base:  $CODE$ + 13550
; Maximum stack depth per invocation:  6 words
```

N2


```
;   7445  1     BGNMSG (EMS_01);


                                        .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.01::JSR     PC,M$EMS.01                     ;       7445
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 13606
; Maximum stack depth per invocation:  2 words

;   7446  2     PRINTB (EBS_01, MAX_UNITS);                 ! "MORE THAN XX UNITS SPECIFIED"
;   7447  1     ENDMSG;


                                        .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  012746  000004                M$EMS.01:
                                        MOV     #4,-(SP)                        ;       7446
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                           ;       7445
000024  000207                                  RTS     PC

; Routine Size:  11 words,     Routine Base:  $CODE$ + 13616
; Maximum stack depth per invocation:  5 words
```

```
;   7448  1     BGNMSG (EMS_10);


                                        .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.10::JSR     PC,M$EMS.10                     ;       7448
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 13644
; Maximum stack depth per invocation:  2 words


;   7449  2     PRINTB (EBD_10, .RDRX_ADDR + .OF_RC);           ! "NO RESPONSE AT ADDRESS XXXXXX"
;   7450  1     ENDMSG;


                                        .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.10:
                                        MOV     RDRX.ADDR,-(SP)                 ;       7449
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                           ;       7448
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 13654
; Maximum stack depth per invocation:  5 words
```

```
;   7451  1     BGNMSG (EMS_12);


                                        .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.12::JSR     PC,M$EMS.12                     ;                       7451
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 13706
; Maximum stack depth per invocation:  2 words

;   7452  2     PRINTB (EBD_12, .RDRX_ADDR);                ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX"
;   7453  1     ENDMSG;


                                        .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G         M$EMS.12:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       7452
000004  012746  000000G                 MOV     #EBD.12,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0
000016  104414                          TRAP    14                              ; SP,*
000020  062706  000006                  ADD     #6,SP
000024  000207                          RTS     PC                              ;                       7451

; Routine Size:  11 words,      Routine Base:  $CODE$ + 13716
; Maximum stack depth per invocation:  5 words
```

D3


```
;   7454  1     BGNMSG (EMS_13);


                                        .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.13::JSR     PC,M$EMS.13                     ;                       7454
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 13744
; Maximum stack depth per invocation:  2 words


;   7455  2     PRINTB (EBD_13, .STEP);                 ! "STEP X READ ERROR"
;   7456  2     EMS_SA ();                              ! PRINTX SA CONTENTS
;   7457  1     ENDMSG;


                                        .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G          M$EMS.13:
                                        MOV     STEP,-(SP)                      ;                       7455
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  010204'                 JSR     PC,EMS.SA                       ;                       7456
000024  062706  000006                  ADD     #6,SP                           ;                       7454
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 13754
; Maximum stack depth per invocation:  5 words
```

E3


```
;   7458  1     BGNMSG (EMS_14);


                                        .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.14::JSR     PC,M$EMS.14                     ;                                   7458
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14006
; Maximum stack depth per invocation:  2 words


;   7459  2     PRINTB (EBD_14, .IRDRX_ADDR);                  ! "BAD SA CODE FROM DEVICE XXXXXX"
;   7460  2     EMS_SA ();                                     ! PRINTX SA REGISTER CONTENTS
;   7461  1     ENDMSG;


                                        .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G          M$EMS.14:
                                        MOV     IRDRX.ADDR,-(SP)                ;                                   7459
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  010204'                 JSR     PC,EMS.SA                       ;                                   7460
000024  062706  000006                  ADD     #6,SP                           ;                                   7458
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14016
; Maximum stack depth per invocation:  5 words
```

F3


```
;   7462  1     BGNMSG (EMS_18);


                                        .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.18::JSR     PC,M$EMS.18                     ;                       7462
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14050
; Maximum stack depth per invocation:  2 words


;   7463  2     PRINTB (EBD_18, .CDISK);                ! "DISK XXX WENT OFFLINE"
;   7464  2     EMS_R2 (.RP_ADDR);                      ! PRINTX RELEVANT RETPKT FIELDS
;   7465  1     ENDMSG;


                                        .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.18:
                                        MOV     CDISK,-(SP)                     ;                       7463
000004  012746  000000G                 MOV     #EBD_18,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  013716  000000G                 MOV     RP.ADDR,(SP)                    ;                       7464
000024  004737  012614'                 JSR     PC,EMS.R2
000030  062706  000006                  ADD     #6,SP                           ;                       7462
000034  000207                          RTS     PC

; Routine Size:  15 words,     Routine Base:  $CODE$ + 14060
; Maximum stack depth per invocation:  5 words
```

G3


```
;   7466  1     BGNMSG (EMS_21);


                                        .SBTTL  EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.21::JSR     PC,M$EMS.21                     ;                       7466
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14116
; Maximum stack depth per invocation:  2 words

;   7467  2     EMS_R1 (.RP_ADDR);                              ! CONTENTS OF RETURN PACKET
;   7468  1     ENDMSG;


                                        .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.21:
                                        MOV     RP.ADDR,-(SP)                   ;                       7467
000004  004737  012744'                 JSR     PC,EMS.R1
000010  005726                          TST     (SP)+                           ;                       7466
000012  000207                          RTS     PC

; Routine Size:  6 words,      Routine Base:  $CODE$ + 14126
; Maximum stack depth per invocation:  2 words
```

H3


```
;   7469  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                      ;                                  7469
000004  104423                                  TRAP    23
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14142
; Maximum stack depth per invocation:  2 words


;   7470  2     EMS_DBN ();                                                              !ZZZ
;   7471  1     ENDMSG;                                                                  !ZZZ


                                                .SBTTL  M$EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  011666'                 M$EMS.22:
                                                JSR     PC,EMS.DBN                       ;                                  7470
000004  000207                                  RTS     PC                               ;                                  7469

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14152
; Maximum stack depth per invocation:  1 word
```

```
;   7472  1     BGNMSG (EMS_24);


                                          .SBTTL  EMS.24 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.24::JSR     PC,M$EMS.24                     ;                       7472
000004  104423                               TRAP    23
000006  000207                               RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14160
; Maximum stack depth per invocation:  2 words


;   7473  2     PRINTB (EBD_24, .CDISK);                    ! "DISK XXX WENT TO THE AVAILABLE STATE"
;   7474  2     EMS_R2 (.RP_ADDR);                          ! PRINTX RELEVANT RETPKT FIELDS
;   7475  1     ENDMSG;


                                          .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G             M$EMS.24:
                                          MOV     CDISK,-(SP)                     ;                       7473
000004  012746  000000G                   MOV     #EBD_24,-(SP)
000010  012746  000002                    MOV     #2,-(SP)
000014  010600                            MOV     SP,R0                           ; SP,*
000016  104414                            TRAP    14
000020  013716  000000G                   MOV     RP.ADDR,(SP)                    ;                       7474
000024  004737  012614'                   JSR     PC,EMS.R2
000030  062706  000006                    ADD     #6,SP                           ;                       7472
000034  000207                            RTS     PC

; Routine Size:  15 words,     Routine Base:  $CODE$ + 14170
; Maximum stack depth per invocation:  5 words
```

J3


```
;   7476  1     BGNMSG (EMS_30);


                                        .SBTTL  EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.30::JSR     PC,M$EMS.30                    ;                    7476
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14226
; Maximum stack depth per invocation:  2 words


;   7477  2     EMS_ERR ();                                  ! PRINT ALL RELEVANT DATA ON DETECTING AN ERROR
;   7478  1     ENDMSG;


                                        .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  013376'          M$EMS.30:
                                        JSR     PC,EMS.ERR                     ;                    7477
000004  000207                          RTS     PC                             ;                    7476

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14236
; Maximum stack depth per invocation:  1 word


;   7479  1
;   7480  1     end
;   7481  1
;   7482  0     eludom


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


;                                PSECT SUMMARY
;
;       Psect Name                Words     Attributes
;        $OWN$                       82       RW ,  D  ,  LCL,  REL,  CON
;        $CODE$                    3154       RO ,  I  ,  LCL,  REL,  CON


;                          Library Statistics
;
;                                  -------- Symbols --------      Pages      Processing
;       File                       Total    Loaded   Percent      Mapped     Time
;
;  DISK$USER:[DUNCAN.RELEASE]ZRQDA0.L16;1
;                                   412       295        71        21        00:00.1
```

K3


```
;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQDA0.BL1/LIST=ZRQDA0.LS1/OBJECT=ZRQDA0.OB1/SOURCE=PAGE:53

; Size:         2978 code + 7001 data words
; Run Time:       01:44.7
; Elapsed Time:   02:04.5
; Lines/CPU Min:   4286
; Lexemes/CPU-Min: 39925
; Memory Used:  724 pages
; Compilation Complete
```

```
;    0001  0     module ZRQDM3 (
;    0002  0
;    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                 ident = 'V02.3',
;    0005  0                 addressing_mode (absolute),
;    0006  0                 environment (noeis)
;    0007  0                 ) =
;    0008  0
;    0009  1     begin
;    0010  1
;    0011  1     %sbttl 'DECLARATIONS'
;    0012  1
;    0013  1     library 'ZRQDA0.L16';                       ! RDRX EXERCISER GLOBAL LIBRARY
;    0014  1
;    0015  1     !MMM require 'BLSMAC.REQ';                  ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    0016  1     require 'HSAXA0.BLB';                       ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    1757  1
;    1758  1     EQUALS;
;    1759  1
;    1760  1     forward routine                             ! ROUTINES APPEAR IN THIS ORDER
;    1761  1         INIT_TEST : novalue,                    !    INDENTATION IMPLIES CALLED SUBROUTINE
;    1762  1             DRIVER_INIT : novalue,
;    1763  1                 CTLR_INIT : novalue,
;    1764  1                     INI_CTLR_DAT : novalue,
;    1765  1                     REG_EXIST,
;    1766  1                     VEC_BR_TEST,
;    1767  1                     INT_GEN,
;    1768  1                     HARD_INIT,
;    1769  1                     INI_RRING : novalue,
;    1770  1                     SET_CTLR_CHAR,
;    1771  1                 UNIT_INIT : novalue,
;    1772  1                     DR_ERR : novalue,
;    1773  1             ACCESS : novalue,
;    1774  1         MULTI_DRIVE : novalue,
;    1775  1             MD_INIT : novalue,
;    1776  1                 INIT_IO_BUFF : novalue,
;    1777  1             FATAL_ERROR : novalue,
;    1778  1             QIO_OR,
;    1779  1                 QIO_OUT,
;    1780  1                 QIO_GEN : novalue,
;    1781  1                     GET_RANDOM : novalue,
;    1782  1                     QIO_UNIT : novalue,
;    1783  1                     QIO_FUNC : novalue,
;    1784  1                         DUP : NOVALUE,                          !ZZZ
;    1785  1                             DUPWRTDBN : NOVALUE,                !ZZZ
;    1786  1                             DUPREDDBN : NOVALUE,                !ZZZ
;    1787  1                                 DUPCOMMAND : NOVALUE,           !ZZZ
;    1788  1                                 DUPIDLE : NOVALUE,              !ZZZ
;    1789  1                     QIO_LBN : novalue,
;    1790  1                     QIO_SIZE : novalue,
;    1791  1                     FILL_BUFF : novalue,
;    1792  1                 PROC_RETPKT : novalue,
;    1793  1                     DIO_RETPKT : NOVALUE,                       !ZZZ
```

```
;  1794  1                        DUP_COMPARE : NOVALUE,                                  !ZZZ
;  1795  1                    IO_RETPKT : novalue,
;  1796  1                        FSET_UPAR : novalue,
;  1797  1                        HARD_ERROR : novalue,
;  1798  1                            ERR_HRD_RTNE : novalue,
;  1799  1                            ERR_HRD_RTNE_APT : novalue,
;  1800  1                        UPD_IO_TALLY : novalue,
;  1801  1                            OVF_CHK : novalue,
;  1802  1                            ROUND_OUTPUT : novalue,
;  1803  1                        HOST_WRT_CHK,
;  1804  1    !                       ERR_HRD_RTNE : novalue,
;  1805  1    !                       ERR_HRD_RTNE_APT : novalue,
;  1806  1                        SWEEP : novalue,
;  1807  1                            RPS_REM,
;  1808  1                    DR_RETPKT : novalue,
;  1809  1                AZINTO : L$ISR novalue,
;  1810  1                    AZINT : novalue,
;  1811  1    !                   FATAL_ERROR : novalue,
;  1812  1                        POLL_CRING : novalue,
;  1813  1                        POLL_RRING : novalue,
;  1814  1                            DUP_RSP : NOVALUE,
;  1815  1                            DISR_RSP : novalue,
;  1816  1                                SEQUEN : novalue,
;  1817  1    !MMM                            SCAN_ERRLOG : novalue,
;  1818  1    !MMM                                ERR_SOFT_RTNE : novalue,
;  1819  1    !MMM                                ERR_SOFT_RTNE_APT : novalue,
;  1820  1    !MMM                                SOFT_ERROR : novalue,
;  1821  1                                DATAGM : novalue,
;  1822  1                                    ERR_SOFT_RTNE : novalue,
;  1823  1                                    ERR_SOFT_RTNE_APT : novalue;
;  1824  1    !DDD                            SOFT_ERROR : novalue;
;  1825  1
;  1826  1    external
;  1827  1        CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;  1828  1                        ! RUN-TIME CONTROLLER STATUS TABLES
;  1829  1        CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;  1830  1                        ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;  1831  1        DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;  1832  1                        ! DRIVER CONTROLLER TABLES
;  1833  1        DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;  1834  1                        ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;  1835  1        RDRX_ADDR : ref rdrx field (RC_REG),
;  1836  1                        ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;  1837  1        IRDRX_ADDR : ref rdrx field (RC_REG),
;  1838  1                        ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;  1839  1        BST : BLOCKVECTOR [MAX_UNITS, 2, WORD]                                  !ZZZ
;  1840  1                        !BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS            !ZZZ
;  1841  1                        !RANDOM SEEK) MODE                                      !ZZZ
;  1842  1        TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;  1843  1                        ! STATISTICS TABLES
;  1844  1        T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;  1845  1                        ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;  1846  1        DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
```

N3


```
1                                   !INFO FROM RECEIVE AND SEND COMMANDS                ZZZ
1          TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TK DIRECTION            ZZZ
1          RDM_CNT : WORD,          !NO. OF RANDOM NOS.                KEEP\\           ZZZ
1          RANDOM : VECTOR [RDM_LEN, WORD],     !RAND NO TABLE TOGET//HER               ZZZ
1          C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
1                                   ! STATISTICS TABLE FOR CONTROLLER ERRORS
1          MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
1                                   ! MSCP PACKET POOL
1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
1                                   ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
1          PKT_USE : vector [PKT_CNT, byte, signed],
1                                   ! MSCP PACKET POOL ALLOCATION TABLE
1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
1                                   ! RETURN PACKET POOL
1          RP_USE : vector [RP_CNT, byte, signed],
1                                   ! RETURN PACKET POOL ALLOCATION TABLE
1          RP_INDX : word,          ! CURRENT RETURN PACKET INDEX
1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
1                                   ! CURRENT RETURN PACKET ADDRESS
1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
1                                   ! ERROR-LOG PACKET SAVE AREA
1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
1          IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
1          DUP_FLAGS : WORD,                                    !DUP FLAGS                                ZZZ
1          CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
1          DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
1          ST_CODE : word,                                      ! CURRENT STATUS CODE
1          SB_CODE : word,                                      ! CURRENT SUB-CODE
1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
1          TEMP1 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
1          TEMP2 : WORD,                                        !TEMPORARY STORAGE WD USED IN BGNCLN          !ZZZ
1          CREDIT_BAL : word,                                   ! CREDIT BALANCE
1          NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
1          HOURS : byte,                                        ! TIME OF DAY (HOURS)
1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
1          CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
```

```
;    1900  1          CLK_PRESENT : byte,                          ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    1901  1          HOE_FLAG : byte,                             ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    1902  1          FORCED_ERROR : byte,                         ! "FORCED ERROR" DETECTED IN LAST READ
;    1903  1          FER0_LBN : word,                             !LO LBN ADR OF THE "FORCED ERROR" BLOCK                ZZZ
;    1904  1          FER1_LBN : word,                             !HI LBN ADR OF THE "FORCED ERROR" BLOCK                ZZZ
;    1905  1    !     FER_LBN : word,                              ! LBN OF THE "FORCED ERROR" BLOCK                      ZZZ
;    1906  1          FER_BC : word,                               ! BYTE COUNT OF THE "FORCED ERROR" BLOCK               ZZZ
;    1907  1          INIT_OCCURED : byte,                         ! EXERCISER INITIALIZATION COMPLETE
;    1908  1          ADDR_VECT_OK : byte,                         ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    1909  1
;    1910  1          TYPER : VECTOR [MAX_UNITS, WORD],    !READ I/O COUNTER                          ZZZ MMM
;    1911  1          TYPEW : VECTOR [MAX_UNITS, WORD],    !WRITE I/O COUNTER                         ZZZ MMM
;    1912  1          BAL_IN_PROGRESS : VECTOR [MAX_UNITS, WORD], !FLAG SET TO BALANCE I/O TYPES      ZZZ MMM
;    1913  1          FORCE_WR : VECTOR [MAX_UNITS, WORD],         !FLAG TO ISSUE MORE WRITES WHEN BALANCING MMM
;    1914  1          CSR_MEM : WORD,                              !MMM
;    1915  1          CSR_ADD : WORD,                              !MMM
;    1916  1          TST_PAR : WORD,                              !MMM
;    1917  1          MAN_TST : WORD,                              !MMM
;    1918  1          S_PATTERN : WORD,                    !PATTERN WRITTEN TO DBNS                   ZZZ
;    1919  1          S_DUPPKT : WORD,                     !DBN BYTE COUNTER                          ZZZ
;    1920  1          P_INDEX : SIGNED WORD,               !CURRENT MESSAGE PACKET INDEX              ZZZ
;    1921  1          RD_COUNT : WORD,                     ! NUMBER OF WINCHESTER UNITS               ZZZ
;    1922  1          BRLEVEL : word,                      ! CURRENT DEVICE'S BR LEVEL                ZZZ
;    1923  1          D_FAIL : BYTE,                       !SIGNIFIES DUP TYPE ERROR                  ZZZ
;    1924  1          DBM12,
;    1925  1          DBM18,
;    1926  1          DBM19,
;    1927  1          DBM20,
;    1928  1          DBM21,
;    1929  1          DBM22,
;    1930  1          DBM23,
;    1931  1          DBM25,
;    1932  1          DBM26,
;    1933  1          DBM27,
;    1934  1          DBM29,
;    1935  1          DBM108,
;    1936  1          DBM109,
;    1937  1          DBM111,
;    1938  1          DBM112,
;    1939  1          DBM120,
;    1940  1          DBM121,
;    1941  1          DBM123,                                                                         !MMM
;    1942  1          DBM125,                                                                         !MMM
;    1943  1          DBM126,                                                                         !MMM
;    1944  1          DBM127,                                                                         !MMM
;    1945  1          DBM128,                                                                         !MMM
;    1946  1          EH_0,                                                                           !ZZZ
;    1947  1          EH_1,                                                                           !ZZZ
;    1948  1          EH_2,                                                                           !ZZZ
;    1949  1          EH_3,                                                                           !ZZZ
;    1950  1          EH_4,                                                                           !ZZZ
;    1951  1          EH_5,                                                                           !ZZZ
;    1952  1          EH_6,                                                                           !ZZZ
```

```
;    1953  1          EH_7,
;    1954  1          EH_8,
;    1955  1          EH_9,
;    1956  1          EH_10,
;    1957  1          EH_12,
;    1958  1          EH_13,
;    1959  1          MSG_02,
;    1960  1          MSG_03,
;    1961  1          EGS_02,
;    1962  1          EGD_10,
;    1963  1          EGD_11,
;    1964  1          EGD_12,
;    1965  1          EGD_13,
;    1966  1          EGD_14,
;    1967  1          EGD_15,
;    1968  1          EGD_16,
;    1969  1          EGD_17,
;    1970  1          EGD_18,
;    1971  1          EGD_19,
;    1972  1          EGD_20,
;    1973  1          EGD_21,
;    1974  1          EGD_22,
;    1975  1          EGD_23,
;    1976  1          EGD_24,
;    1977  1          EGH_30,
;    1978  1          DF_MSG,
;    1979  1          HRD_MSG,
;    1980  1          SFT_MSG,
;    1981  1          HRD_SUB,
;    1982  1          CRLF,
;    1983  1          SWP_ERROR : word,                          ! HARD ERROR LIMIT FOR DROPPING UNIT
;    1984  1          SWP_XFER : word,                           ! TRANSFER LIMIT FOR DROPPING UNIT
;    1985  1          SWP_FLAGS : word,                          ! FLAGS (SEE DOCUMENTATION)
;    1986  1           DUPROUND  : WORD,                         !DUP TESTING RATIO
;    1987  1          SWP_RAT : word,                            ! RD51/52 OPERATION RATIO
;    1988  1          SWP_DPAT : word,                           ! DATA PATTERN NUMBER
;    1989  1          SWP_UCNT : word,                           ! USER DATA PATTERN COUNT
;    1990  1          SWP_TIME : word,                           ! TIME OF DAY
;    1991  1          SWP_UDPAT : vector [MAX_UDP_CNT, word],    ! USER DATA PATTERN
;    1992  1          L$LON,
;    1993  1          L$UNIT;
;    1994  1
;    1995  1      psect
;    1996  1          own = $GGG$(read, nowrite, execute, local, concatenate);
;    1997  1
;    1998  1      own
;    1999  1          COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
;    2000  1                              ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
;    2001  1      !!ZZZ    BST : vector [MAX_UNITS, word, signed],
;    2002  1                              ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
;    2003  1          DPST : vector [MAX_UNITS, byte],           ! DATA PATTERN SEQUENCE TABLE
;    2004  1          MAX_LBN : vector [MAX_UNITS, word],        ! LARGEST LBN ALLOWED
;    2005  1          STORAGE : vector [MAX_UNITS, word],        ! DUMMY STORAGE
```

!ZZZ
!ZZZ
!ZZZ
!ZZZ
!ZZZ
!ZZZ
!ZZZ

ZZZ

```
;  2006  1          ICOM_ADDR : ref block [COMM_LEN, word] field (COM_FIELDS),
;  2007  1                      ! ADDRESS OF INTERRUPTING CONTROLLER'S COMMUNICATION AREA
;  2008  1          ICST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;  2009  1                      ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
;  2010  1          IDCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;  2011  1                      ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;  2012  1          INT_ADDR : vector [MAX_CTLR] initial (AZINT0 %(, AZINT1, AZINT2, AZINT3)%),
;  2013  1                      ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
;  2014  1      !!ZZZ    RDM_CNT : word initial (RDM_LEN),                ! NUMBER OF RANDOM NUMBERS  \ KEEP
;  2015  1      !!ZZZ    RANDOM : vector [RDM_LEN, word],                 ! RANDOM NUMBER TABLE       / TOGETHER
;  2016  1          ICTLR : word,                          ! INTERRUPTING CONTROLLING NUMBER
;  2017  1          EL_FLUSH : vector [MAX_CTLR, word],    ! STOP QIO TO PROCESS ERROR LOGS    MMM
;  2018  1          MX1 : signed word,                     ! MSCP PKT INDEX FOR FIRST QIO
;  2019  1          MX2 : signed word,                     ! MSCP PKT INDEX FOR SECOND QIO
;  2020  1          MAD1 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  2021  1                      ! ADDRESS OF MSCP PACKET FOR FIRST QIO
;  2022  1          MAD2 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  2023  1                      ! ADDRESS OF MSCP PACKET FOR SECOND QIO
;  2024  1          LAST_PKT : blockvector [MAX_CTLR, LAST_PKT_LEN, word] field (LAST_PKT_FIELDS),
;  2025  1                      ! SAVE AREA FOR INFO ABOUT LAST RESPONSE PACKET
;  2026  1
;  2027  1          RNDY0 : WORD,                                  !32-BIT RANDOM PATTERN LO WD          ZZZ
;  2028  1          RNDY1 : WORD,                                  !32-BIT RANDOM PATTERN HI WD          ZZZ
;  2029  1          FRAME_CNT : WORD,                              !WHICH 7-BIT FRAME OF R_STRING IN USE ZZZ
;  2030  1          R_STRING: WORD,                                !BITS USED IN PATTERN SELECTION       ZZZ
;  2031  1          RNDYIN :  vector [9, word] initial (%o'127102',      !NINE SEED WORDS                ZZZ
;  2032  1                    %o'143662', %o'036750', %o'121624', %o'023267',                          !ZZZ
;  2033  1                    %o'036561', %o'063714', %o'560255', %o'134230'),                         !ZZZ
;  2034  1          RNDMS0 :  vector [8, word] initial (%o'17',          !MASK FOR LOW WORD              ZZZ
;  2035  1                    %o'377', %o'7777', %o'177777', %o'177777',                               !ZZZ
;  2036  1                    %o'177777', %o'177777', %o'177777'),                                     !ZZZ
;  2037  1          RNDMS1 :  vector [8, word] initial (%o'0000',        !MASK FOR HIGH WORD             ZZZ
;  2038  1                    %o'0000', %o'0000', %o'0000', %o'17',                                    !ZZZ
;  2039  1                    %o'377', %o'7777', %o'177777'),                                          !ZZZ
;  2040  1
;  2041  1          PAT02 : vector [2] initial (1,                                 ! PATTERN 2
;  2042  1                  %o'000000'),
;  2043  1          PAT03 : vector [2] initial (1,                                 ! PATTERN 3
;  2044  1                  %o'177777'),
;  2045  1          PAT04 : vector [2] initial (1,                                 ! PATTERN 4
;  2046  1                  %o'105613'),
;  2047  1          PAT05 : vector [2] initial (1,                                 ! PATTERN 5
;  2048  1                  %o'031463'),
;  2049  1          PAT06 : vector [2] initial (1,                                 ! PATTERN 6
;  2050  1                  %o'030221'),
;  2051  1          PAT07 : vector [17] initial (16,                               ! PATTERN 7
;  2052  1                    %o'000001', %o'000003', %o'000007', %o'000017',
;  2053  1                    %o'000037', %o'000077', %o'000177', %o'000377',
;  2054  1                    %o'000777', %o'001777', %o'003777', %o'007777',
;  2055  1                    %o'017777', %o'037777', %o'077777', %o'177777'),
;  2056  1          PAT08 : vector [17] initial (16,                               ! PATTERN 8
;  2057  1                    %o'177776', %o'177774', %o'177770', %o'177760',
;  2058  1                    %o'177740', %o'177700', %o'177600', %o'177400',
```

E4


```
;    2059  1                %o'177000', %o'176000', %o'174000', %o'170000',
;    2060  1                %o'160000', %o'140000', %o'100000', %o'000000'),
;    2061  1            PAT09 : vector [17] initial (16,                          ! PATTERN 9
;    2062  1                rep 3 of (%o'000000'), rep 3 of (%o'177777'),
;    2063  1                rep 2 of (%o'000000'), rep 2 of (%o'177777'),
;    2064  1                %o'000000', %o'177777', %o'000000', %o'177777',
;    2065  1                %o'000000', %o'177777'),
;    2066  1            PAT10 : vector [2] initial (1,                            ! PATTERN 10
;    2067  1                %o'133331'),
;    2068  1            PAT11 : vector [17] initial (16,                          ! PATTERN 11
;    2069  1                rep 3 of (%o'052525'), rep 3 of (%o'125252'),
;    2070  1                rep 2 of (%o'052525'), rep 2 of (%o'125252'),
;    2071  1                %o'052525', %o'125252', %o'052525', %o'125252',
;    2072  1                %o'052525', %o'125252'),
;    2073  1            PAT12 : vector [21] initial (20,                          ! PATTERN 12
;    2074  1                rep 3 of (%o'026455'), rep 3 of (%o'151322'),
;    2075  1                rep 2 of (%o'026455'), rep 2 of (%o'151322'),
;    2076  1                rep 2 of (%o'026455'),
;    2077  1                %o'151322', %o'026455', %o'151322', %o'026455',
;    2078  1                %o'151322', %o'026455', %o'151322', %o'026455'),
;    2079  1            PAT13 : vector [2] initial (1,                            ! PATTERN 13
;    2080  1                %o'066666'),
;    2081  1            PAT14 : vector [17] initial (16,                          ! PATTERN 14
;    2082  1                %o'000001', %o'000002', %o'000004', %o'000010',
;    2083  1                %o'000020', %o'000040', %o'000100', %o'000200',
;    2084  1                %o'000400', %o'001000', %o'002000', %o'004000',
;    2085  1                %o'010000', %o'020000', %o'040000', %o'100000'),
;    2086  1            PAT15 : vector [17] initial (16,                          ! PATTERN 15
;    2087  1                %o'177776', %o'177775', %o'177773', %o'177767',
;    2088  1                %o'177757', %o'177737', %o'177677', %o'177577',
;    2089  1                %o'177377', %o'176777', %o'175777', %o'173777',
;    2090  1                %o'167777', %o'157777', %o'137777', %o'077777'),
;    2091  1            PAT16 : vector [17] initial (16,                          ! PATTERN 16
;    2092  1                rep 3 of (%o'133331'), rep 3 of (%o'155554'),
;    2093  1                rep 2 of (%o'133331'), rep 2 of (%o'155554'),
;    2094  1                %o'133331', %o'155554', %o'133331', %o'155554',
;    2095  1                %o'133331', %o'155554'),
;    2096  1            PAT17 : vector [22] initial (21,                          ! PATTERN 17
;    2097  1                %o'000000', rep 2 of (%o'106466'),
;    2098  1                rep 3 of (%o'071311'), rep 4 of (%o'106466'),
;    2099  1                rep 5 of (%o'071311'), rep 6 of (%o'106466')),
;    2100  1            PAT18 : vector [22] initial (21,                          ! PATTERN 18
;    2101  1                %o'106466', %o'000000', %o'071311',
;    2102  1                rep 3 of (%o'106466'), rep 4 of (%o'071311'),
;    2103  1                rep 5 of (%o'106466'), rep 6 of (%o'071311')),
;    2104  1            PAT19 : vector [22] initial (21,                          ! PATTERN 19
;    2105  1                %o'000000', rep 2 of (%o'134631'),
;    2106  1                rep 3 of (%o'043146'), rep 4 of (%o'134631'),
;    2107  1                rep 5 of (%o'043146'), rep 6 of (%o'134631')),
;    2108  1            PAT20 : vector [22] initial (21,                          ! PATTERN 20
;    2109  1                %o'134631', %o'000000', %o'043146',
;    2110  1                rep 3 of (%o'134631'), rep 4 of (%o'043146'),
;    2111  1                rep 5 of (%o'134631'), rep 6 of (%o'043146')),
```

F4


```
;   2112  1          PAT21 : vector [2] initial (1,                          ! PATTERN 21
;   2113  1                %o'000000'),                                      !    (LBN)
;   2114  1          DPA_TBL : vector [DP_CNT] initial                       ! DATA PATTERN ADDRESS TABLE
;   2115  1                    (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
;   2116  1                    PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
;   2117  1                    PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
;   2118  1                    PAT18, PAT19, PAT20, PAT21),
;   2119  1          BST_CNT : word initial (0),                             ! CURRENT SEQUENTIAL BLOCK COUNT
;   2120  1          BST_DEV : word initial (0),                             ! CURRENT SEQUENTIAL BLOCK DEVICE
;   2121  1          CURRENT_VECTOR : word,                                  ! CURRENT DEVICE'S VECTOR ADDRESS
;   2122  1      !ZZZ    BRLEVEL : word,                                     ! CURRENT DEVICE'S BR LEVEL                ZZZ
;   2123  1          DUOFF   : WORD,                                         !DUP OFFSET INTO CST                       ZZZ
;   2124  1          DRS_START,                                              ! START OF THE SUPERVISOR
;   2125  1          PAR_TSD : word,                                         !MMM
;   2126  1          APT_MODE : byte initial (byte (FALSE)),                 ! FLAG SET IF EXERCISER RUNNING UNDER APT
;   2127  1          MAIL_BOX_TESTNUM,                                       ! ADDRESS OF TEST NUMBER LOCATION IN APT MAIL-BOX
;   2128  1          MAIL_BOX_SUBTST,                                        ! ADDRESS OF SUB-TEST NUMBER LOCATION IN APT MAIL-BOX
;   2129  1          COMPARE_DATA : byte,                                    ! FLAG CLEARED TO BYPASS HOST COMPARES
;   2130  1          DRS_FLAGS: word,                                        ! FLAGS USED IN START/RESTART OF THE EXERCISER
;   2131  1          RD_MAX_SEQ_CNT : word,                                  ! COUNT USED IN SEQUENTIAL ACCESS OPERATIONS
;   2132  1          RX_MAX_SEQ_CNT : word;                                  !
;   2133  1
;   2134  1      external routine
;   2135  1          NEX_TRAP : L$ISR novalue,
;   2136  1          PARITY : novalue,                                       !MMM
;   2137  1          TTMF : L$ISR novalue,
;   2138  1          SET_CPAR : novalue,
;   2139  1          SET_UPAR : novalue,
;   2140  1          OUT_IODQ,
;   2141  1          IN_IODQ : novalue,
;   2142  1          GET_PKT,
;   2143  1          PUT_PKT : novalue,
;   2144  1          GET_RETPKT,
;   2145  1          PUT_RETPKT : novalue,
;   2146  1          GET_IO_BUFF : novalue,
;   2147  1          PUT_IO_BUFF : novalue,
;   2148  1          PUTA_BUFF : novalue,
;   2149  1          SEND,
;   2150  1          WAIT : novalue,
;   2151  1          MODULAS,                                                !ZZZ
;   2152  1          DROP_CTLR : novalue,
;   2153  1          DRV_CTLERR : novalue,
;   2154  1          EMS_R2 : novalue,                                       !MMM
;   2155  1          EMS_R1 : novalue,                                       !MMM
;   2156  1      !DDD    EMS_EL : novalue,
;   2157  1          EMS_CMP : novalue,
;   2158  1          EMS_ERR : novalue,
;   2159  1          EMS_10 : novalue,
;   2160  1          EMS_12 : novalue,
;   2161  1          EMS_13 : novalue,
;   2162  1          EMS_14 : novalue,
;   2163  1          EMS_18 : novalue,
;   2164  1          EMS_21 : novalue,
```

```
;   2165  1          EMS_22 : NOVALUE,
;   2166  1          EMS_24 : novalue,
;   2167  1          EMS_30 : novalue;
```

!ZZZ

```
;  2168  1     %sbttl 'TEST SECTION'
;  2169  1
;  2170  1
;  2171  1     !+
;  2172  1     !       THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RDRX EXERCISER.
;  2173  1     !       THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO A NUMBER OF
;  2174  1     !       SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;  2175  1     !-
;  2176  1
;  2177  1
;  2178  3     BGNTST;
;  2179  3
;  2180  3     local
;  2181  3         DUMMY_0 : word,
;  2182  3         DUMMY_1 : word;
;  2183  3
;  2184  3
;  2185  3     EOP_FLAG = TRUE;                                        ! ASSUME NO UNIT AVAILABLE
;  2186  3     COMPARE_DATA = TRUE;                                    ! ALLOW HOST COMAPRES IF ASKED FOR
;  2187  3     DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);              !CLEAR DUP INIT FLAG     ZZZ
;  2188  3     HOE_FLAG = FALSE;                                       ! ASSUME 'HOE' FLAG NOT SET
;  2189  3     FORCED_ERROR = FALSE;                                   ! INITIALIZE "FORCED ERROR" FLAG
;  2190  3
;  2191  3
;  2192  3     incr I from 0 to PKT_CNT - 1 do                         ! INITIALIZE PACKET AREA
;  2193  4         begin
;  2194  4
;  2195  4         incr J from 0 to PKT_LEN - 1 do
;  2196  4             MSCP_PKT [.I, .J, 0, 16, 0] = 0;
;  2197  4
;  2198  4         MSCP_PKT [.I, RSP_RECEIVED] = FALSE;
;  2199  3         end;
;  2200  3
;  2201  3     incr I from 0 to RP_CNT - 1 do                          ! INITIALIZE RESPONSE SAVE AREA
;  2202  3         incr J from 0 to RP_LEN - 1 do
;  2203  3             RETPKT [.I, .J, 0, 16, 0] = 0;
;  2204  3
;  2205  3     incr I from 0 to EP_CNT do                              ! INITIALIZE ERROR-LOG SAVE AREA
;  2206  4         begin
;  2207  4
;  2208  4         incr J from 0 to EP_LEN - 1 do
;  2209  4             ELOG_PKT [.I, .J, 0, 16, 0] = 0;
;  2210  4
;  2211  4         ELOG_PKT [.I, EL_CONTENTS] = EMPTY;
;  2212  3         end;
;  2213  3
;  2214  4     if BIT_TST (SWP_FLAGS, SWF_CWC)                         ! NO SIMULTANEOUS CNTR/HOST WRIE CHECKS
;  2215  3     then
;  2216  3         SWP_FLAGS = .SWP_FLAGS and (not SWF_HWC);
;  2217  3
;  2218  4     if BIT_TST (SWP_FLAGS, SWF_RDM)                         ! NO SIMULTANEOUS RANDOM/SEQUENTIAL SELECTS
;  2219  3     then
;  2220  3         SWP_FLAGS = .SWP_FLAGS and (not SWF_SEQ);
```

```
;    2221  3
;    2222  3      if not .INIT_OCCURED
;    2223  3      then
;    2224  4          begin
;    2225  4          DRS_START = .FREE_MEM_ADDR + 2 + (..FREE_MEM_ADDR * 2);          ! START OF SUPERVISOR
;    2226  4
;    2227  4
;    2228  4
;Z   2229  4      !-  THE FOLLOWING DETERMINES WHETHER THE TEST IS TO BE RUN IN APT MODE:                                !ZZ
;Z   2230  4                                                                                                           !ZZ
;Z   2231  5      IF BIT_TST (SWP_FLAGS, SWF_APT)                                  !IF APT                              !ZZ
;    2232  4          then                                                                                             !ZZ
;    2233  5              begin
;    2234  5              APT_MODE = TRUE;
;    2235  5              MAIL_BOX_TESTNUM = .DRS_START + %o'62' + %o'6';          ! APT MAIL-BOX IS OFFSET AT OCTAL 62 FROM
;    2236  5              MAIL_BOX_SUBTST = .DRS_START + %o'62' + %o'4';           ! BEGINNING OF SUPERVISOR
;    2237  4              end;
;    2238  4
;    2239  4
;    2240  4          NEX = FALSE;
;    2241  4          CLK_PRESENT = FALSE;                                         ! CHECK IF LINE-CLOCK PRESENT
;    2242  4          SETVEC (4, NEX_TRAP, PRI07);                                 ! SET TRAP CATCHER ADDRESS
;    2243  4          DUMMY_0 = .LINE_CLOCK;                                       ! TRY TO ADDRESS THE CLOCK
;    2244  4          DUMMY_1 = 0;                                                 ! DUMMY INSTRUCTION
;    2245  4          CLRVEC (4);                                                  ! RETURN LOC 4 TO THE SUPERVISOR
;    2246  4
;    2247  4
;    2248  4          if not .NEX
;    2249  4          then
;    2250  5              begin
;    2251  5              CLK_PRESENT = TRUE;                                      ! SET FLAG IF CLOCK PRESENT
;    2252  5              CLK_TICKS = 0;                                           ! INITIALIZE THE LINE-CLOCK TICK COUNT
;    2253  5              HOURS = .SWP_TIME / 100;                                 ! TIME OF DAY (HOURS)
;    2254  5              MINUTES = (.SWP_TIME mod 100) + 1;                       ! TIME OF DAY (MINUTES)
;    2255  5
;    2256  5              while .MINUTES gequ 60 do                                ! NORMALIZE MINUTES
;    2257  6                  begin
;    2258  6                  MINUTES = .MINUTES - 60;
;    2259  6                  HOURS = .HOURS + 1;
;    2260  5                  end;
;    2261  5
;    2262  5              HOURS = .HOURS mod 24;                                   ! NORMALIZE HOURS
;    2263  4              end;
;    2264  4
;    2265  4          NEX = FALSE;
;    2266  4          CSR_MEM = FALSE;                                             ! CHECK IF MEMORY CSR PRESENT MMM
;    2267  4          SETVEC (4, NEX_TRAP, PRI07);                                 ! MMM
;    2268  4          DUMMY_0 = ..CSR_ADD;                                         ! SET TRAP CATCHER ADDRESS
;    2269  4          DUMMY_1 = 0;                                                 ! TRY TO ADDRESS THE MEMORY CSR MMM
;    2270  4          CLRVEC (4);                                                  ! DUMMY INSTRUCTION MMM
;    2271  4                                                                       ! RETURN LOC 4 TO THE SUPERVISOR MMM
;    2272  4
;    2273  4          if not .NEX                                                  ! MMM
```

J4


```
;    2274  4          then
;    2275  5              begin
;    2276  5              CSR_MEM = TRUE;                                    ! MMM
;    2277  4              end;                                               ! MMM
;    2278  4                                                                 ! SET FLAG IF CSR PRESENT  MMM
;    2279  3          end;                                                   ! MMM
;    2280  3
;    2281  3
;    2282  3          if .CLK_PRESENT
;    2283  3          then
;    2284  4              begin
;    2285  4              SETVEC (%o'100', TIME, PRI06);                     ! LINE-CLOCK VECTOR
;    2286  4              LINE_CLOCK = BIT6;                                 ! START THE CLOCK
;    2287  3              end;
;    2288  3
;    2289  3          if .CSR_MEM and .TST_PAR                               ! MEMORY CSR PRESENT AND MMM
;    2290  3          then                                                   ! ENABLE MEMORY PARITY REQUESTED MMM
;    2291  4              begin                                              ! THEN ENABLE MEMORY PARITY MMM
;    2292  4              SETVEC (%o'114', PARITY, PRI07);                   ! MEMORY PARITY VECTOR MMM
;    2293  4              .CSR_ADD = %o'1';                                  ! ENABLE MEMORY PARITY MMM
;    2294  3              end;
;    2295  3
;    2296  3          RFLAGS (DRS_FLAGS);                                    ! READ DRS FLAGS INTO LOC DRS_FLAGS
;    2297  3
;    2298  3          if BIT_TST (DRS_FLAGS, HOE) eql HOE
;    2299  3          then
;    2300  3              HOE_FLAG = TRUE;                                   ! SET FLAG IF 'HOE' SET
;    2301  3
;    2302  3
;    2303  3          INIT_TEST ();                                          ! INITIALIZE TEST ENVIRONMENT
;    2304  3
;    2305  3          incr CTLR from 0 to (MAX_CTLR - 1) do                  ! FOR EVERY CONTROLLER
;    2306  3
;    2307  3              if (.CST [.CTLR, STATE] eql ONLINE) and            ! IF CONTROLLER ONLINE
;    2308  3                 (.DCT [.CTLR, STAT] eql ONLINE) and
;    2309  4                 (.CST [.CTLR, U_CNT] gequ 0)
;    2310  3              then
;    2311  3                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + 4) by UNIT_SIZE do
;    2312  3
;    2313  3                      if .CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE      ! IF AT LEAST ONE UNIT ALIVE
;    2314  3                      then
;    2315  4                          begin
;    2316  4                          EOP_FLAG = FALSE;                      ! NOT END OF PASS
;    2317  4                          exitloop;
;    2318  3                          end;
;    2319  3
;    2320  3          if not .EOP_FLAG
;    2321  3          then
;    2322  3              MULTI_DRIVE ();                                    ! RUN MULTI-DRIVE TEST
;    2323  1          ENDTST;


                                    .TITLE  ZRQDM3 RD/RX EXERCISER
```

K4


```
                                .IDENT  /V02.3/
                                .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                  EL.FLUSH:
                                .BLKW   1
000110                  MX1:    .BLKW   1
000112                  MX2:    .BLKW   1
000114                  MAD1:   .BLKW   1
000116                  MAD2:   .BLKW   1
000120                  LAST.PKT:
                                .BLKW   3
000126                  RNDY0:  .BLKW   1
000130                  RNDY1:  .BLKW   1
000132                  FRAME.CNT:
                                .BLKW   1
000134                  R.STRING:
                                .BLKW   1
000136  127102          RNDYIN: .WORD   -50676
000140  143662                  .WORD   -34116
000142  036750                  .WORD   36750
000144  121624                  .WORD   -56154
000146  023267                  .WORD   23267
000150  036561                  .WORD   36561
000152  063714                  .WORD   63714
000154  160255                  .WORD   -17523
000156  134230                  .WORD   -43550
000160  000017          RNDMS0: .WORD   17
000162  000377                  .WORD   377
000164  007777                  .WORD   7777
000166  177777                  .WORD   -1
000170  177777                  .WORD   -1
000172  177777                  .WORD   -1
000174  177777                  .WORD   -1
000176  177777                  .WORD   -1
000200  000000          RNDMS1: .WORD   0
000202  000000                  .WORD   0
000204  000000                  .WORD   0
```

```
000206  000000                          .WORD   0
000210  000017                          .WORD   17
000212  000377                          .WORD   377
000214  007777                          .WORD   7777
000216  177777                          .WORD   -1
000220  000001                  PAT02:  .WORD   1
000222  000000                          .WORD   0
000224  000001                  PAT03:  .WORD   1
000226  177777                          .WORD   -1
000230  000001                  PAT04:  .WORD   1
000232  105613                          .WORD   -72165
000234  000001                  PAT05:  .WORD   1
000236  031463                          .WORD   31463
000240  000001                  PAT06:  .WORD   1
000242  030221                          .WORD   30221
000244  000020                  PAT07:  .WORD   20
000246  000001                          .WORD   1
000250  000003                          .WORD   3
000252  000007                          .WORD   7
000254  000017                          .WORD   17
000256  000037                          .WORD   37
000260  000077                          .WORD   77
000262  000177                          .WORD   177
000264  000377                          .WORD   377
000266  000777                          .WORD   777
000270  001777                          .WORD   1777
000272  003777                          .WORD   3777
000274  007777                          .WORD   7777
000276  017777                          .WORD   17777
000300  037777                          .WORD   37777
000302  077777                          .WORD   77777
000304  177777                          .WORD   -1
000306  000020                  PAT08:  .WORD   20
000310  177776                          .WORD   -2
000312  177774                          .WORD   -4
000314  177770                          .WORD   -10
000316  177760                          .WORD   -20
000320  177740                          .WORD   -40
000322  177700                          .WORD   -100
000324  177600                          .WORD   -200
000326  177400                          .WORD   -400
000330  177000                          .WORD   -1000
000332  176000                          .WORD   -2000
000334  174000                          .WORD   -4000
000336  170000                          .WORD   -10000
000340  160000                          .WORD   -20000
000342  140000                          .WORD   -40000
000344  100000                          .WORD   -100000
000346  000000                          .WORD   0
000350  000020                  PAT09:  .WORD   20
000352  000000                          .WORD   0
000354  000000                          .WORD   0
000356  000000                          .WORD   0
```

M4


```
000360  177777                          .WORD   -1
000362  177777                          .WORD   -1
000364  177777                          .WORD   -1
000366  000000                          .WORD   0
000370  000000                          .WORD   0
000372  177777                          .WORD   -1
000374  177777                          .WORD   -1
000376  000000                          .WORD   0
000400  177777                          .WORD   -1
000402  000000                          .WORD   0
000404  177777                          .WORD   -1
000406  000000                          .WORD   0
000410  177777                          .WORD   -1
000412  000001                  PAT10:  .WORD   1
000414  133331                          .WORD   -44447
000416  000020                  PAT11:  .WORD   20
000420  052525                          .WORD   52525
000422  052525                          .WORD   52525
000424  052525                          .WORD   52525
000426  125252                          .WORD   -52526
000430  125252                          .WORD   -52526
000432  125252                          .WORD   -52526
000434  052525                          .WORD   52525
000436  052525                          .WORD   52525
000440  125252                          .WORD   -52526
000442  125252                          .WORD   -52526
000444  052525                          .WORD   52525
000446  125252                          .WORD   -52526
000450  052525                          .WORD   52525
000452  125252                          .WORD   -52526
000454  052525                          .WORD   52525
000456  125252                          .WORD   -52526
000460  000024                  PAT12:  .WORD   24
000462  026455                          .WORD   26455
000464  026455                          .WORD   26455
000466  026455                          .WORD   26455
000470  151322                          .WORD   -26456
000472  151322                          .WORD   -26456
000474  151322                          .WORD   -26456
000476  026455                          .WORD   26455
000500  026455                          .WORD   26455
000502  151322                          .WORD   -26456
000504  151322                          .WORD   -26456
000506  026455                          .WORD   26455
000510  026455                          .WORD   26455
000512  151322                          .WORD   -26456
000514  026455                          .WORD   26455
000516  151322                          .WORD   -26456
000520  026455                          .WORD   26455
000522  151322                          .WORD   -26456
000524  026455                          .WORD   26455
000526  151322                          .WORD   -26456
000530  026455                          .WORD   26455
```

```
000532  000001                  PAT13:  .WORD   1
000534  066666                          .WORD   66666
000536  000020                  PAT14:  .WORD   20
000540  000001                          .WORD   1
000542  000002                          .WORD   2
000544  000004                          .WORD   4
000546  000010                          .WORD   10
000550  000020                          .WORD   20
000552  000040                          .WORD   40
000554  000100                          .WORD   100
000556  000200                          .WORD   200
000560  000400                          .WORD   400
000562  001000                          .WORD   1000
000564  002000                          .WORD   2000
000566  004000                          .WORD   4000
000570  010000                          .WORD   10000
000572  020000                          .WORD   20000
000574  040000                          .WORD   40000
000576  100000                          .WORD   -100000
000600  000020                  PAT15:  .WORD   20
000602  177776                          .WORD   -2
000604  177775                          .WORD   -3
000606  177773                          .WORD   -5
000610  177767                          .WORD   -11
000612  177757                          .WORD   -21
000614  177737                          .WORD   -41
000616  177677                          .WORD   -101
000620  177577                          .WORD   -201
000622  177377                          .WORD   -401
000624  176777                          .WORD   -1001
000626  175777                          .WORD   -2001
000630  173777                          .WORD   -4001
000632  167777                          .WORD   -10001
000634  157777                          .WORD   -20001
000636  137777                          .WORD   -40001
000640  077777                          .WORD   77777
000642  000020                  PAT16:  .WORD   20
000644  133331                          .WORD   -44447
000646  133331                          .WORD   -44447
000650  133331                          .WORD   -44447
000652  155554                          .WORD   -22224
000654  155554                          .WORD   -22224
000656  155554                          .WORD   -22224
000660  133331                          .WORD   -44447
000662  133331                          .WORD   -44447
000664  155554                          .WORD   -22224
000666  155554                          .WORD   -22224
000670  133331                          .WORD   -44447
000672  155554                          .WORD   -22224
000674  133331                          .WORD   -44447
000676  155554                          .WORD   -22224
000700  133331                          .WORD   -44447
000702  155554                          .WORD   -22224
```

B5


```
000704  000025                  PAT17:  .WORD   25
000706  000000                          .WORD   0
000710  106466                          .WORD   -71312
000712  106466                          .WORD   -71312
000714  071311                          .WORD   71311
000716  071311                          .WORD   71311
000720  071311                          .WORD   71311
000722  106466                          .WORD   -71312
000724  106466                          .WORD   -71312
000726  106466                          .WORD   -71312
000730  106466                          .WORD   -71312
000732  071311                          .WORD   71311
000734  071311                          .WORD   71311
000736  071311                          .WORD   71311
000740  071311                          .WORD   71311
000742  071311                          .WORD   71311
000744  106466                          .WORD   -71312
000746  106466                          .WORD   -71312
000750  106466                          .WORD   -71312
000752  106466                          .WORD   -71312
000754  106466                          .WORD   -71312
000756  106466                          .WORD   -71312
000760  000025                  PAT18:  .WORD   25
000762  106466                          .WORD   -71312
000764  000000                          .WORD   0
000766  071311                          .WORD   71311
000770  106466                          .WORD   -71312
000772  106466                          .WORD   -71312
000774  106466                          .WORD   -71312
000776  071311                          .WORD   71311
001000  071311                          .WORD   71311
001002  071311                          .WORD   71311
001004  071311                          .WORD   71311
001006  106466                          .WORD   -71312
001010  106466                          .WORD   -71312
001012  106466                          .WORD   -71312
001014  106466                          .WORD   -71312
001016  106466                          .WORD   -71312
001020  071311                          .WORD   71311
001022  071311                          .WORD   71311
001024  071311                          .WORD   71311
001026  071311                          .WORD   71311
001030  071311                          .WORD   71311
001032  071311                          .WORD   71311
001034  000025                  PAT19:  .WORD   25
001036  000000                          .WORD   0
001040  134631                          .WORD   -43147
001042  134631                          .WORD   -43147
001044  043146                          .WORD   43146
001046  043146                          .WORD   43146
001050  043146                          .WORD   43146
001052  134631                          .WORD   -43147
001054  134631                          .WORD   -43147
```

```
001056  134631                          .WORD   -43147
001060  134631                          .WORD   -43147
001062  043146                          .WORD   43146
001064  043146                          .WORD   43146
001066  043146                          .WORD   43146
001070  043146                          .WORD   43146
001072  043146                          .WORD   43146
001074  134631                          .WORD   -43147
001076  134631                          .WORD   -43147
001100  134631                          .WORD   -43147
001102  134631                          .WORD   -43147
001104  134631                          .WORD   -43147
001106  134631                          .WORD   -43147
001110  000025                  PAT20:  .WORD   25
001112  134631                          .WORD   -43147
001114  000000                          .WORD   0
001116  043146                          .WORD   43146
001120  134631                          .WORD   -43147
001122  134631                          .WORD   -43147
001124  134631                          .WORD   -43147
001126  043146                          .WORD   43146
001130  043146                          .WORD   43146
001132  043146                          .WORD   43146
001134  043146                          .WORD   43146
001136  134631                          .WORD   -43147
001140  134631                          .WORD   -43147
001142  134631                          .WORD   -43147
001144  134631                          .WORD   -43147
001146  134631                          .WORD   -43147
001150  043146                          .WORD   43146
001152  043146                          .WORD   43146
001154  043146                          .WORD   43146
001156  043146                          .WORD   43146
001160  043146                          .WORD   43146
001162  043146                          .WORD   43146
001164  000001                  PAT21:  .WORD   1
001166  000000                          .WORD   0
001170  000000G                 DPA.TBL:.WORD   RDM.CNT
001172  000220'                         .WORD   PAT02
001174  000224'                         .WORD   PAT03
001176  000230'                         .WORD   PAT04
001200  000234'                         .WORD   PAT05
001202  000240'                         .WORD   PAT06
001204  000244'                         .WORD   PAT07
001206  000306'                         .WORD   PAT08
001210  000350'                         .WORD   PAT09
001212  000412'                         .WORD   PAT10
001214  000416'                         .WORD   PAT11
001216  000460'                         .WORD   PAT12
001220  000532'                         .WORD   PAT13
001222  000536'                         .WORD   PAT14
001224  000600'                         .WORD   PAT15
001226  000642'                         .WORD   PAT16
```

```
001230  000704'                         .WORD   PAT17
001232  000760'                         .WORD   PAT18
001234  001034'                         .WORD   PAT19
001236  001110'                         .WORD   PAT20
001240  001164'                         .WORD   PAT21
001242  000000                 BST.CNT: .WORD   0
001244  000000                 BST.DEV: .WORD   0
001246                         CURRENT.VECTOR:
                                        .BLKW   1
001250                         DUOFF:   .BLKW   1
001252                         DRS.START:
                                        .BLKW   1
001254                         PAR.TSD: .BLKW   1
001256                         APT.MODE:
001256  000                             .BYTE   0
                                        .EVEN
001260                         MAIL.BOX.TESTNUM:
                                        .BLKW   1
001262                         MAIL.BOX.SUBTST:
                                        .BLKW   1
001264                         COMPARE.DATA:
                                        .BLKB   1
                                        .EVEN
001266                         DRS.FLAGS:
                                        .BLKW   1
001270                         RD.MAX.SEQ.CNT:
                                        .BLKW   1
001272                         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, CTLR, 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, TEMP1
                               .GLOBL  TEMP2, CREDIT.BAL, NEXT.PKT.USE
                               .GLOBL  HOURS, MINUTES, CLK.TICKS, CLK.PRESENT
                               .GLOBL  HOE.FLAG, FORCED.ERROR, FERO.LBN
                               .GLOBL  FER1.LBN, FER.BC, INIT.OCCURED
                               .GLOBL  ADDR.VECT.OK, TYPER, TYPEW, BAL.IN.PROGRESS
                               .GLOBL  FORCE.WR, CSR.MEM, CSR.ADD, TST.PAR
                               .GLOBL  MAN.TST, S.PATTERN, S.DUPPKT, P.INDEX
                               .GLOBL  RD.COUNT, BRLEVEL, D.FAIL, DBM12
                               .GLOBL  DBM18, DBM19, DBM20, DBM21, DBM22
                               .GLOBL  DBM23, DBM25, DBM26, DBM27, DBM29
```

```
                .GLOBL  DBM108, DBM109, DBM111, DBM112
                .GLOBL  DBM120, DBM121, DBM123, DBM125
                .GLOBL  DBM126, DBM127, DBM128, 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, PARITY, TIME, SET.CPAR
                .GLOBL  SET.UPAR, QUT.IODQ, IN.IODQ, GET.PKT
                .GLOBL  PUT.PKT, GET.RETPKT, PUT.RETPKT
                .GLOBL  GET.IO.BUFF, PUT.IO.BUFF, PUTA.BUFF
                .GLOBL  SEND, WAIT, MODULAS, DROP.CTLR
                .GLOBL  DRV.CTLERR, EMS.R2, EMS.R1, EMS.CMP
                .GLOBL  EMS.ERR, EMS.10, EMS.12, EMS.13
                .GLOBL  EMS.14, EMS.18, EMS.21, EMS.22
                .GLOBL  EMS.24, EMS.30

000001          ON==            1
000002          OFF==           2
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
```

F5


```
000035          EF.NEW==        35
000034          EF.PWR==        34
000040          EF.START==      40
000037          EF.RESTART==    37
000036          EF.CONTINUE==   36
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


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

000000 004137 000000G   $T1:    JSR     R1,$SAVE3                       ;                       2167
000004 112737 000001 000000G    MOVB    #1,EOP.FLAG                     ;                       2185
000012 112737 000001 001264'    MOVB    #1,COMPARE.DATA                 ;                       2186
000020 042737 000002 000000G    BIC     #2,DUP.FLAGS                    ;                       2187
000026 105037 000000G           CLRB    HOE.FLAG                        ;                       2188
000032 105037 000000G           CLRB    FORCED.ERROR                    ;                       2189
000036 005002                   CLR     R2                              ; I                     2192
000040 010246           1$:     MOV     R2,-(SP)                        ; I,*                   2196
000042 012746 000043            MOV     #43,-(SP)
000046 004737 000000G           JSR     PC,BL$MUL
000052 005001                   CLR     R1
000054 010003           2$:     MOV     R0,R3                           ; J                     2195
000056 060103                   ADD     R1,R3                           ;                       2196
000060 006303                   ASL     R3                              ; J,*
000062 005063 000000G           CLR     MSCP.PKT(R3)
000066 005201                   INC     R1                              ; J                     2195
000070 020127 000042            CMP     R1,#42                          ; J,*
000074 003767                   BLE     2$
000076 010216                   MOV     R2,(SP)                         ; I,*                   2198
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                   ;                       2193
000120  005202                           INC     R2                      ; I                     2192
000122  020227  000013                   CMP     R2,#13                  ; I,*
000126  003744                           BLE     1$
000130  005002                           CLR     R2                      ; I                     2201
000132  005001                   3$:     CLR     R1                      ; J                     2202
000134  010200                   4$:     MOV     R2,R0                   ; I,*                   2203
000136  060100                           ADD     R1,R0                   ; J,*
000140  006300                           ASL     R0
000142  005060  000000G                  CLR     RETPKT(R0)
000146  005201                           INC     R1                      ; J                     2202
000150  020127  000025                   CMP     R1,#25                  ; J,*
000154  003767                           BLE     4$
000156  062702  000026                   ADD     #26,R2                  ; *,I                   2201
000162  020227  000232                   CMP     R2,#232                 ; I,*
000166  003761                           BLE     3$
000170  005002                           CLR     R2                      ; I                     2205
000172  010246                   5$:     MOV     R2,-(SP)                ; I,*                   2209
000174  012746  000041                   MOV     #41,-(SP)
000200  004737  000000G                  JSR     PC,BL$MUL
000204  005001                           CLR     R1                      ; J                     2208
000206  010003                   6$:     MOV     R0,R3                   ;                       2209
000210  060103                           ADD     R1,R3                   ; J,*
000212  006303                           ASL     R3
000214  005063  000000G                  CLR     ELOG.PKT(R3)
000220  005201                           INC     R1                      ; J                     2208
000222  020127  000040                   CMP     R1,#40                  ; J,*
000226  003767                           BLE     6$
000230  010216                           MOV     R2,(SP)                 ; I,*                   2211
000232  012746  000102                   MOV     #102,-(SP)
000236  004737  000000G                  JSR     PC,BL$MUL
000242  105060  000001G                  CLRB    ELOG.PKT+1(R0)
000246  062706  000006                   ADD     #6,SP                   ;                       2206
000252  005202                           INC     R2                      ; I                     2205
000254  020227  000014                   CMP     R2,#14                  ; I,*
000260  003744                           BLE     5$
000262  032737  000020  000000G          BIT     #20,SWP.FLAGS           ;                       2214
000270  001403                           BEQ     7$
000272  042737  000040  000000G          BIC     #40,SWP.FLAGS           ;                       2216
000300  032737  000002  000000G   7$:    BIT     #2,SWP.FLAGS            ;                       2218
000306  001403                           BEQ     8$
000310  042737  001000  000000G          BIC     #1000,SWP.FLAGS         ;                       2220
000316  132737  000001  000000G   8$:    BITB    #1,INIT.OCCURED         ;                       2222
000324  001402                           BEQ     9$
000326  000137  000732'                  JMP     15$
000332  017700  000000G           9$:    MOV     @FREE.MEM.ADDR,R0       ;                       2225
000336  006300                           ASL     R0
000340  063700  000000G                  ADD     FREE.MEM.ADDR,R0
000344  010037  001252'                  MOV     R0,DRS.START
000350  062737  000002  001252'          ADD     #2,DRS.START
000356  032737  000001  000000G          BIT     #1,SWP.FLAGS            ;                       2231
000364  001417                           BEQ     10$
```

H5


```
000366  112737  000001  001256'         MOVB    #1,APT.MODE
000374  013737  001252' 001260'         MOV     DRS.START,MAIL.BOX.TESTNUM      ;                       2234
000402  062737  000070  001260'         ADD     #70,MAIL.BOX.TESTNUM            ;                       2235
000410  013737  001252' 001262'         MOV     DRS.START,MAIL.BOX.SUBTST
000416  062737  000066  001262'         ADD     #66,MAIL.BOX.SUBTST             ;                       2236
000424  005037  000000G           10$:   CLR     NEX
000430  105037  000000G                   CLRB    CLK.PRESENT                     ;                       2240
000434  012746  000340                    MOV     #340,-(SP)                      ;                       2241
000440  012746  000000G                   MOV     #NEX.TRAP,-(SP)                 ;                       2242
000444  012746  000004                    MOV     #4,-(SP)
000450  012746  000003                    MOV     #3,-(SP)
000454  104437                           TRAP    37
000456  013701  177546                   MOV     @#177546,R1                     ; *,DUMMY.0
000462  005002                           CLR     R2                              ; DUMMY.1               2243
000464  012700  000004                    MOV     #4,R0                           ;                       2244
000470  104436                           TRAP    36                              ;                       2245
000472  032737  000001  000000G          BIT     #1,NEX                          ;                       2248
000500  001060
000502  112737  000001  000000G          MOVB    #1,CLK.PRESENT                  ;                       2251
000510  005037  000000G                   CLR     CLK.TICKS                       ;                       2252
000514  013716  000000G                   MOV     SWP.TIME,(SP)                   ;                       2253
000520  012746  000144                    MOV     #144,-(SP)
000524  004737  000000G                   JSR     PC,BL$DIV
000530  110037  000000G                   MOVB    R0,HOURS
000534  013716  000000G                   MOV     SWP.TIME,(SP)                   ;                       2254
000540  012746  000144                    MOV     #144,-(SP)
000544  004737  000000G                   JSR     PC,BL$MOD
000550  010003                           MOV     R0,R3
000552  005203                           INC     R3
000554  110337  000000G                   MOVB    R3,MINUTES
000560  123727  000000G 000074  11$:     CMPB    MINUTES,#74                     ;                       2256
000566  103412                           BLO     12$
000570  005000                           CLR     R0                              ;                       2258
000572  153700  000000G                   BISB    MINUTES,R0
000576  162700  000074                    SUB     #74,R0
000602  110037  000000G                   MOVB    R0,MINUTES
000606  105237  000000G                   INCB    HOURS                           ;                       2259
000612  000762                           BR      11$                             ;                       2256
000614  005016                   12$:    CLR     (SP)                            ;                       2262
000616  113716  000000G                   MOVB    HOURS,(SP)
000622  012746  000030                    MOV     #30,-(SP)
000626  004737  000000G                   JSR     PC,BL$MOD
000632  110037  000000G                   MOVB    R0,HOURS
000636  062706  000006                    ADD     #6,SP
000642  005037  000000G           13$:    CLR     NEX                             ;                       2250
000646  005037  000000G                   CLR     CSR.MEM                         ;                       2265
000652  012716  000340                    MOV     #340,(SP)                       ;                       2266
000656  012746  000000G                   MOV     #NEX.TRAP,-(SP)                 ;                       2267
000662  012746  000004                    MOV     #4,-(SP)
000666  012746  000003                    MOV     #3,-(SP)
000672  104437                           TRAP    37
000674  017701  000000G                   MOV     @CSR.ADD,R1                     ; *,DUMMY.0             2268
000700  005002                           CLR     R2                              ; DUMMY.1               2269
```

```
000702  012700  000004                  MOV     #4,R0                           ;                               2270
000706  104436                          TRAP    36
000710  032737  000001  000000G         BIT     #1,NEX                          ;                               2273
000716  001003                          BNE     14$
000720  012737  000001  000000G         MOV     #1,CSR.MEM                      ;                               2276
000726  062706  000016                  ADD     #16,SP                          ;                               2224
000732  132737  000001  000000G 15$:    BITB    #1,CLK.PRESENT                  ;                               2282
000740  001416                          BEQ     16$
000742  012746  000300                  MOV     #300,-(SP)                      ;                               2285
000746  012746  000000G                 MOV     #TIME,-(SP)
000752  012746  000100                  MOV     #100,-(SP)
000756  012746  000003                  MOV     #3,-(SP)
000762  104437                          TRAP    37
000764  012737  000100  177546          MOV     #100,@#177546                   ;                               2286
000772  062706  000010                  ADD     #10,SP                          ;                               2284
000776  032737  000001  000000G 16$:    BIT     #1,CSR.MEM                      ;                               2289
001004  001422                          BEQ     17$
001006  032737  000001  000000G         BIT     #1,TST.PAR
001014  001416                          BEQ     17$
001016  012746  000340                  MOV     #340,-(SP)                      ;                               2292
001022  012746  000000G                 MOV     #PARITY,-(SP)
001026  012746  000114                  MOV     #114,-(SP)
001032  012746  000003                  MOV     #3,-(SP)
001036  104437                          TRAP    37
001040  012777  000001  000000G         MOV     #1,@CSR.ADD                     ;                               2293
001046  062706  000010                  ADD     #10,SP                          ;                               2291
001052  104421          17$:    TRAP    21                                      ;                               2296
001054  010037  001266'                 MOV     R0,DRS.FLAGS
001060  042700  077777                  BIC     #77777,R0                       ;                               2298
001064  020027  100000                  CMP     R0,#-100000
001070  001003                          BNE     18$
001072  012700  000001                  MOV     #1,R0
001076  000401                          BR      19$
001100  005000          18$:    CLR     R0
001102  020027  100000          19$:    CMP     R0,#-100000
001106  001003                          BNE     20$
001110  112737  000001  000000G         MOVB    #1,HOE.FLAG                     ;                               2300
001116  004737  000000V         20$:    JSR     PC,INIT.TEST                    ;                               2303
001122  005002                          CLR     R2                              ; CTLR                          2305
001124  010246          21$:    MOV     R2,-(SP)                        ; CTLR,*                        2307
001126  012746  000126                  MOV     #126,-(SP)
001132  004737  000000G                 JSR     PC,BL$MUL
001136  022626                          CMP     (SP)+,(SP)+
001140  005760  000002G                 TST     CST+2(R0)
001144  100040                          BPL     25$
001146  010246                          MOV     R2,-(SP)                        ; CTLR,*                        2308
001150  012746  000022                  MOV     #22,-(SP)
001154  004737  000000G                 JSR     PC,BL$MUL
001160  022626                          CMP     (SP)+,(SP)+
001162  005760  000000G                 TST     DCT(R0)
001166  100027                          BPL     25$
001170  010246                          MOV     R2,-(SP)                        ; CTLR,*                        2313
001172  012746  000053                  MOV     #53,-(SP)
```

J5


```
001176  004737  000000G                 JSR     PC,BL$MUL
001202  012701  000003                  MOV     #3,R1                           ; *,OFFSET                      2311
001206  010003                  22$:     MOV     R0,R3                           ;                               2313
001210  060103                          ADD     R1,R3                           ; OFFSET,*
001212  006303                          ASL     R3
001214  032763  020000  000000G         BIT     #20000,CST(R3)
001222  001403                          BEQ     23$
001224  105037  000000G                 CLRB    EOP.FLAG                        ;                               2316
001230  000405                          BR      24$                             ;                               2315
001232  062701  000012          23$:     ADD     #12,R1                          ; *,OFFSET                      2311
001236  020127  000042                  CMP     R1,#42                          ; OFFSET,*
001242  003761                          BLE     22$
001244  022626                  24$:     CMP     (SP)+,(SP)+
001246  005202                  25$:     INC     R2                              ; CTLR                          2305
001250  000243                          .WORD CLV!CLC

001252  003724                          BLE     21$
001254  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                               2320
001262  001002                          BNE     26$
001264  004737  000000V                 JSR     PC,MULTI.DRIVE                  ;                               2322
001270  000207                  26$:     RTS     PC                              ;                               2167

; Routine Size:  349 words,     Routine Base:  $CODE$ + 0000
; Maximum stack depth per invocation:  13 words



                                        .SBTTL  T1 TEST SECTION
000000  004737  000000'         T1::
000000                          1$:     JSR     PC,$T1                          ;                               2322
000004  104466                          TRAP    66
000006  006000                          ROR     R0
000010  103773                          BLO     1$
000012  000207                          RTS     PC

; Routine Size:  6 words,       Routine Base:  $CODE$ + 1272
; Maximum stack depth per invocation:  2 words
```

K5


```
1   %sbttl 'INITIALIZATION TEST ROUTINES'
1
1   GLOBAL routine INIT_TEST : novalue =
1
1   !+
1   !       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
1   !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
1   !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE TEST
1   !       OR THE DM EXERCISER.
1   !
1   !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
1   !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
1   !       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
1   !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
1   !       EACH DISK VIA ROUTINE "ACCESS". THE INITIALIZATION TEST IS DEEMED A
1   !       SUCCESS IF A BLOCK OF EACH DISK CAN BE ACCESSED.
1   !-
1
2       begin
2       DRIVER_INIT ();                                 ! INIT DRIVER DATA AND DEVICES
2
2       incr CTLR from 0 to (MAX_CTLR - 1) do           ! FOR EACH CONTROLLER
3           begin
3           SET_CPAR (.CTLR);                           ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
3
3           if .CST_ADDR [STATE] eql ONLINE             ! IF CONTROLLER IS STILL ALIVE
3           then                                        ! FOR EACH DISK
3
3               incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
3
3                   if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
3                      (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
4                      (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
3                   then
4                       begin
4                       SET_UPAR (.OFFSET);             ! SET UP UNIT-RELATED DATA ITEMS
5                       IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)          !ZZZ
4                          THEN ACCESS ();                                 !ZZZ
4                                                   !SKIP IF DUP CAUSED INIT ZZZ
4
3                       end;                            ! IF UNIT IS PRESENT AND ONLINE
3
2           end;                                        ! CONTROLLER LOOP
2
1       end;                                            ! ROUTINE INIT_TEST
```

```
                                        .SBTTL  INIT.TEST INITIALIZATION TEST ROUTINES
000000  004137  000000G                 INIT.TEST::
                                        JSR     R1,$SAVE2                       ;                       2326
000004  004737  000000V                 JSR     PC,DRIVER.INIT                  ;                       2343
000010  005002                          CLR     R2                              ; CTLR                  2345
000012  010246                  1$:     MOV     R2,-(SP)                        ; CTLR,*                2347
```

L5


```
000014  004737  000000G                        JSR     PC,SET.CPAR
000020  013700  000000G                        MOV     CST.ADDR,R0
000024  005760  000002                         TST     2(R0)                   ;                               2349
000030  100035                                 BPL     4$
000032  012701  000003                         MOV     #3,R1
000036  010100                         2$:     MOV     R1,R0                   ; *,OFFSET                      2352
000040  006300                                 ASL     R0                      ; OFFSET,*                      2354
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  032710  040000                         BIT     #40000,(R0)
000052  001417                                 BEQ     3$
000054  032710  020000                         BIT     #20000,(R0)             ;                               2355
000060  001414                                 BEQ     3$
000062  032710  010000                         BIT     #10000,(R0)             ;                               2356
000066  001011                                 BNE     3$
000070  010116                                 MOV     R1,(SP)
000072  004737  000000G                        JSR     PC,SET.UPAR             ; OFFSET,*                      2359
000076  032737  000002  000000G                BIT     #2,DUP.FLAGS
000104  001002                                 BNE     3$                      ;                               2360
000106  004737  000000V                        JSR     PC,ACCESS
000112  062701  000012                 3$:     ADD     #12,R1                  ;                               2361
000116  020127  000041                         CMP     R1,#41                  ; *,OFFSET                      2352
000122  003745                                 BLE     2$                      ; OFFSET,*
000124  005726                         4$:     TST     (SP)+
000126  005202                                 INC     R2                      ;                               2346
000130  000243                                 .WORD CLV!CLC                   ; CTLR                          2345

000132  003727                                 BLE     1$
000134  000207                                 RTS     PC                      ;                               2326

; Routine Size:  47 words,      Routine Base:  $CODE$ + 1306
; Maximum stack depth per invocation:  5 words
```

M5


```
;    2369  1     GLOBAL routine DRIVER_INIT : novalue =
;    2370  1
;    2371  1     !+
;    2372  1     !       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
;    2373  1     !       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
;    2374  1     !       DRIVER DATA, AND TO BRING EACH RDRX CONTROLLER AND UNIT (DISK)
;    2375  1     !       ONLINE.
;    2376  1     !-
;    2377  1
;    2378  2         begin
;    2379  2
;    2380  2         local
;    2381  2             PKT_ADDR;
;    2382  2
;    2383  2         PKT_ADDR = MSCP_PKT + 10;                                   ! ADDR (TEXT + 0) OF 1ST MSCP PKT
;    2384  2         NEXT_PKT_USE = 0;                                           ! NEXT PACKET TO ALLOCATE
;    2385  2
;    2386  2         incr COUNT from 0 to (PKT_CNT - 1) do                       ! FOR EACH MSCP PACKET
;    2387  3             begin
;    2388  3             PKT_USE [.COUNT] = -1;                                  ! MARK PACKET FREE
;    2389  3             MSCP_PKT [.COUNT, PKT_LO] = .PKT_ADDR;                  ! LOAD ADDR INTO BUFFER DESCRIPTOR
;    2390  3             MSCP_PKT [.COUNT, PKT_HI] = 0;
;    2391  3             MSCP_PKT [.COUNT, CONNID] = CID_DISK;                   ! SET CONNECTION ID TO MSCP ID
;    2392  3             PKT_ADDR = .PKT_ADDR + (PKT_LEN * 2);                   ! ADVANCE ADDR TO NEXT PACKET
;    2393  3             end;
;    2394  2
;    2395  2         incr CTLR from 0 to (MAX_CTLR - 1) do                       ! FOR EACH CONTROLLER
;    2396  2
;    2397  2             if .CST [.CTLR, IP_ADDR] neqa 0                         ! IF CONTROLLER IS PRESENT
;    2398  2             then
;    2399  3                 begin
;    2400  3                 SET_CPAR (.CTLR);                                   ! CURRENT CONTROLLER PARAMETERS
;    2401  3                 CURRENT_VECTOR = .CST_ADDR [VEC_ADDR];              ! CURRENT CONTROLLER'S VECTOR
;    2402  3                 BRLEVEL = .CST_ADDR [BR_LEV] + 5;                   ! SET CURRENT CONTROLLER'S BR LEVEL
;    2403  3                 CTLR_INIT ();                                       ! INIT DEVICE AND CTLR DATA
;    2404  3
;    2405  3                 if .DCT_ADDR [STAT] eql ONLINE                      ! IF CONTROLLER IS STILL ALIVE
;    2406  3                 then                                                ! FOR EACH DIAK UNIT
;    2407  3
;    2408  3                     incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    2409  3
;    2410  3                         if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and  ! IF UNIT EXISTS
;    2411 *4                             (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2412  3                         then
;    2413  4                             begin
;    2414  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_0] = %o'40';      ! BLANK DEVICE NAME
;    2415  4                             CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_1] = %o'40';      !
;    2416  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_2] = %o'40';      !
;    2417  4                             CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_3] = %o'40';      !
;    2418  4                             SET_UPAR (.OFFSET);                                     ! SET UP UNIT-RELATED DATA ITEMS
;    2419  4                             UNIT_INIT ();                                           ! BRING UNIT ONLINE
;    2420  3                             end;                                                    ! IF UNIT EXISTS
;    2421  3
```

N5


```
;   2422  2                 end;                                                    ! IF CONTROLLER IS PRESENT
;   2423  2
;   2424  1         end;                                                            ! ROUTINE DRIVER_INIT

                                                .SBTTL  DRIVER.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G                 DRIVER.INIT::
                                                JSR     R1,$SAVE3                       ;                                       2369
000004  012702  000012G                         MOV     #MSCP.PKT+12,R2                 ; *,PKT.ADDR                            2383
000010  105037  000000G                         CLRB    NEXT.PKT.USE                    ;                                       2384
000014  005001                                  CLR     R1                              ; COUNT                                 2386
000016  112761  000377  000000G         1$:     MOVB    #377,PKT.USE(R1)                ; *,*(COUNT)                            2388
000024  010146                                  MOV     R1,-(SP)                        ; COUNT,*                               2389
000026  012746  000106                          MOV     #106,-(SP)
000032  004737  000000G                         JSR     PC,BL$MUL
000036  010260  000000G                         MOV     R2,MSCP.PKT(R0)                 ; PKT.ADDR,*
000042  005060  000002G                         CLR     MSCP.PKT+2(R0)                  ;                                       2390
000046  105060  000011G                         CLRB    MSCP.PKT+11(R0)                 ;                                       2391
000052  062702  000106                          ADD     #106,R2                         ; *,PKT.ADDR                            2392
000056  022626                                  CMP     (SP)+,(SP)+                     ;                                       2387
000060  005201                                  INC     R1                              ; COUNT                                 2386
000062  020127  000013                          CMP     R1,#13                          ; COUNT,*
000066  003753                                  BLE     1$
000070  005003                                  CLR     R3                              ; CTLR                                  2395
000072  010346                          2$:     MOV     R3,-(SP)                        ; CTLR,*                                2397
000074  012746  000126                          MOV     #126,-(SP)
000100  004737  000000G                         JSR     PC,BL$MUL
000104  022626                                  CMP     (SP)+,(SP)+
000106  005760  000000G                         TST     CST(R0)
000112  001503                                  BEQ     6$
000114  010346                                  MOV     R3,-(SP)                        ; CTLR,*                                2400
000116  004737  000000G                         JSR     PC,SET.CPAR
000122  013700  000000G                         MOV     CST.ADDR,R0                     ;                                       2401
000126  016037  000002  001246'                 MOV     2(R0),CURRENT.VECTOR
000134  042737  177000  001246'                 BIC     #177000,CURRENT.VECTOR
000142  005016                                  CLR     (SP)                            ;                                       2402
000144  116016  000004                          MOVB    4(R0),(SP)
000150  012746  000005                          MOV     #5,-(SP)
000154  004737  000000G                         JSR     PC,BL$SHF
000160  010037  000000G                         MOV     R0,BRLEVEL
000164  004737  000000V                         JSR     PC,CTLR.INIT                    ;                                       2403
000170  005777  000000G                         TST     @DCT.ADDR                       ;                                       2405
000174  100051                                  BPL     5$
000176  012701  000003                          MOV     #3,R1                           ; *,OFFSET                              2408
000202  013702  000000G                 3$:     MOV     CST.ADDR,R2                     ;                                       2410
000206  010100                                  MOV     R1,R0                           ; OFFSET,*
000210  006300                                  ASL     R0
000212  060200                                  ADD     R2,R0
000214  032710  040000                          BIT     #40000,(R0)
000220  001432                                  BEQ     4$
000222  032710  010000                          BIT     #10000,(R0)                     ;                                       2411
000226  001027                                  BNE     4$
000230  010100                                  MOV     R1,R0                           ; OFFSET,*                              2414
```

B6


```
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)
000252  010100                                  MOV     R1,R0                   ;                               2415
000254  006300                                  ASL     R0                      ; OFFSET,*                      2416
000256  060200                                  ADD     R2,R0
000260  112760  000040  000014                  MOVB    #40,14(R0)
000266  112760  000040  000015                  MOVB    #40,15(R0)
000274  010116                                  MOV     R1,(SP)                 ;                               2417
000276  004737  000000G                         JSR     PC,SET.UPAR             ; OFFSET,*                      2418
000302  004737  000000V                         JSR     PC,UNIT.INIT
000306  062701  000012                  4$:     ADD     #12,R1                  ;                               2419
000312  020127  000041                          CMP     R1,#41                  ; *,OFFSET                      2408
000316  003731                                  BLE     3$                      ; OFFSET,*
000320  022626                          5$:     CMP     (SP)+,(SP)+
000322  005203                          6$:     INC     R3                      ;                               2399
000324  000243                                  .WORD CLV!CLC                   ; CTLR                          2395

000326  003661                                  BLE     2$
000330  000207                                  RTS     PC                      ;                               2369

; Routine Size:  109 words,    Routine Base:  $CODE$ + 1444
; Maximum stack depth per invocation:  7 words
```

C6


```
;   2425  1     GLOBAL routine CTLR_INIT : novalue =
;   2426  1
;   2427  1     !+
;   2428  1     !       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;   2429  1     !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
;   2430  1     !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;   2431  1     !
;   2432  1     !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;   2433  1     !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
;   2434  1     !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;   2435  1     !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;   2436  1     !               ADDRESS AND INTERRUPT REQUEST LEVEL,
;   2437  1     !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;   2438  1     !
;   2439  1     !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;   2440  1     !       DEVICE ARE DROPPED.
;   2441  1     !-
;   2442  1
;   2443  2         begin
;   2444  2
;   2445  2         local
;   2446  2             RESULT : byte;
;   2447  2
;   2448  2         INI_CTLR_DAT ();                                    ! INITIALIZE CONTROLLER DATA
;   2449  2     !ZZZ    SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4);
;   2450  2         SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], .BRLEVEL);     ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;   2451  2         DCT_ADDR [IG_INT] = TRUE;                           ! SET DEVICE'S ASSUMED VECTOR ADDRESS    ZZZ
;   2452  2         L$LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];        ! SET "IGNORE INTERRUPT" BIT
;   2453  2                                                             ! GET FIRST UNIT NUMBER OF CONTROLLER
;   2454  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)               ! (USED BY DRS FOR DEVICE-FATAL CTLR ERRORS)
;   2455  3     THEN                                                    !IF DUP  ZZZ
;   2456  3                                             !CAUSED INIT, SKIP THIS CODE   ZZZ
;   2457  3         if REG_EXIST () eql FAILURE                         ! REGISTER EXISTENCE TEST
;   2458  3         then
;   2459  3             begin
;   2460  3             DROP_CTLR (.CCTLR, DU_INIT);                    ! DROP ALL CONTROLLER'S UNITS
;   2461  3             return;
;   2462  3             end;
;   2463  3
;   2464  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)               !IF DUP  ZZZ
;   2465  3     THEN                                            !CAUSED INIT, SKIP THIS CODE   ZZZ
;   2466  3
;   2467  3         if VEC_BR_TEST () eql FAILURE                       ! VECTOR ADDR AND BR LEVEL TEST
;   2468  3         then
;   2469  3             begin
;   2470  3             DROP_CTLR (.CCTLR, DU_INIT);                    ! DROP ALL CONTROLLER'S UNITS
;   2471  3             return;
;   2472  3             end;
;   2473  3
;   2474  3         RESULT = HARD_INIT ();                              ! ATTEMPT HARD DEVICE INIT
;   2475  3         DCT_ADDR [IG_INT] = FALSE;                          ! CLAER "IGNORE INTERRUPT" BIT
;   2476  3
;   2477  3
```

D6


```
;  2478  2          if .RESULT eql SUCCESS                          ! IF HARD INIT WAS SUCCESSFUL
;  2479  2          then
;  2480  3              begin
;  2481  3              ADDR_VECT_OK = TRUE;                        ! ADDRESS/VECTOR TEST PASSED
;  2482  3              INI_RRING ();                               ! INITIALIZE RESPONSE RING
;  2483  3              WRT_RDRX (RCSA, RC_ALL, SA_GO);             ! SET "GO" BIT (START CTLR POLLING)
;  2484  3
;  2485  3              if SET_CTLR_CHAR () eql SUCCESS             ! SET CONTROLLER CHARACTERISTICS
;  2486  3              then
;  2487  4                  begin
;  2488  4                  DCT_ADDR [STAT] = ONLINE;               ! MARK CONTROLLER ONLINE IN "DRIVER"
;  2489  4                  CST_ADDR [STATE] = ONLINE;              ! MARK CONTROLLER ONLINE IN "PROGRAM"
;  2490  3                  end;
;  2491  3              end
;  2492  3
;  2493  2          else                                            ! HARD INIT FAILED
;  2494  3              begin
;  2495  3              DROP_CTLR (.CCTLR, DU_INIT);                ! DROP ALL CONTROLLER'S UNITS
;  2496  2              end;
;  2497  2
;  2498  1          end;                                            ! ROUTINE CTLR_INIT

                                .SBTTL  CTLR.INIT INITIALIZATION TEST ROUTINES
000000  010146          CTLR.INIT::
                                MOV     R1,-(SP)
000002  004737  000000V        JSR     PC,INI.CTLR.DAT             ;                   2425
000006  013746  000000G        MOV     BRLEVEL,-(SP)               ;                   2448
000012  013700  000000G        MOV     CCTLR,R0                    ;                   2450
000016  006300                  ASL     R0
000020  016046  000102'         MOV     INT.ADDR(R0),-(SP)
000024  013746  001246'         MOV     CURRENT.VECTOR,-(SP)
000030  012746  000003          MOV     #3,-(SP)
000034  104437                  TRAP    37
000036  052777  040000  000000G BIS     #40000,@DCT.ADDR            ;                   2451
000044  013700  000000G        MOV     CST.ADDR,R0                 ;                   2452
000050  016001  000006          MOV     6(R0),R1
000054  000301                  SWAB    R1
000056  042701  177760          BIC     #177760,R1
000062  010137  000000G        MOV     R1,L$LUN
000066  032737  000002  000000G BIT     #2,DUP.FLAGS                ;                   2454
000074  001025                  BNE     2$
000076  004737  000000V        JSR     PC,REG.EXIST                ;                   2457
000102  005700                  TST     R0
000104  001410                  BEQ     1$                          ;                   2460
000106  032737  000002  000000G BIT     #2,DUP.FLAGS                ;                   2464
000114  001015                  BNE     2$
000116  004737  000000V        JSR     PC,VEC.BR.TEST              ;                   2467
000122  005700                  TST     R0
000124  001011                  BNE     2$
000126  013716  000000G  1$:    MOV     CCTLR,(SP)                  ;                   2470
000132  012746  000002          MOV     #2,-(SP)
000136  004737  000000G        JSR     PC,DROP.CTLR
```

E6


```
000142  062706  000012                          ADD     #12,SP                          ;                               2471
000146  000453                                  BR      5$                              ;                               2469
000150  004737  000000V                 2$:     JSR     PC,HARD.INIT                    ;                               2474
000154  110001                                  MOVB    R0,R1                           ; *,RESULT
000156  042777  040000  000000G                BIC     #40000,@DCT.ADDR                ;                               2475
000164  120127  000001                          CMPB    R1,#1                           ; RESULT,*                      2478
000170  001031                                  BNE     3$
000172  112737  000001  000000G                MOVB    #1,ADDR.VECT.OK                 ;                               2481
000200  004737  000000V                         JSR     PC,INI.RRING                    ;                               2482
000204  012701  000001                          MOV     #1,R1                           ; *,RC.REG                      2483
000210  013700  000000G                         MOV     RDRX.ADDR,R0
000214  010160  000002                          MOV     R1,2(R0)                        ; RC.REG,*
000220  004737  000000V                         JSR     PC,SET.CTLR.CHAR                ;                               2485
000224  020027  000001                          CMP     R0,#1
000230  001020                                  BNE     +$
000232  052777  100000  000000G                BIS     #100000,@DCT.ADDR               ;                               2488
000240  013700  000000G                         MOV     CST.ADDR,R0                     ;                               2489
000244  052760  100000  000002                  BIS     #100000,2(R0)
000252  000407                                  BR      4$
000254  013716  000000G                 3$:     MOV     CCTLR,(SP)                      ;                               2478
000260  012746  000002                          MOV     #2,-(SP)                        ;                               2495
000264  004737  000000G                         JSR     PC,DROP.CTLR
000270  005726                                  TST     (SP)+
000272  062706  000010                 4$:     ADD     #10,SP                          ;                               2494
000276  012601                         5$:     MOV     (SP)+,R1                        ;                               2443
000300  000207                                  RTS     PC                              ;                               2425

; Routine Size:  97 words,      Routine Base:  $CODE$ + 1776
; Maximum stack depth per invocation:  7 words
```

F6


```
;  2499  1   GLOBAL routine INI_CTLR_DAT : novalue =
;  2500  1
;  2501  1   !+
;  2502  1   !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER-RELATED
;  2503  1   !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;  2504  1   !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;  2505  1   !
;  2506  1   !       IMPLICIT INPUTS:
;  2507  1   !               CCTLR - CURRENT CONTROLLER NUMBER
;  2508  1   !               DCT_ADDR - ADDRESS OF CURENT CONTROLLER'S DCT
;  2509  1   !-
;  2510  1
;  2511  2       begin
;  2512  2       DCT_ADDR [WORD0] = 0;                                          ! CLEAR FIRST DCT WORD
;  2513  2       DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;  2514  2       DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4); ! LAST SLOT IN RESPONSE RING
;  2515  2       DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                    ! START OF COMMAND RING
;  2516  2       DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4); ! LAST SLOT IN COMMAND RING
;  2517  2       DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG];                       ! FIRST RRING SLOT TO POLL
;  2518  2       DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;  2519  1       end;


                                        .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G                 INI.CTLR.DAT::
                                        JSR     R1,$SAVE2                       ;                       2499
000004  013701  000000G                 MOV     DCT,ADDR,R1                     ;                       2512
000010  005011                          CLR     (R1)
000012  012702  000004                  MOV     #4,R2                           ;                       2513
000016  060102                          ADD     R1,R2
000020  013746  000000G                 MOV     CCTLR,-(SP)
000024  012746  000050                  MOV     #50,-(SP)
000030  004737  000000G                 JSR     PC,BL$MUL
000034  062700  000010'                 ADD     #COMM.AREA+10,R0
000040  010012                          MOV     R0,(R2)
000042  010061  000006                  MOV     R0,6(R1)                        ;                       2514
000046  062761  000014  000006          ADD     #14,6(R1)
000054  012700  000010                  MOV     #10,R0                          ;                       2515
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)                     ;                       2516
000076  062761  000014  000012          ADD     #14,12(R1)
000104  011261  000014                  MOV     (R2),14(R1)                     ;                       2517
000110  011061  000020                  MOV     (R0),20(R1)                     ;                       2518
000114  011061  000016                  MOV     (R0),16(R1)
000120  022626                          CMP     (SP)+,(SP)+                     ;                       2511
000122  000207                          RTS     PC                              ;                       2499

; Routine Size:  42 words,      Routine Base:  $CODE$ + 2300
; Maximum stack depth per invocation:  6 words
```

G6


```
;    2520  1     GLOBAL routine REG_EXIST =
;    2521  1     !+
;    2522  1     !       THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
;    2523  1     !       THE PRESENCE OF AN RDRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
;    2524  1     !       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
;    2525  1     !       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
;    2526  1     !       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
;    2527  1     !       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
;    2528  1     !       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
;    2529  1     !-
;    2530  2         begin
;    2531  2
;    2532  2         local
;    2533  2             DUMMY_0 : word,                         ! TEMP FOR READING SA AND IP
;    2534  2             DUMMY_1 : word;                         !
;    2535  2
;    2536  2         if .ENTRY_REASON eql NEW_PASS
;    2537  2         then
;    2538  2             return SUCCESS;                         ! SKIP TEST FOR NEXT PASS
;    2539  2
;    2540  2         OF_RC = 2;                                  ! SET UP TO READ SA FIRST
;    2541  2
;    2542  2         do
;    2543  3             begin
;    2544  3             NEX = FALSE;                            ! SET TO "TRAP NOT RECEIVED"
;    2545  3             SETVEC (4, NEX_TRAP, PRI07);            ! SET LOCATION 4 TRAP VECTOR ADDRESS
;    2546  3             DUMMY_0 = .(.RDRX_ADDR + .OF_RC);       ! READ REGISTER (THEN TRAP OR CONTINUE)
;    2547  3             DUMMY_1 = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
;    2548  3                                                     !       (TRAP RETURNS TO NEXT INSTRUCTION)
;    2549  3             CLRVEC (4);                             ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
;    2550  3
;    2551  3             if .NEX                                 ! IF NEX TRAP OCCURRED
;    2552  3             then
;    2553  4                 begin
;    2554  4                 C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2555  4
;    2556  4                 if .APT_MODE
;    2557  4                 then
;    2558  5                     begin
;    2559  5                     .MAIL_BOX_TESTNUM = 1;
;    2560  5                     .MAIL_BOX_SUBTST = 0;
;    2561  4                     end;
;    2562  4
;    2563  4                 ERRDF (10, EGD_10, EMS_10);         ! REGISTER EXISTENCE TEST FAILED
;    2564  4                 SETPRI (PRI00);                     ! LOWER PRIORITY
;    2565  4                 return FAILURE;
;    2566  4                 end
;    2567  3             else
;    2568  3                 OF_RC = .OF_RC - 2;                 ! SET UP FOR IP REG OR QUIT
;    2569  3
;    2570  3             end
;    2571  2         until .OF_RC lss 0;
;    2572  2
```

H6


```
;   2573  2          return SUCCESS;
;   2574  1          end;


                                        .SBTTL  REG.EXIST INITIALIZATION TEST ROUTINES
000000  004137  000000G                REG.EXIST::
                                        JSR     R1,$SAVE2
000004  123727  000000G 000005                  CMPB    ENTRY.REASON,#5                 ;                       2520
000012  001472                                  BEQ     4$                              ;                       2536
000014  012737  000002  000000G                MOV     #2,OF.RC                        ;                       2538
000022  005037  000000G              1$:       CLR     NEX                             ;                       2540
000026  012746  000340                        MOV     #340,-(SP)                      ;                       2544
000032  012746  000000G                        MOV     #NEX.TRAP,-(SP)                 ;                       2545
000036  012746  000004                        MOV     #4,-(SP)
000042  012746  000003                        MOV     #3,-(SP)
000046  104437                                TRAP    37
000050  013700  000000G                        MOV     RDRX.ADDR,R0                    ;                       2546
000054  063700  000000G                        ADD     OF.RC,R0
000060  011001                                MOV     (R0),R1                         ; #,DUMMY.0
000062  005002                                CLR     R2                              ; DUMMY.1
000064  012700  000004                        MOV     #4,R0                           ;                       2547
000070  104436                                TRAP    36                              ;                       2549
000072  032737  000001  000000G               BIT     #1,NEX                          ;                       2551
000100  001427                                BEQ     3$
000102  013700  000000G                        MOV     CCTLR,R0                        ;                       2554
000106  006300                                ASL     R0
000110  105260  000000G                        INCB    C.ERR.TBL(R0)
000114  032737  000001  001256'               BIT     #1,APT.MODE                     ;                       2556
000122  001405                                BEQ     2$
000124  012777  000001  001260'               MOV     #1,@MAIL.BOX.TESTNUM            ;                       2559
000132  005077  001262'                       CLR     @MAIL.BOX.SUBTST                ;                       2560
000136  104455                      2$:       TRAP    55                              ;                       2563
000140  000012                                .WORD   12
000142  000000G                               .WORD   EGD.10
000144  000000G                               .WORD   EMS.10
000146  005000                                CLR     R0                              ;                       2564
000150  104441                                TRAP    41
000152  062706  000010                        ADD     #10,SP                          ;                       2565
000156  000413                                BR      5$                              ;                       2553
000160  162737  000002  000000G       3$:       SUB     #2,OF.RC                      ;                       2568
000166  062706  000010                        ADD     #10,SP                          ;                       2543
000172  005737  000000G                       TST     OF.RC                           ;                       2571
000176  002311                                BGE     1$
000200  012700  000001              4$:       MOV     #1,R0                           ;                       2530
000204  000207                                RTS     PC
000206  005000                      5$:       CLR     R0                              ;                       2520
000210  000207                                RTS     PC

; Routine Size:  69 words,      Routine Base:  $CODE$ + 2424
; Maximum stack depth per invocation:  9 words
```

```
;    2575  1     GLOBAL routine VEC_BR_TEST =
;    2576  1
;    2577  1     !+
;    2578  1     !       THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RDRX VECTOR ADDRESS GIVEN
;    2579  1     !       BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;    2580  1     !       USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;    2581  1     !       OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;    2582  1     !       AN RDRX INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;    2583  1     !       THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;    2584  1     !       "FUNCTIONAL TEST STARTED" IS PRINTED BEFORE THE TEST, AND
;    2585  1     !       "EXERCISER STARTED" IS PRINTED AT ITS SUCCESSFUL CONCLUSION. IF
;    2586  1     !       EITHER "FUNCTIONAL TEST ..." OR "EXERCISER ..." DOES NOT APPEAR, THEN
;    2587  1     !       PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP IS LIKELY TO OCCUR. AT
;    2588  1     !       THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;    2589  1     !
;    2590  1     !       IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;    2591  1     !       PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;    2592  1     !       USER. A FORCED INTERRUPT SHOULD NOT OCCURR. THEN, BY LOWERING THE
;    2593  1     !       PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;    2594  1     !-
;    2595  1
;    2596  2         begin
;    2597  2
;    2598  2         if .ENTRY_REASON eql NEW_PASS
;    2599  2         then
;    2600  3             begin
;    2601  3             SETPRI (PRI00);                         ! LOWER PRIORITY
;    2602  3             return SUCCESS;                         ! SKIP TEST IF NEXT PASS
;    2603  2             end;
;    2604  2
;    2605  2         PRINTF (MSG_02);                            ! "FUNCTIONAL TEST STARTED"
;    2606  2
;    2607  2             if INT_GEN () eql FALSE                 ! FORCE AN INTERRUPT
;    2608  2             then
;    2609  3              begin                                  ! IF INTERRUPT DID NOT OCCUR
;    2610  3              C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2611  3
;    2612  3              if .APT_MODE
;    2613  3              then
;    2614  4                  begin
;    2615  4                  .MAIL_BOX_TESTNUM = 1;
;    2616  4                  .MAIL_BOX_SUBTST = 0;
;    2617  3                  end;
;    2618  3
;    2619  3              ERRDF (11, EGD_11, 0);                 ! VECTOR TEST FAILED
;    2620  3              return FAILURE;
;    2621  3              end
;    2622  2         else
;    2623  3             begin                                  ! INTERRUPT DID OCCUR
;    2624  3            PRINTF (MSG_03);                ! "EXERCISER STARTED"
;    2625  3             SETPRI (.BRLEVEL);                     ! SET PRIORITY TO ASSUMED BR LEVEL
;    2626  3
;    2627  3              if INT_GEN () eql FALSE                ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
```

J6


```
;  2628  3              then
;  2629  4                  begin                           ! IF INTERRUPT DID NOT OCCUR
;  2630  4                  SETPRI (.BRLEVEL - %o'40');     ! LOWER PRIORITY BY 1
;  2631  4                  DELAY (1);                      ! WAIT
;  2632  4                  BREAK;                          ! MMM
;  2633  4
;  2634  4                  if .DCT_ADDR [SA_SAVE] neq 0    ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON-ZERO)
;  2635  4                  then
;  2636  5                      begin
;  2637  5                      SETPRI (PRI00);             ! RESTORE PROCESSOR PRIORITY TO 0
;  2638  5                      return SUCCESS;             ! ONLY SUCCESSFUL EXIT POINT
;  2639  4                      end;
;  2640  4
;  2641  3                  end;
;  2642  3
;  2643  2              end;
;  2644  2
;  2645  2          SETPRI (PRI00);                         ! COME HERE ONLY FOR BR TEST FAILURE
;  2646  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  2647  2
;  2648  2          if .APT_MODE
;  2649  2          then
;  2650  3              begin
;  2651  3              .MAIL_BOX_TESTNUM = 1;
;  2652  3              .MAIL_BOX_SUBTST = 0;
;  2653  2              end;
;  2654  2
;  2655  2          ERRDF (12, EGD_12, EMS_12);
;  2656  2          return FAILURE;
;  2657  1          end;


                                .GLOBL  L$DLY

                                .SBTTL  VEC.BR.TEST INITIALIZATION TEST ROUTINES
000000  010146          VEC.BR.TEST::
                        MOV     R1,-(SP)                        ;                       2575
000002  005746          TST     -(SP)
000004  123727  000000G 000005 CMPB    ENTRY.REASON,#5                 ;               2598
000012  001003          BNE     1$
000014  005000          CLR     R0                              ;                       2601
000016  104441          TRAP    41
000020  000505          BR      8$                              ;                       2600
000022  012746  000000G 1$:     MOV     #MSG,02,-(SP)           ;                       2605
000026  012746  000001          MOV     #1,-(SP)
000032  010600                  MOV     SP,R0                   ; SP,*
000034  104417                  TRAP    17
000036  004737  000000V         JSR     PC,INT.GEN              ;                       2607
000042  005700                  TST     R0
000044  001023                  BNE     3$
000046  013700  000000G         MOV     CCTLR,R0                ;                       2610
000052  006300                  ASL     R0
```

K6


```
000054  105260  000000G                        INCB    C.ERR.TBL(R0)
000060  032737  000001  001256'                BIT     #1,APT.MODE                     ;                               2612
000066  001405                                  BEQ     2$
000070  012777  000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               2615
000076  005077  001262'                        CLR     @MAIL.BOX.SUBTST                ;                               2616
000102  104455                          2$:   TRAP    55                              ;                               2619
000104  000013                                .WORD   13
000106  000000G                               .WORD   EGD.11
000110  000000                                .WORD   0
000112  000500                                BR      11$
000114  012716  000000G                 3$:   MOV     #MSG.03,(SP)                    ;                               2620
000120  012746  000001                        MOV     #1,-(SP)                        ;                               2624
000124  010600                                MOV     SP,R0                           ; SP,*
000126  104417                                TRAP    17
000130  013700  000000G                        MOV     BRLEVEL,R0                      ;                               2625
000134  104441                                TRAP    41
000136  004737  000000V                        JSR     PC,INT.GEN                      ;                               2627
000142  005700                                TST     R0
000144  001036                                BNE     9$
000146  013700  000000G                        MOV     BRLEVEL,R0                      ;                               2630
000152  162700  000040                        SUB     #40,R0
000156  104441                                TRAP    41
000160  012701  000001                        MOV     #1,R1                           ; *,$$TMP2                      2631
000164  001411                          4$:   BEQ     7$
000166  013700  000000G                        MOV     L$DLY,R0                        ; *,$$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  104422                          7$:   TRAP    22
000212  013700  000000G                        MOV     DCT.ADDR,R0                     ;                               2634
000216  005760  000002                        TST     2(R0)
000222  001407                                BEQ     9$
000224  005000                                CLR     R0                              ;                               2637
000226  104441                                TRAP    41
000230  062706  000006                        ADD     #6,SP                           ;                               2638
000234  012700  000001                  8$:   MOV     #1,R0                           ;                               2636
000240  000427                                BR      12$
000242  005726                          9$:   TST     (SP)+                           ;                               2623
000244  005000                                CLR     R0                              ;                               2645
000246  104441                                TRAP    41
000250  013700  000000G                        MOV     CCTLR,R0                        ;                               2646
000254  006300                                ASL     R0
000256  105260  000000G                        INCB    C.ERR.TBL(R0)
000262  032737  000001  001256'                BIT     #1,APT.MODE                     ;                               2648
000270  001405                                BEQ     10$
000272  012777  000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               2651
000300  005077  001262'                        CLR     @MAIL.BOX.SUBTST                ;                               2652
000304  104455                          10$:  TRAP    55                              ;                               2655
000306  000014                                .WORD   14
000310  000000G                               .WORD   EGD.12
```

L6


```
000312  000000G                         .WORD   EMS,12
000314  022626                  11$:    CMP     (SP)+,(SP)+                     ;                       2656
000316  005000                          CLR     R0                              ;                       2575
000320  005726                  12$:    TST     (SP)+
000322  012601                          MOV     (SP)+,R1
000324  000207                          RTS     PC

; Routine Size:  107 words,    Routine Base:  $CODE$ + 2636
; Maximum stack depth per invocation:  7 words
```

M6


```
;    2658  1     GLOBAL routine INT_GEN =
;    2659  1
;    2660  1     !+
;    2661  1     !       THIS ROUTINE BEGINS AN RDRX INITIALIZATION SEQUENCE, BUT ONLY
;    2662  1     !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RDRX
;    2663  1     !       INTERRUPT (AT THE COMPLETEION OF STEP 1) IN ORDER TO HELP VERIFY THE
;    2664  1     !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;    2665  1     !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;    2666  1     !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;    2667  1     !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;    2668  1     !-
;    2669  1
;    2670  2         begin
;    2671  2
;    2672  2         local
;    2673  2             SA : word;                                  ! STORAGE FOR STEP 1 READ AND WRITE
;    2674  2
;    2675  2         DCT_ADDR [SA_SAVE] = 0;                         ! ZERO OUT SA SAVE WORD IN DCT
;    2676  2         WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;    2677  2         DELAY (2);                                      ! WAIT
;    2678  2         BREAK;                                          ! MMM
;    2679  2         INCR COUNT FROM 1 TO 500 DO                     !MAKE SURE WE GET INTO STEP 1  ZZZ
;    2680  3             BEGIN                                       !BEFORE STEP 1 WRITE           ZZZ
;    2681  3             SA = .RDRX_ADDR [RCSA, RC_ALL];             ! STEP 1 READ
;    2682  3               IF (.SA AND S1_MASK) EQL SA_S1            !DID WE GET THE S1 BIT?        ZZZ
;    2683  3               THEN                                      !                              ZZZ
;    2684  3               EXITLOOP;                                 !EXIT IF SO                    ZZZ
;    2685  3             DELAY (1);                                  !                              ZZZ
;    2686  3             BREAK;                                      !                              MMM
;    2687  3             END;                                        !                              ZZZ
;    2688  2
;    2689  2         SA = (WR_RING ! 8) or (.CURRENT_VECTOR ! -2) or SA_INT;     ! STEP 1 WRITE VALUE
;    2690  2          WRT_RDRX (RCSA, RC_ALL, .SA);                              ! STEP 1 WRITE
;    2691  2
;    2692  2          incr COUNT from 1 to 8000 do
;    2693  3              begin
;    2694  3              DELAY (1);                                 ! TOTAL DELAY COUNT OF 8,000
;    2695  3
;    2696  3              if .DCT_ADDR [SA_SAVE] neq 0               ! IF SA WAS CHANGED
;    2697  3              then
;    2698  3                  return TRUE;                           ! INTERRUPT OCCURED
;    2699  3
;    2700  3              BREAK;
;    2701  3              end;
;    2702  2
;    2703  2          return FALSE;                                  ! IF INTERRUPT DID NOT OCCUR
;    2704  1          end;

                                .SBTTL  INT.GEN INITIALIZATION TEST ROUTINES
000000  004137  000000G  INT.GEN::
                            JSR     R1,$SAVE3
000004  024646            CMP     -(SP),-(SP)                     ;                       2658
```

N6


```
000006 013700 000000G                  MOV     DCT.ADDR,R0                     ;                                       2675
000012 005060 000002                   CLR     2(R0)
000016 012700 177777                   MOV     #-1,R0                          ; *,RC.REG                              2676
000022 010077 000000G                  MOV     R0,@RDRX.ADDR                   ; RC.REG,*
000026 012701 000002                   MOV     #2,R1                           ; *,$$TMP2                              2677
000032 001411                  1$:     BEQ     4$
000034 013700 000000G                  MOV     L$DLY,R0                        ; *,$$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                              ; $$TMP2
000054 000766                          BR      1$
000056 104422                  4$:     TRAP    22
000060 012703 000764                   MOV     #764,R3                         ; *,COUNT                               2679
000064 013700 000000G          5$:     MOV     RDRX,ADDR,R0                    ;                                       2681
000070 016016 000002                   MOV     2(R0),(SP)                      ; *,RC.REG
000074 011602                          MOV     (SP),R2                         ; RC.REG,SA
000076 010200                          MOV     R2,R0                           ; SA,*                                  2682
000100 042700 001777                   BIC     #1777,R0
000104 020027 004000                   CMP     R0,#4000
000110 001417                          BEQ     10$                             ;                                       2684
000112 012701 000001                   MOV     #1,R1                           ; *,$$TMP2                              2685
000116 001411                  6$:     BEQ     9$
000120 013700 000000G                  MOV     L$DLY,R0                        ; *,$$TMP1
000124 001404                          BEQ     8$
000126 005066 000002           7$:     CLR     2(SP)                           ; $$TMP
000132 005300                          DEC     R0                              ; $$TMP1
000134 001374                          BNE     7$
000136 005301                  8$:     DEC     R1                              ; $$TMP2
000140 000766                          BR      6$
000142 104422                  9$:     TRAP    22
000144 005303                          DEC     R3                              ; COUNT                                 2679
000146 001346                          BNE     5$
000150 013700 001246'          10$:    MOV     CURRENT.VECTOR,R0               ;                                       2689
000154 006200                          ASR     R0
000156 006200                          ASR     R0
000160 010002                          MOV     R0,R2                           ; *,SA
000162 052702 111200                   BIS     #111200,R2                      ; *,SA
000166 010201                          MOV     R2,R1                           ; SA,RC.REG                             2690
000170 013700 000000G                  MOV     RDRX,ADDR,R0
000174 010160 000002                   MOV     R1,2(R0)                        ; RC.REG,*
000200 012702 017500                   MOV     #17500,R2                       ; *,COUNT                               2692
000204 012701 000001           11$:    MOV     #1,R1                           ; *,$$TMP2                              2694
000210 001411                  12$:    BEQ     15$
000212 013700 000000G                  MOV     L$DLY,R0                        ; *,$$TMP1
000216 001404                          BEQ     14$
000220 005066 000002           13$:    CLR     2(SP)                           ; $$TMP
000224 005300                          DEC     R0                              ; $$TMP1
000226 001374                          BNE     13$
000230 005301                  14$:    DEC     R1                              ; $$TMP2
000232 000766                          BR      12$
000234 013700 000000G          15$:    MOV     DCT.ADDR,R0                     ;                                       2696
```

B7


```
000240  005760  000002                        TST     2(R0)
000244  001403                                BEQ     16$
000246  012700  000001                        MOV     #1,R0
000252  000404                                BR      17$                             ;                               2698
000254  104422                        16$:    TRAP    22
000256  005302                                DEC     R2                              ; COUNT                         2692
000260  001351                                BNE     11$
000262  005000                                CLR     R0                              ;                               2670
000264  022626                        17$:    CMP     (SP)+,(SP)+                     ;                               2658
000266  000207                                RTS     PC

; Routine Size:  92 words,      Routine Base:  $CODE$ + 3164
; Maximum stack depth per invocation:  8 words
```

```
;  2705  1     GLOBAL routine HARD_INIT =
;  2706  1
;  2707  1     !+
;  2708  1     !       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;  2709  1     !       INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;  2710  1     !       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;  2711  1     !       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;  2712  1     !       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;  2713  1     !-
;  2714  1
;  2715  2         begin
;  2716  2
;  2717  2         local
;  2718  2             IE_VEC : word;                              ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
;  2719  2             !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;  2720  2
;  2721  2         IE_VEC = .CURRENT_VECTOR ^ -2;                  ! GET VECTOR ADDR/4 (IE = 0)
;  2722  2
;  2723  2         incr ATTEMPTS from 1 to INI_ATT do
;  2724  2             begin
;  2725  2
;  2726  2             label
;  2727  2                 STEP_1_READ,
;  2728  2                 STEP_2_READ,
;  2729  2                 STEP_3_READ,
;  2730  2                 STEP_4_READ;
;  2731  3
;  2732  3             WRT_RDRX (RCIP, RC_ALL, ALL_ONES);          ! WRITE IP TO START INIT SEQUENCE
;  2733  3     !
;  2734  3     !       STEP 1 READ
;  2735  3     !
;  2736  3             STEP = 1;
;  2737  3             STEP_1_READ:
;  2738  4                 begin
;  2739  4
;  2740  4                 incr COUNT from 1 to 500 do
;  2741  5                     begin
;  2742  5                     DELAY (1);                          ! TOTAL DELAY COUNT OF 500 FOR STEP 1
;  2743  5                     SA_REG = .RDRX_ADDR (RCSA, RC_ALL); ! READ SA
;  2744  5
;  2745  5                     if (.SA_REG and S1_MASK) eql SA_S1  ! IF STEP 1 READ IS O.K.
;  2746  5                     then
;  2747  5                         leave STEP_1_READ;
;  2748  5
;  2749  5                     BREAK;
;  2750  4                     end;
;  2751  4
;  2752  4                 exitloop;
;  2753  3                 end;
;  2754  3
;  2755  3     !
;  2756  3     !       STEP 1 WRITE
;  2757  3     !
```

```
;  2758  3               SA_REG = (WR_RING ^ 8) or .IE_VEC;                      ! STEP 1 WRITE VALUE
;  2759  3               WRT_RDRX (RCSA, RC_ALL, .SA_REG);                       ! STEP 1 WRITE
;  2760  3       !
;  2761  3       !       STEP 2 READ
;  2762  3       !
;  2763  3               STEP = .STEP + 1;
;  2764  3               STEP_2_READ:
;  2765  4                   begin
;  2766  4
;  2767  4                   incr COUNT from 1 to 10000 do
;  2768  5                       begin
;  2769  5                       DELAY (1);                                      ! TOTAL DELAY COUNT OF 10,000 FOR STEP 2
;  2770  5                       SA_REG = .RDRX_ADDR [RCSA, RC_ALL];             ! READ SA
;  2771  5
;  2772  6                       if (.SA_REG and S2_MASK) eql (SA_S2 or WR_RING) ! IF STEP 2 READ IS O.K.
;  2773  6                       then
;  2774  6                           leave STEP_2_READ;
;  2775  5
;  2776  5                       BREAK;
;  2777  4                       end;
;  2778  4
;  2779  4                   exitloop;
;  2780  3                   end;
;  2781  3
;  2782  3       !
;  2783  3       !       STEP 2 WRITE
;  2784  3       !
;  2785  3               WRT_RDRX (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]);            ! RINGBASE-LO, PI = 0
;  2786  3       !
;  2787  3       !       STEP 3 READ
;  2788  3       !
;  2789  3               STEP = .STEP + 1;
;  2790  3               STEP_3_READ:
;  2791  4                   begin
;  2792  4
;  2793  4                   incr COUNT from 1 to 10000 do
;  2794  5                       begin
;  2795  5                       DELAY (1);                                      ! TOTAL DELAY COUNT OF 10,000 FOR STEP 3 READ
;  2796  5                       SA_REG = .RDRX_ADDR [RCSA, RC_ALL];             ! READ SA
;  2797  5
;  2798  6                       if (.SA_REG and S3_MASK) eql (SA_S3 or .IE_VEC) ! IF STEP 3 READ IS O.K.
;  2799  5                       then
;  2800  5                           leave STEP_3_READ;
;  2801  5
;  2802  5                       BREAK;
;  2803  4                       end;
;  2804  4
;  2805  4                   exitloop;
;  2806  3                   end;
;  2807  3
;  2808  3       !
;  2809  3       !       STEP 3 WRITE
;  2810  3       !
```

```
;    2811  3          WRT_RDRX (RCSA, RC_ALL, 0);                          ! PP, RINGBASE-HI = 0
;    2812  3      !
;    2813  3      !   STEP 4 READ
;    2814  3      !
;    2815  3          STEP = .STEP + 1;
;    2816  3          STEP_4_READ:
;    2817  4              begin
;    2818  4
;    2819  4              incr COUNT from 1 to 10000 do
;    2820  5                  begin
;    2821  5                  DELAY (1);                                   ! TOTAL DELAY COUNT OF 10,000 FOR STEP 4 READ
;    2822  5                  SA_REG = .RDRX_ADDR [RCSA, RC_ALL];          ! READ SA
;    2823  5
;    2824  5                  if (.SA_REG and S4_MASK) eql SA_S4           ! IF STEP 4 READ IS O.K.
;    2825  5                  then
;    2826  6                      leave STEP_4_READ;
;    2827  5
;    2828  5                  BREAK;
;    2829  4                  end;
;    2830  4
;    2831  4              exitloop;
;    2832  3              end;
;    2833  3      !
;    2834  3      !   STEP 4 WRITE
;    2835  3      !
;    2836  3          CREDIT_BAL = 1;                                      ! START WITH A CREDIT BALANCE = 1
;    2837  3          WRT_RDRX (RCSA, RC_ALL, 0);                          ! BURST, LF, GO = 0
;    2838  3          return SUCCESS;                                      ! SUCCESS EXIT POINT
;    2839  3
;    2840  2          end;                                                 ! TRY AGAIN OR GIVE UP
;    2841  2
;    2842  2          CREDIT_BAL = 0;                                      ! NO CREDIT BALANCE
;    2843  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2844  2
;    2845  2          if .APT_MODE
;    2846  2          then
;    2847  3              begin
;    2848  3              .MAIL_BOX_TESTNUM = 1;
;    2849  3              .MAIL_BOX_SUBTST = 0;
;    2850  2              end;
;    2851  2
;    2852  2          ERRDF (13, EGD_13, EMS_13);                          ! INIT SEQUENCE FAILED
;    2853  2          return FAILURE;
;    2854  1          end;                                                 ! ROUTINE HARD_INIT

                                    .SBTTL  HARD.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G       HARD.INIT::
                                    JSR     R1,$SAVE5                    ;                                    2705
000004  162706  000012              SUB     #12,SP
000010  013704  001246'              MOV     CURRENT.VECTOR,R4            ; *,IE.VEC                           2721
000014  006204                       ASR     R4                           ; IE.VEC
000016  006204                       ASR     R4                           ; IE.VEC
```

F7


```
000020  012705  000002                        MOV     #2,R5                           ; *,ATTEMPTS                    2723
000024  012700  177777                        MOV     #-1,R0                          ; *,RC.REG                      2732
000030  010077  000000G                       MOV     R0,@RDRX.ADDR                   ; RC.REG,*
000034  012737  000001  000000G               MOV     #1,STEP                         ;                               2736
000042  012702  000764                        MOV     #764,R2                         ; *,COUNT                       2740
000046  012701  000001                1$:     MOV     #1,R1                           ; *,$$TMP2                      2742
000052  001411                        2$:     BEQ     5$
000054  013700  000000G                       MOV     L$DLY,R0                        ; *,$$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                              ; $$TMP2
000074  000766                                BR      2$
000076  013700  000000G                5$:     MOV     RDRX,ADDR,R0                    ;                               2743
000102  016016  000002                        MOV     2(R0),(SP)                      ; *,RC.REG
000106  011637  000000G                       MOV     (SP),$A.REG                     ; RC.REG,*
000112  011600                                MOV     (SP),R0                         ; SA.REG,*
000114  042700  001777                        BIC     #1777,R0                        ;                               2745
000120  020027  004000                        CMP     R0,#4000
000124  001404                                BEQ     6$
000126  104422                                TRAP    22                              ;                               2747
000130  005302                                DEC     R2                              ; COUNT                         2740
000132  001345                                BNE     1$
000134  000532                                BR      18$                             ;                               2724
000136  010437  000000G                6$:     MOV     R4,SA.REG                       ; IE.VEC,*                      2758
000142  052737  111000  000000G               BIS     #111000,SA.REG
000150  013701  000000G                       MOV     SA.REG,R1                       ; *,RC.REG                      2759
000154  013700  000000G                       MOV     RDRX.ADDR,R0
000160  010160  000002                        MOV     R1,2(R0)                        ; RC.REG,*
000164  005237  000000G                       INC     STEP                            ;                               2763
000170  012702  023420                        MOV     #23420,R2                       ; *,COUNT                       2767
000174  012701  000001                7$:     MOV     #1,R1                           ; *,$$TMP2                      2769
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$:    MOV     RDRX,ADDR,R0                    ;                               2770
000230  016066  000002  000002                MOV     2(R0),2(SP)                     ; *,RC.REG
000236  016637  000002  000000G               MOV     2(SP),SA.REG                    ; RC.REG,*
000244  016600  000002                        MOV     2(SP),R0                        ; SA.REG,*
000250  042700  003400                        BIC     #3400,R0                        ;                               2772
000254  020027  010222                        CMP     R0,#10222
000260  001404                                BEQ     12$
000262  104422                                TRAP    22                              ;                               2774
000264  005302                                DEC     R2                              ; COUNT                         2767
000266  001342                                BNE     7$
000270  000537                                BR      26$                             ;                               2724
000272  013700  000000G                12$:    MOV     DCT.ADDR,R0                     ;                               2785
```

```
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                    ;                                       2789
000316  010403                                MOV     R4,R3                   ; IE.VEC,*                              2798
000320  052703  020000                        BIS     #20000,R3
000324  012702  023420                        MOV     #23420,R2               ; *,COUNT                               2793
000330  012701  000001              13$:      MOV     #1,R1                   ; *,$$TMP2                              2795
000334  001411                      14$:      BEQ     17$
000336  013700  000000G                       MOV     L$DLY,R0                ; *,$$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                      ; $$TMP2
000356  000766                                BR      14$
000360  013700  000000G             17$:      MOV     RDRX.ADDR,R0            ;                                       2796
000364  016066  000002  000004               MOV     2(R0),4(SP)             ; *,RC.REG
000372  016637  000004  000000G               MOV     4(SP),SA.REG            ; RC.REG,*
000400  016600  000004                        MOV     4(SP),R0                ; SA.REG,*                              2798
000404  042700  003400                        BIC     #3400,R0
000410  020003                                CMP     R0,R3
000412  001404                                BEQ     19$
000414  104422                                TRAP    22                      ;                                       2800
000416  005302                                DEC     R2
000420  001343                                BNE     13$                     ; COUNT                                 2793
000422  000462                      18$:      BR      26$
000424  013700  000000G             19$:      MOV     RDRX.ADDR,R0            ;                                       2724
000430  005060  000002                        CLR     2(R0)                   ;                                       2811
000434  005237  000000G                       INC     STEP
000440  012703  023420                        MOV     #23420,R3               ;                                       2815
000444  012701  000001              20$:      MOV     #1,R1                   ; *,COUNT                               2819
000450  001411                      21$:      BEQ     24$                     ; *,$$TMP2                              2821
000452  013700  000000G                       MOV     L$DLY,R0
000456  001404                                BEQ     23$                     ; *,$$TMP1
000460  005066  000010              22$:      CLR     10(SP)
000464  005300                                DEC     R0                      ; $$TMP
000466  001374                                BNE     22$                     ; $$TMP1
000470  005301                      23$:      DEC     R1
000472  000766                                BR      21$                     ; $$TMP2
000474  013700  000000G             24$:      MOV     RDRX.ADDR,R0            ;                                       2822
000500  016066  000002  000006               MOV     2(R0),6(SP)             ; *,RC.REG
000506  016637  000006  000000G               MOV     6(SP),SA.REG            ; RC.REG,*
000514  016600  000006                        MOV     6(SP),R0                ; SA.REG,*                              2824
000520  042700  003777                        BIC     #3777,R0
000524  020027  040000                        CMP     R0,#40000
000530  001404                                BEQ     25$
000532  104422                                TRAP    22                      ;                                       2826
000534  005303                                DEC     R3
000536  001342                                BNE     20$                     ; COUNT                                 2819
000540  000413                                BR      26$
000542  012737  000001  000000G     25$:      MOV     #1,CREDIT.BAL           ;                                       2724
000550  005001                                CLR     R1                      ;                                       2836
                                                                              ; RC.REG                                2837
```

H7


```
000552  013700  000000G                        MOV     RDRX,ADDR,R0
000556  005060  000002                         CLR     2(R0)
000562  012700  000001                         MOV     #1,R0
000566  000425                                 BR      28$                             ;                       2724
000570  005037  000000G             26$:       CLR     CREDIT.BAL                      ;                       2842
000574  013700  000000G                        MOV     CCTLR,R0                        ;                       2843
000600  006300                                 ASL     R0                                                      --
000602  105260  000000G                        INCB    C.ERR.TBL(R0)
000606  032737  000001  001256'                BIT     #1,APT.MODE                     ;                       2845
000614  001405                                 BEQ     27$
000616  012777  000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM            ;                       2848
000624  005077  001262'                        CLR     @MAIL.BOX.SUBTST                ;                       2849
000630  104455                     27$:       TRAP    55                              ;                       2852
000632  000015                                 .WORD   15
000634  000000G                                .WORD   EGD.13
000636  000000G                                .WORD   EMS.13
000640  005000                                 CLR     R0
000642  062706  000012             28$:       ADD     #12,SP                          ;                       2715
000646  000207                                 RTS     PC                              ;                       2705

; Routine Size:  212 words,    Routine Base:  $CODE$ + 3454
; Maximum stack depth per invocation:  13 words
```

I7


```
1   GLOBAL routine INI_RRING : novalue =
1
1   !+
1   !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
1   !       FILL AN RDRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
1   !       (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
1   !       RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
1   !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
1   !       CONTROLLER-OWNED.
1   !
1   !       IMPLICIT INPUTS:
1   !               CCTLR - CURRENT CONTROLLER NUMBER
1   !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
1   !-
1
2       begin
2
2       local
2           index : word,
2           RRING_ADDR;
2
2       RRING_ADDR = .DCT_ADDR [RR_BEG];                    ! FIRST RESPONSE RING SLOT
2
2       incr COUNT from 1 to RRING_LEN do
3           begin
3           index = GET_PKT (.CCTLR);                       ! GET AN MSCP PACKET
3           .RRING_ADDR = .MSCP_PKT [.index, PKT_LO];       ! LOAD LO-ORDER BUFF DESC INTO SLOT
3           RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO SECOND WORD
3           .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];       ! LOAD HI-ORDER BUFF DESC INTO SLOT
3           PKT_USE [.index] = .CCTLR;                      ! PACKET IN USE
3           .RRING_ADDR = ..RRING_ADDR or ED_OWN or ED_FLAG;! GIVE OWNERSHIP TO CONTRLLER
3           RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO NEXT SLOT
2           end;
2
1       end;
```

```
                                        .SBTTL  INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G                 INI.RRING::
                                        JSR     R1,$SAVE4                       ;                       2855
000004  013700  000000G                 MOV     DCT.ADDR,R0                     ;                       2876
000010  016001  000004                  MOV     4(R0),R1                        ; *,RRING.ADDR
000014  013703  000000G                 MOV     CCTLR,R3                        ;                       2880
000020  012704  000004                  MOV     #4,R4                           ; *,COUNT               2878
000024  010346                  1$:     MOV     R3,-(SP)                        ;                       2880
000026  004737  000000G                 JSR     PC,GET.PKT
000032  010002                          MOV     R0,R2                           ; *,INDEX
000034  010216                          MOV     R2,(SP)                         ; INDEX,*               2881
000036  012746  000106                  MOV     #106,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  016021  000000G                 MOV     MSCP.PKT(R0),(R1)+              ; *,RRING.ADDR
000052  016011  000002G                 MOV     MSCP.PKT+2(R0),(R1)             ; *,RRING.ADDR           2883
000056  013703  000000G                 MOV     CCTLR,R3                        ;                       2884
```

J7


```
000062  110362  000000G                 MOVB    R3,PKT.USE(R2)          ; *,*(INDEX)
000066  052721  140000                  BIS     #140000,(R1)+           ; *,RRING.ADDR                  2885
000072  022626                          CMP     (SP)+,(SP)+             ;                               2879
000074  005304                          DEC     R4                      ; COUNT                         2878
000076  001352                          BNE     1$                                                      2855
000100  000207                          RTS     PC

; Routine Size:  33 words,      Routine Base:  $CODE$ + 4324
; Maximum stack depth per invocation:  8 words
```

K7


```
;    2890  1     GLOBAL routine SET_CTLR_CHAR =
;    2891  1
;    2892  1     !+
;    2893  1     !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RDRX HAS BEEN HARD-
;    2894  1     !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;    2895  1     !       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
;    2896  1     !
;    2897  1     !       IMPLICIT INPUTS:
;    2898  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2899  1     !-
;    2900  1
;    2901  2         begin
;    2902  2
;    2903  2         local
;    2904  2             P_INDEX : word;
;    2905  2
;    2906  2
;    2907  2     ! MISCELLANEOUS INITIALIZATION                                          !ZZZ
;    2908  2                                                                            !ZZZ
;    2909  2     QIO [.CCTLR] = 0;                   !INIT NO OF OUTSTANDING QIOS       !ZZZ
;    2910  2     CST [.CCTLR, U_CNT] = 0;            !CLEAR UNITS IN CST TABLE          !ZZZ
;    2911  2     INCR COUNT FROM 0 TO (RP_CNT - 1) DO    !INIT RETURN PACKET POOL       !ZZZ
;    2912  2          RP_USE [.COUNT] = -1;                                             !ZZZ
;    2913  2                                                                            !ZZZ
;    2914  2     IODQ_IN = IODQ_OUT = 0;             !INIT I/O DONE QUEUE POINTERS      !ZZZ
;    2915  2
;    2916  2
;    2917  2         P_INDEX = GET_PKT (.CCTLR);                                    ! GET AN MSCP PACKET
;    2918  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                          ! PACKET SIZE
;    2919  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                          ! OPCODE = SET CTLR CHAR
;    2920  2         MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                        ! CONTROLLER FLAGS
;    2921  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = IMM_CMD;                       ! IMMEDIATE COMMAND
;    2922  2
;    2923  2         if SEND (.P_INDEX) eql FAILURE                                 ! ATTEMPT SEND
;    2924  2         then
;    2925  3             begin                                                      ! IF SEND WAS UNSUCCESSFUL
;    2926  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2927  3
;    2928  3             if .APT_MODE
;    2929  3             then
;    2930  4                 begin
;    2931  4                 .MAIL_BOX_TESTNUM = 1;
;    2932  4                 .MAIL_BOX_SUBTST = 0;
;    2933  3                 end;
;    2934  3
;    2935  3             ERRDF (20, EGD_20, 0);                                 ! FATAL ERROR
;    2936  3             PUT_PKT (.P_INDEX);                                    ! RETURN PACKET TO POOL
;    2937  3             DROP_CTLR (.CCTLR, DU_CFATAL);                         ! DROP CONTROLLER
;    2938  3             return FAILURE;
;    2939  3             end
;    2940  2         else
;    2941  3             begin                                                  ! IF SEND WAS SUCCESSFUL
;    2942  3
```

```
;    2943  3                do
;    2944  4                    begin
;    2945  4                    WAIT ();                                        ! WAIT FOR RETPKT RESPONSE
;    2946  4                    RP_INDX = OUT_IODQ ();                          ! GET INDEX OF RETPKT
;    2947  4                    RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;    2948  4
;    2949  4                    if .RP_ADDR [MESTYP] neq MT_SEQ                 ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2950  4                    then
;    2951  4                        PUT_RETPKT (.RP_INDX);
;    2952  4
;    2953  4                    end
;    2954  3                until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    2955  4                      ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    2956  3                       ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2957  3
;    2958  3                if .RP_ADDR [CONID] eql CID_DRIVER                  ! IF RETPKT IS FROM "DRIVER"
;    2959  3                then
;    2960  4                    begin
;    2961  4                    PRINTF (DBM23);                                 ! "ERROR IN SET CTLR CHAR"
;    2962  4                    PUT_RETPKT (.RP_INDX);                          ! RELEASE RETURN PACKET
;    2963  4                    DR_ERR ();                                      ! DROP CONTROLLER
;    2964  4                    return FAILURE;
;    2965  4                    end
;    2966  3                else
;    2967  4                    begin                                           ! ELSE - RETPKT IS FROM DISK MSCP
;    2968  4
;    2969  4                    if (.RP_ADDR [ENDCOD] neq (OP_SCC or OP_END)) or    ! IF WRONG ENDCODE
;    2970  5                       ((.RP_ADDR [C_FLGS] and CF_MASK) neq CF_MASK)    ! OR FLAGS IN ERROR
;    2971  4                    then
;    2972  5                        begin
;    2973  5                        C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2974  5
;    2975  5                        if .APT_MODE
;    2976  5                        then
;    2977  6                            begin
;    2978  6                            .MAIL_BOX_TESTNUM = 1;
;    2979  6                            .MAIL_BOX_SUBTST = 0;
;    2980  5                            end;
;    2981  5
;    2982  5                        ERRDF (21, EGD_21, EMS_21);                 ! FATAL ERROR
;    2983  5                        DROP_CTLR (.CCTLR, DU_CFATAL);              ! DROP CONTROLLER
;    2984  5                        PUT_RETPKT (.RP_INDX);                      ! RELEASE RETURN PACKET
;    2985  5                        return FAILURE;
;    2986  5                        end
;    2987  4                    else
;    2988  5                        begin                                       ! RETPKT HAS CORRECT ENDCODE
;    2989  5                        CMD_TIME = .RP_ADDR [C_TIME] * 2;
;    2990  5
;    2991  5     !ZZZ               if BIT_TST (SWP_FLAGS, SWF_TRC)
;    2992  5     !ZZZ               then
;    2993  5     !ZZZ                   PRINTF (DBM25, .RP_ADDR [C_TIME]);
;    2994  5
;    2995  4                        end;                                        ! RETPKT HAS CORRECT ENDCODE
```

M7


```
;  2996  4
;  2997  3                  end;                                    ! IF RETPKT WAS SENT BY DISK MSCP
;  2998  3
;  2999  3              PUT_RETPKT (.RP_INDX);
;  3000  3              return SUCCESS;                             ! IF SEND WAS SUCCESSFUL
;  3001  2              end;
;  3002  2
;  3003  1          end;                                            ! ROUTINE SET_CTLR_CHAR


                                .SBTTL  SET.CTLR.CHAR INITIALIZATION TEST ROUTINES
000000  010146          SET.CTLR.CHAR::
                        MOV     R1,-(SP)                ;                                       2890
000002  013701  000000G MOV     CCTLR,R1                ;                                       2909
000006  105061  000000G CLRB    QIO(R1)
000012  010146          MOV     R1,-(SP)                ;                                       2910
000014  012746  000126  MOV     #126,-(SP)
000020  004737  000000G JSR     PC,BL$MUL
000024  105060  000005G CLRB    CST+5(R0)
000030  005000          CLR     R0                      ; COUNT                                 2911
000032  112760  000377  000000G 1$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)            2912
000040  005200          INC     R0                      ; COUNT                                 2911
000042  020027  000007  CMP     R0,#7                   ; COUNT,*
000046  003771          BLE     1$
000050  005037  000000G CLR     IODQ.OUT                ;                                       2914
000054  005037  000000G CLR     IODQ.IN
000060  010116          MOV     R1,(SP)                 ;                                       2917
000062  004737  000000G JSR     PC,GET.PKT
000066  010001          MOV     R0,R1                   ; *,P.INDEX
000070  010116          MOV     R1,(SP)                 ; P.INDEX,*                             2918
000072  012746  000106  MOV     #106,-(SP)
000076  004737  000000G JSR     PC,BL$MUL
000102  012760  000040  000006G MOV     #40,MSCP.PKT+6(R0)
000110  112760  000004  000022G MOVB    #4,MSCP.PKT+22(R0)      ;                               2919
000116  012760  000120  000030G MOV     #120,MSCP.PKT+30(R0)    ;                               2920
000124  105060  000004G CLRB    MSCP.PKT+4(R0)          ;                                       2921
000130  010116          MOV     R1,(SP)                 ; P.INDEX,*                             2923
000132  004737  000000G JSR     PC,SEND
000136  005700          TST     R0
000140  001036          BNE     3$
000142  013700  000000G MOV     CCTLR,R0                ;                                       2926
000146  006300          ASL     R0
000150  105260  000000G INCB    C.ERR.TBL(R0)
000154  032737  000001  001256' BIT     #1,APT.MODE             ;                               2928
000162  001405          BEQ     2$
000164  012777  000001  001260' MOV     #1,@MAIL.BOX.TESTNUM    ;                               2931
000172  005077  001262' CLR     @MAIL.BOX.SUBTST        ;                                       2932
000176  104455          TRAP    55                      ;                                       2935
000200  000024          .WORD   24
000202  000000G         .WORD   EGD.20
000204  000000          .WORD   0
000206  010116          MOV     R1,(SP)                 ; P.INDEX,*                             2936
000210  004737  000000G JSR     PC,PUT.PKT
```

N7


```
000214  013716  000000G                        MOV     CCTLR,(SP)                      ;                                       2937
000220  012746  000006                         MOV     #6,-(SP)
000224  004737  000000G                        JSR     PC,DROP.CTLR
000230  005726                                 TST     (SP)+
000232  005000                                 CLR     R0                              ;                                       2925
000234  000554                                 BR      12$                             ;                                       2941
000236  004737  000000G                3$:     JSR     PC,WAIT
000242  004737  000000G                        JSR     PC,OUT.IODQ                     ;                                       2945
000246  010037  000000G                        MOV     R0,RP.INDX                      ;                                       2946
000252  010016                                 MOV     R0,(SP)                         ; RP.INDX,*                             2947
000254  012746  000054                         MOV     #54,-(SP)
000260  004737  000000G                        JSR     PC,BL$MUL
000264  062700  000000G                        ADD     #RETPKT,R0
000270  010037  000000G                        MOV     R0,RP.ADDR
000274  132760  000360  000002                BITB    #360,2(R0)                      ;                                       2949
000302  001404                                 BEQ     4$
000304  013716  000000G                        MOV     RP.INDX,(SP)                    ;                                       2951
000310  004737  000000G                        JSR     PC,PUT.RETPKT
000314  005726                         4$:     TST     (SP)+                           ;                                       2944
000316  013701  000000G                        MOV     RP.ADDR,R1                      ;                                       2954
000322  005000                                 CLR     R0
000324  126127  000003  000003                CMPB    3(R1),#3
000332  001002                                 BNE     5$
000334  005200                                 INC     R0
000336  000407                                 BR      6$
000340  132761  000360  000002        5$:     BITB    #360,2(R1)                      ;                                       2955
000346  001333                                 BNE     3$
000350  105761  000014                         TSTB    14(R1)                          ;                                       2956
000354  100330                                 BPL     3$
000356  006000                         6$:     ROR     R0                              ;                                       2958
000360  103015                                 BCC     7$
000362  012716  000000G                        MOV     #DBM23,(SP)                     ;                                       2961
000366  012746  000001                         MOV     #1,-(SP)
000372  010600                                 MOV     SP,R0                           ; SP,*
000374  104417                                 TRAP    17
000376  013716  000000G                        MOV     RP.INDX,(SP)                    ;                                       2962
000402  004737  000000G                        JSR     PC,PUT.RETPKT
000406  004737  000000V                        JSR     PC,DR.ERR                       ;                                       2963
000412  000447                                 BR      10$                             ;                                       2964
000414  126127  000014  000204        7$:     CMPB    14(R1),#204                     ;                                       2969
000422  001007                                 BNE     8$
000424  016100  000022                         MOV     22(R1),R0                       ;                                       2970
000430  042700  177657                         BIC     #177657,R0
000434  020027  000120                         CMP     R0,#120
000440  001437                                 BEQ     11$
000442  013700  000000G                8$:     MOV     CCTLR,R0                        ;                                       2973
000446  006300                                 ASL     R0
000450  105260  000000G                        INCB    C.ERR.TBL(R0)
000454  032737  000001  001256'                BIT     #1,APT.MODE                     ;                                       2975
000462  001405                                 BEQ     9$
000464  012777  000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM            ;                                       2978
000472  005077  001262'                        CLR     @MAIL.BOX.SUBTST                ;                                       2979
000476  104455                         9$:     TRAP    55                              ;                                       2982
```

B8


```
000500  000025                                  .WORD   25
000502  000000G                                 .WORD   EGD.21
000504  000000G                                 .WORD   EMS.21
000506  013716  000000G                         MOV     CCTLR,(SP)
000512  012746  000006                          MOV     #6,-(SP)                ;                       2983
000516  004737  000000G                         JSR     PC,DROP.CTLR
000522  013716  000000G                         MOV     RP.INDX,(SP)
000526  004737  000000G                         JSR     PC,PUT.RETPKT           ;                       2984
000532  062706  000010                  10$:    ADD     #10,SP
000536  000416                                  BR      13$                     ;                       2985
000540  016137  000024  000000G       11$:    MOV     24(R1),CMD.TIME         ;                       2972
000546  006337  000000G                         ASL     CMD.TIME                ;                       2989
000552  013716  000000G                         MOV     RP.INDX,(SP)
000556  004737  000000G                         JSR     PC,PUT.RETPKT           ;                       2999
000562  012700  000001                          MOV     #1,R0
000566  062706  000006                  12$:    ADD     #6,SP                   ;                       2941
000572  000401                                  BR      14$                     ;                       2923
000574  005000                          13$:    CLR     R0                      ;                       2901
000576  012601                          14$:    MOV     (SP)+,R1                ;                       2890
000600  000207                                  RTS     PC

; Routine Size:  193 words,    Routine Base:  $CODE$ + 4426
; Maximum stack depth per invocation:  7 words
```

```
;    3004  1     routine UNIT_INIT : novalue =
;    3005  1
;    3006  1     !+
;    3007  1     !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
;    3008  1     !       (DISK) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
;    3009  1     !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
;    3010  1     !       MESSAGE, AND TO VERIFY THE RESPONSE.
;    3011  1     !
;    3012  1     !       IMPLICIT INPUTS:
;    3013  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3014  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    3015  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    3016  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3017  1     !-
;    3018  1
;    3019  2         begin
;    3020  2         local
;    3021  2             MAX0_LBNS : WORD UNSIGNED,              ! UNIT'S MAXIMUM LO WORD LBN
;    3022  2             MAX1_LBNS : WORD UNSIGNED;              ! UNIT'S MAXIMUM HI WORD LBN
;    3023  2
;    3024  2
;    3025  2         P_INDEX = GET_PKT (.CCTLR);                 ! GET AN MSCP PACKET
;    3026  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;       ! PACKET SIZE
;    3027  2         MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;       ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;    3028  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;       ! OPCODE FOR "ONLINE"
;    3029  2     !ZZZ MSCP_PKT [.P_INDEX, DDPAR] = BIT00;        ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;    3030  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = SEQ_CMD;    ! SEQUENTIAL COMMAND
;    3031  2
;    3032  2         if SEND (.P_INDEX) eql FAILURE              ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;    3033  2         then
;    3034  3             begin
;    3035  3             T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    3036  3
;    3037  3             if .APT_MODE                            !ZZZ
;    3038  3             then
;    3039  4                 begin
;    3040  4                 .MAIL_BOX_TESTNUM = 1;
;    3041  4                 .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    3042  3                 end;
;    3043  3
;    3044  3             CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;    3045  3             ERRDF (22, EGD_22, 0);                  !
;    3046  3             DUR [.L$LUN] = DU_ONLINE;               ! SETUP REASON TO DROP UNIT
;    3047  3             DODU (.L$LUN);                          ! DROP UNIT
;    3048  3             PUT_PKT (.P_INDEX);                     ! RETURN PACKET TO POOL
;    3049  3             end
;    3050  2         else
;    3051  3             begin                                   ! OTHERWISE (SEND WAS SUCCESSFUL)
;    3052  3
;    3053  3             do
;    3054  4                 begin
;    3055  4                 WAIT ();                            ! WAIT FOR RETPKT RESPONSE
;    3056  4                 RP_INDX = OUT_IODQ ();              ! GET INDEX OF RETPKT
```

```
;    3057  4                RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;    3058  4
;    3059  4                if .RP_ADDR [MESTYP] neq MT_SEQ     ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    3060  4                then
;    3061  4                    PUT_RETPKT (.RP_INDX);
;    3062  4
;    3063  4                end
;    3064  3            until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    3065  4                  ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    3066  3                   ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    3067  3
;    3068  3            if .RP_ADDR [CONID] eql CID_DRIVER      ! IF RETPKT IS FROM "DRIVER"
;    3069  3            then
;    3070  4                begin
;    3071  4                PRINTF (DBM26);                     ! "ERROR IN UNIT INIT"
;    3072  4                DR_ERR ();                          ! DROP CONTROLLER
;    3073  4                end
;    3074  3            else
;    3075  3
;    3076  4                if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
;    3077  3                then
;    3078  4                    begin
;    3079  4                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    3080  4
;    3081  4                    if .APT_MODE                    !ZZZ
;    3082  4                    then
;    3083  5                        begin
;    3084  5                        .MAIL_BOX_TESTNUM = 1;
;    3085  5                        .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    3086  4                        end;
;    3087  4
;    3088  4                    CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    3089  4                    ERRDF (23, EGD_23, EMS_21);     ! FATAL ERROR
;    3090  4                    DUR [.L$LUN] = DU_ONLINE;       ! SETUP REASON TO DROP UNIT
;    3091  4                    DODU (.L$LUN);                  ! DROP UNIT
;    3092  4                    end
;    3093  3                else
;    3094  4                    begin                           ! RETPKT HAS GOOD ENDCODE
;    3095  4                    ST_CODE = .RP_ADDR [STSCOD];    ! GET STATUS CODE
;    3096  4                    SB_CODE = .RP_ADDR [SUBCOD];    ! GET SUB-CODE
;    3097  4
;ZZ  3098  4                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_0] = .RP_ADDR [NAME_0] + %o'100';                  ! UNIT NAME Z
;    3099  4                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .RP_ADDR [NAME_1_HI] * 16;                    !ZZZ
;    3100  4                    CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] = .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] +    !ZZZ
;    3101  4                                                              .RP_ADDR [NAME_1_LO] + %o'100';               !ZZZ
;    3102  4                    CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_2] = .RP_ADDR [NAME_NUM] / 10 + %o'60';            !ZZZ
;    3103  4                    CST_ADDR [.CUOFF + OF_NAME_2, D_NAME_3] = (.RP_ADDR [NAME_NUM] mod 10) + %o'60';        !ZZZ
;    3104  4
;    3105  4
;    3106  4
;    3107  4                    IF .CST_ADDR [.CUOFF + OF_NAME_0, D_NAME_1] EQL %O'104'         !IF NAME IS _D          !ZZZ
;    3108  4                    THEN                                                                                    !ZZZ
;    3109  4                    CST_ADDR [.CUOFF, D_TYPE] = FIXED                               !ITS FIXED.             !ZZZ
```

```
4                   ELSE
4                   CST_ADDR [.CUOFF, D_TYPE] = REMOVABLE;                          !OTHERWISE REMOVABLE    !ZZZ
4                                                                                                           !ZZZ
4
4
4                   if .ST_CODE neq ST_SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
4                   then
5                       begin
5                       T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
5
5                       if .APT_MODE                !ZZZ
5                       then
6                           begin
6                           .MAIL_BOX_TESTNUM = 1;
6                           .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
5                           end;
5
5                       CST_ADDR [.CUOFF, D_FATAL] = TRUE;
5                       ERRDF (15, EGD_15, EMS_30); ! ONLINE FAILED
5                       DUR [.L$LUN] = DU_ONLINE;   ! SET UP REASON FOR DROPPING UNIT
5                       DODU (.L$LUN);              ! DROP UNIT
5                       end
4                   else
5                       begin                       ! SUCCESFUL OPERATION
5
5                       MAX0_LBNS = .RP_ADDR [SIZE0];                                   ! LOAD LOWER WORD OF UNIT SIZE
5                       MAX1_LBNS = .RP_ADDR [SIZE1];                                   ! LOAD UPPER WORD OF UNIT SIZE
5
6                       if (.MAX0_LBNS eql 0)                                           ! THIS SUBTRACTS ONE FROM THE TOTAL
5                       then                                                            ! BECAUSE EVERYTHING STARTS AT 0
6                           begin                                                       ! THROUGH (MAXIMUM - 1)
6                           MAX0_LBNS = %O'177777';
6                           MAX1_LBNS = .MAX1_LBNS - 1;
6                           end
5                       else
5                           MAX0_LBNS = .MAX0_LBNS - 1;
5
5                       if  (.CST_ADDR [.CUOFF + 2, D_BEG1] gtru .MAX1_LBNS) or                 ! THIS SECTION CHECKS TO SEE
5                           ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and                ! IN SOFTWARE QUESTIONS WERE
6                           (.CST_ADDR [.CUOFF + 1, D_BEG0] gtru (.MAX0_LBNS - 1)))              ! DEVICE SPECIFIED
6                                                                                           ! note 1 less then max. or diagnostic
5                       then                                                                ! operator error
6                           begin
6                           CST_ADDR [.CUOFF + 2, D_BEG1] = 0;
6                           CST_ADDR [.CUOFF + 1, D_BEG0] = 0;                          ! change beginning lbn to 0
5                           end;
5
5                       if
5                           (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or
5
6                           ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
6                           (.CST_ADDR [.CUOFF + 3, D_END0] gtru .MAX0_LBNS))
```

F8


```
;    3163  5                        then
;    3164  6                            begin
;    3165  6                            CST_ADDR [.CUOFF + 4, D_END1] = .MAX1_LBNS;
;    3166  6                            CST_ADDR [.CUOFF + 3, D_END0] = .MAX0_LBNS;                ! and ending lbn to max_lbn
;    3167  5                            end;
;    3168  5
;    3169  6!ZZZ                     if  (.CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] gtru              !MAKE SURE START ADDRESS
;    3170  5!ZZZ                          .CST_ADDR [.CUOFF + OF_END1, D_END1]) or               !IS NO LARGER THAN END ADDRESS
;    3171  5!ZZZ                                                                                 !
;    3172  7!ZZZ                         ((.CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] eqlu             !
;    3173  6!ZZZ                           .CST_ADDR [.CUOFF + OF_END1, D_END1]) and             !
;    3174  7!ZZZ                         (.CST_ADDR [.CUOFF + OF_BEG, D_BEG0] gtru               !
;    3175  6!ZZZ                          .CST_ADDR [.CUOFF + OF_END, D_END0] ))                 !
;    3176  6!ZZZ                                                                                 !
;    3177  5!ZZZ                     then                                                        !
;    3178  6 ZZZ                         begin                                                   !
;    3179  6 ZZZ                         CST_ADDR [.CUOFF + OF_BEG1, D_BEG1] = 0;                !IF IT IS, THEN
;    3180  6 ZZZ                         CST_ADDR [.CUOFF + OF_BEG, D_BEG0] = 0;                 ! change beginning lbn to 0
;    3181  5 ZZZ                         end;                                                    !
;    3182  5!ZZZ
;    3183  5
;    3184  7                             if (((.ENTRY_REASON eql RESTART) or                     ! if restart or
;    3185  6                                 (.ENTRY_REASON eql START)) and                      ! if continue
;    3186  6
;    3187  6                                 (.CRN_LOW leq 8) and                                ! and
;    3188  6                                 (.CRN_HIGH eql 0))                                  ! first initialization
;    3189  6
;    3190  5                             THEN                                                    ! intialize block numbers
;    3191  6                                 begin
;    3192  6                                 BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0];      ! LOAD sequential LBN table
;    3193  6                                 BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1];      !
;    3194  6                                 TRK_SGN [.L$LUN] = 1;                                       ! POSITIVE TRACKING DIRECTION
;    3195  5                                 end;
;    3196  5
;    3197  5
;    3198  5                                                                                     ! THIS SECTION LOADS TYPE INTO  CST TABLE
;TN  3199  5     !ZZZ                            selectoneu .RP_ADDR [R_MODEL] of                    ! MODEL BYTE TELLS WHAT TYPE OF UNIT
;    3200  5     !ZZZ                                                                            ! INDENTIFICATION BLOCK
;    3201  5     !ZZZ                                set
;    3202  5     !ZZZ
;    3203  5     !ZZZ                                [%O'6'] :   CST_ADDR [.CUOFF, D_TYPE] = RD_51;      ! RD 51
;    3204  5     !ZZZ                                [%O'7'] :   CST_ADDR [.CUOFF, D_TYPE] = RX_50;      ! RX 50
;    3205  5     !ZZZ                                [%O'10'] :  CST_ADDR [.CUOFF, D_TYPE] = RD_52;      ! RD 52
;    3206  5     !ZZZ
;    3207  5     !ZZZ                                [otherwise] : BEGIN
;    3208  5     !ZZZ                                              ERRDF (25 ,EGD_24 ,EMS_30);           ! ERROR UNKNOWN DEVICE
;    3209  5     !ZZZ                                              END;
;    3210  5     !ZZZ                                tes;
;    3211  5
;    3212  5
;    3213  5
;    3214  5                         if ((.RP_ADDR [U_FLGS] and UF_WPH) eql UF_WPH) and              ! STATUS CODE IS O.K.
;    3215  6                            (.CST_ADDR [.CUOFF, D_PROT] eql UNPROTECTED)
```

```
;    3216  5                        then
;    3217  6                            begin
;    3218  6                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    3219  6
;    3220  6                            if .APT_MODE                    !ZZZ
;    3221  6                            then
;    3222  7                                begin
;    3223  7                                .MAIL_BOX_TESTNUM = 1;
;    3224  7                                .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;    3225  6                                end;
;    3226  6
;    3227  6                            CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    3228  6                            ERRDF (16, EGD_16, EMS_30);                         ! WRITE-PROTECT CONFLICT
;    3229  6                            DUR [.L$LUN] = DU_PROTECT;                          ! SET REASON TO DROP UNIT
;    3230  6                            DODU (.L$LUN);                                      ! DROP UNIT
;    3231  6                            end
;    3232  5                        else
;    3233  6                            begin                                               ! WRITE PROTECT SWITCH IS O.K.
;    3234  6                            CST_ADDR [.CUOFF, D_STAT] = ONLINE;                 ! SET ONLINE FLAG
;    3235  6                            CST [.CCTLR, U_CNT] = .CST [.CCTLR, U_CNT] + 1;     ! ADD UNIT TO CTRL TABLE
;    3236  5                            end;
;    3237  4                        end;
;    3238  3                    end;                                    ! IF RETPKT HAS CORRECT ENDCODE
;    3239  3
;    3240  3                PUT_RETPKT (.RP_INDX);
;    3241  2                end;                                        ! IF SEND WAS SUCCESSFUL
;    3242  2
;    3243  1            end;                                            ! ROUTINE UNIT-INIT


                                    .SBTTL  UNIT.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G       UNIT.INIT:
                                    JSR     R1,$SAVE5                   ;                   3004
000004  005746                      TST     -(SP)
000006  013746  000000G             MOV     CCTLR,-(SP)                 ;                   3025
000012  004737  000000G             JSR     PC,GET.PKT
000016  010037  000000G             MOV     R0,P.INDEX
000022  010016                      MOV     R0,(SP)                     ; P.INDEX,*         3026
000024  012746  000106              MOV     #106,-(SP)
000030  004737  000000G             JSR     PC,BL$MUL
000034  012760  000044  000006G     MOV     #44,MSCP.PKT+6(R0)
000042  013760  000000G 000016G     MOV     CDISK,MSCP.PKT+16(R0)       ;                   3027
000050  112760  000011  000022G     MOVB    #11,MSCP.PKT+22(R0)         ;                   3028
000056  112760  000001  000004G     MOVB    #1,MSCP.PKT+4(R0)           ;                   3030
000064  013716  000000G             MOV     P.INDEX,(SP)                ;                   3032
000070  004737  000000G             JSR     PC,SEND
000074  005700                      TST     R0
000076  001054                      BNE     2$
000100  013700  000000G             MOV     T.ADDR,R0                   ;                   3035
000104  105260  000051              INCB    51(R0)
000110  032737  000001  001256'     BIT     #1,APT.MODE                 ;                   3037
000116  001415                      BEQ     1$
000120  012777  000001  001260'     MOV     #1,@MAIL.BOX.TESTNUM        ;                   3040
```

```
000126  013700  000000G                       MOV     CUOFF,R0                        ;                               3041
000132  006300                                  ASL     R0
000134  063700  000000G                       ADD     CST.ADDR,R0
000140  111077  001262'                       MOVB    (R0),@MAIL.BOX.SUBTST
000144  042777  177760  001262'               BIC     #177760,@MAIL.BOX.SUBTST
000152  013700  000000G               1$:     MOV     CUOFF,R0                        ;                               3044
000156  006300                                  ASL     R0
000160  063700  000000G                       ADD     CST.ADDR,R0
000164  052710  010000                        BIS     #10000,(R0)
000170  104455                                TRAP    55                              ;                               3045
000172  000026                                .WORD   26
000174  000000G                               .WORD   EGD.22
000176  000000                                .WORD   0
000200  013700  000000G                       MOV     L$LUN,R0                        ;                               3046
000204  112760  000007  000000G               MOVB    #7,DUR(R0)
000212  104451                                TRAP    51                              ;                               3047
000214  013716  000000G                       MOV     P.INDEX,(SP)                    ;                               3048
000220  004737  000000G                       JSR     PC,PUT.PKT
000224  000137  007164'                       JMP     28$                             ;                               3032
000230  004737  000000G               2$:     JSR     PC,WAIT                         ;                               3055
000234  004737  000000G                       JSR     PC,OUT.IODQ                     ;                               3056
000240  010037  000000G                       MOV     R0,RP.INDX
000244  010016                                MOV     R0,(SP)                         ; RP.INDX,*                     3057
000246  012746  000054                        MOV     #54,-(SP)
000252  004737  000000G                       JSR     PC,BL$MUL
000256  062700  000000G                       ADD     #RETPKT,R0
000262  010037  000000G                       MOV     R0,RP.ADDR
000266  132760  000360  000002               BITB    #360,2(R0)                      ;                               3059
000274  001404                                BEQ     3$
000276  013716  000000G                       MOV     RP.INDX,(SP)                    ;                               3061
000302  004737  000000G                       JSR     PC,PUT.RETPKT
000306  005726                        3$:     TST     (SP)+                           ;                               3054
000310  013702  000000G                       MOV     RP.ADDR,R2                      ;                               3064
000314  005000                                CLR     R0
000316  126227  000003  000003               CMPB    3(R2),#3
000324  001002                                BNE     4$
000326  005200                                INC     R0
000330  000407                                BR      5$
000332  132762  000360  000002       4$:     BITB    #360,2(R2)                      ;                               3065
000340  001333                                BNE     2$
000342  105762  000014                        TSTB    14(R2)                          ;                               3066
000346  100330                                BPL     2$
000350  006000                        5$:     ROR     R0                              ;                               3068
000352  103012                                BCC     6$
000354  012716  000000G                       MOV     #DBM26,(SP)                     ;                               3071
000360  012746  000001                        MOV     #1,-(SP)
000364  010600                                MOV     SP,R0                           ; SP,*
000366  104417                                TRAP    17
000370  004737  000000V                       JSR     PC,DR.ERR                       ;                               3072
000374  005726                                TST     (SP)+                           ;                               3070
000376  000456                                BR      8$                              ;                               3068
000400  013766  000000G 000004       6$:     MOV     CUOFF,4(SP)                     ;                               3088
000406  006366  000004                        ASL     4(SP)
```

```
000412  063766  000000G 000004                 ADD     CST.ADDR,4(SP)
000420  126227  000014  000211                 CMPB    14(R2),#211             ;                       3076
000426  001444                                 BEQ     9$
000430  013700  000000G                        MOV     T.ADDR,R0               ;                       3079
000434  105260  000050                         INCB    50(R0)
000440  032737  000001  001256'                BIT     #1,APT.MODE             ;                       3081
000446  001415                                 BEQ     7$
000450  012777  000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM    ;                       3084
000456  013700  000000G                        MOV     CUOFF,R0                ;                       3085
000462  006300                                 ASL     R0
000464  063700  000000G                        ADD     CST.ADDR,R0
000470  111077  001262'                        MOVB    (R0),@MAIL.BOX.SUBTST
000474  042777  177760  001262'                BIC     #177760,@MAIL.BOX.SUBTST
000502  052776  010000  000004        7$:     BIS     #10000,@4(SP)           ;                       3088
000510  104455                                 TRAP    55                      ;                       3089
000512  000027                                 .WORD   27
000514  000000G                                .WORD   EGD.23
000516  000000G                                .WORD   EMS.21
000520  013700  000000G                        MOV     L$LUN,R0
000524  112760  000007  000000G                MOVB    #7,DUR(R0)              ;                       3090
000532  104451                                 TRAP    51
000534  000137  007154'                8$:     JMP     27$                     ;                       3091
000540  116237  000016  000000G        9$:     MOVB    16(R2),ST.CODE          ;                       3076
000546  042737  177740  000000G                BIC     #177740,ST.CODE         ;                       3095
000554  016200  000016                         MOV     16(R2),R0
000560  006200                                 ASR     R0                      ;                       3096
000562  006200                                 ASR     R0
000564  006200                                 ASR     R0
000566  006200                                 ASR     R0
000570  006200                                 ASR     R0
000572  042700  174000                         BIC     #174000,R0
000576  010037  000000G                        MOV     R0,SB.CODE
000602  013701  000000G                        MOV     CUOFF,R1                ;                       3098
000606  006301                                 ASL     R1
000610  063701  000000G                        ADD     CST.ADDR,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               ;                       3099
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                      ;                       3100
000670  156300  000001                         BISB    1(R3),R0
000674  016201  000034                         MOV     34(R2),R1
000700  006201                                 ASR     R1
```

J8


```
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
000722  062701  000100                  ADD     #100,R1                         ;                               3101
000726  110163  000001                  MOVB    R1,1(R3)
000732  013701  000000G                 MOV     CUOFF,R1                        ;                               3102
000736  006301                          ASL     R1
000740  063701  000000G                 ADD     CST.ADDR,R1
000744  116216  000034                  MOVB    34(R2),(SP)
000750  042716  177700                  BIC     #177700,(SP)
000754  012746  000012                  MOV     #12,-(SP)
000760  004737  000000G                 JSR     PC,BL$DIV
000764  010004                          MOV     R0,R4
000766  062704  000060                  ADD     #60,R4
000772  110461  000014                  MOVB    R4,14(R1)
000776  116216  000034                  MOVB    34(R2),(SP)
001002  042716  177700                  BIC     #177700,(SP)                    ;                               3103
001006  012746  000012                  MOV     #12,-(SP)
001012  004737  000000G                 JSR     PC,BL$MOD
001016  010004                          MOV     R0,R4
001020  062704  000060                  ADD     #60,R4
001024  110461  000015                  MOVB    R4,15(R1)
001030  126327  000001  000104          CMPB    1(R3),#104                      ;                               3107
001036  001004                          BNE     10$
001040  152776  000020  000010          BISB    #20,@10(SP)                     ;                               3109
001046  000403                          BR      11$                             ;                               3107
001050  142776  000020  000010  10$:    BICB    #20,@10(SP)                     ;                               3111
001056  005737  000000G         11$:    TST     ST.CODE                         ;                               3115
001062  001440                          BEQ     13$
001064  013700  000000G                 MOV     T.ADDR,R0                       ;                               3118
001070  105260  000050                  INCB    50(R0)
001074  032737  000001  001256'         BIT     #1,APT.MODE                     ;                               3120
001102  001411                          BEQ     12$
001104  012777  000001  001260'         MOV     #1,@MAIL.BOX.TESTNUM            ;                               3123
001112  117677  000010  001262'         MOVB    @10(SP),@MAIL.BOX.SUBTST        ;                               3124
001120  042777  177760  001262'         BIC     #177760,@MAIL.BOX.SUBTST
001126  052776  010000  000010  12$:    BIS     #10000,@10(SP)                  ;                               3127
001134  104455                          TRAP    55                              ;                               3128
001136  000017                          .WORD   17
001140  000000G                         .WORD   EGD.15
001142  000000G                         .WORD   EMS.30
001144  013700  000000G                 MOV     L$LUN,R0                        ;                               3129
001150  112760  000007  000000G         MOVB    #7,DUR(R0)
001156  104451                          TRAP    51                              ;                               3130
001160  000137  007152'                 JMP     26$                             ;                               3115
001164  016203  000044         13$:    MOV     44(R2),R3                       ; *,MAX0.LBNS                   3135
001170  016204  000046                 MOV     46(R2),R4                       ; *,MAX1.LBNS                   3136
001174  005703                          TST     R3                              ; MAX0.LBNS                     3138
001176  001004                          BNE     14$
```

K8


```
001200  012703  177777                        MOV     #-1,R3                  ; *,MAX0.LBNS                   3141
001204  005304                                 DEC     R4                      ; MAX1.LBNS                     3142
001206  000401                                 BR      15$                     ;                               3138
001210  005303                          14$:   DEC     R3                      ; MAX0.LBNS                     3145
001212  013701  000000G                 15$:   MOV     CUOFF,R1                ;                               3147
001216  006301                                 ASL     R1
001220  063701  000000G                         ADD     CST.ADDR,R1
001224  012705  000004                         MOV     #4,R5
001230  060105                                 ADD     R1,R5
001232  021504                                 CMP     (R5),R4                 ; *,MAX1.LBNS
001234  101013                                 BHI     16$
001236  001022                                 BNE     17$                     ;                               3149
001240  013701  000000G                         MOV     CUOFF,R1                ;                               3150
001244  006301                                 ASL     R1
001246  063701  000000G                         ADD     CST.ADDR,R1
001252  010300                                 MOV     R3,R0                   ; MAX0.LBNS,*
001254  005300                                 DEC     R0
001256  026100  000002                         CMP     2(R1),R0
001262  101410                                 BLOS    17$
001264  005015                          16$:   CLR     (R5)                    ;                               3154
001266  013701  000000G                         MOV     CUOFF,R1                ;                               3155
001272  006301                                 ASL     R1
001274  063701  000000G                         ADD     CST.ADDR,R1
001300  005061  000002                         CLR     2(R1)
001304  013701  000000G                 17$:   MOV     CUOFF,R1                ;                               3159
001310  006301                                 ASL     R1
001312  063701  000000G                         ADD     CST.ADDR,R1
001316  012700  000010                         MOV     #10,R0
001322  060100                                 ADD     R1,R0
001324  021004                                 CMP     (R0),R4                 ; *,MAX1.LBNS
001326  101011                                 BHI     18$
001330  001020                                 BNE     19$                     ;                               3161
001332  013701  000000G                         MOV     CUOFF,R1                ;                               3162
001336  006301                                 ASL     R1
001340  063701  000000G                         ADD     CST.ADDR,R1
001344  026103  000006                         CMP     6(R1),R3                ; *,MAX0.LBNS
001350  101410                                 BLOS    19$
001352  010410                          18$:   MOV     R4,(R0)                 ; MAX1.LBNS,*                   3165
001354  013701  000000G                         MOV     CUOFF,R1                ;                               3166
001360  006301                                 ASL     R1
001362  063701  000000G                         ADD     CST.ADDR,R1
001366  010361  000006                         MOV     R3,6(R1)                ; MAX0.LBNS,*
001372  021510                          19$:   CMP     (R5),(R0)               ;                               3169
001374  101017                                 BHI     20$
001376  001026                                 BNE     21$                     ;                               3172
001400  013700  000000G                         MOV     CUOFF,R0                ;                               3174
001404  006300                                 ASL     R0
001406  063700  000000G                         ADD     CST.ADDR,R0
001412  013701  000000G                         MOV     CUOFF,R1                ;                               3175
001416  006301                                 ASL     R1
001420  063701  000000G                         ADD     CST.ADDR,R1
001424  026061  000002  000006                 CMP     2(R0),6(R1)             ;                               3174
001432  101410                                 BLOS    21$
```

```
001434 005015                          20$:  CLR     (R5)
001436 013701 000000G                        MOV     CUOFF,R1                        ;3179
001442 006301                                ASL     R1                              ;3180
001444 063701 000000G                        ADD     CST.ADDR,R1
001450 005061 000002                         CLR     2(R1)
001454 123727 000000G 000002           21$:  CMPB    ENTRY.REASON,#2                 ;3184
001462 001404                                BEQ     22$
001464 123727 000000G 000001                 CMPB    ENTRY.REASON,#1                 ;3185
001472 001031                                BNE     23$
001474 023727 000000G 000010           22$:  CMP     CRN.LOW,#10                     ;3187
001502 003025                                BGT     23$
001504 005737 000000G                        TST     CRN.HIGH                        ;3188
001510 001022                                BNE     23$
001512 013700 000000G                        MOV     L$LUN,R0                        ;3192
001516 010004                                MOV     R0,R4
001520 006304                                ASL     R4
001522 006304                                ASL     R4
001524 013701 000000G                        MOV     CUOFF,R1
001530 006301                                ASL     R1
001532 063701 000000G                        ADD     CST.ADDR,R1
001536 016164 000002  000000G                MOV     2(R1),BST(R4)
001544 011564 000002G                        MOV     (R5),BST+2(R4)
001550 112760 000001  000000G                MOVB    #1,TRK.SGN(R0)                  ;3193
001556 032762 020000  000022           23$:  BIT     #20000,22(R2)                   ;3194
001564 001442                                BEQ     25$                             ;3214
001566 005776 000010                         TST     @10(SP)                         ;3215
001572 100037                                BPL     25$
001574 013700 000000G                        MOV     T.ADDR,R0                       ;3218
001600 105260 000050                         INCB    50(R0)
001604 032737 000001  001256'                BIT     #1,APT.MODE                     ;3220
001612 001411                                BEQ     24$
001614 012777 000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM            ;3223
001622 117677 000010  001262'                MOVB    @10(SP),@MAIL.BOX.SUBTST        ;3224
001630 042777 177760  001262'                BIC     #177760,@MAIL.BOX.SUBTST
001636 052776 010000  000010           24$:  BIS     #10000,@10(SP)                  ;3227
001644 104455                                TRAP    55                              ;3228
001646 000020                                .WORD   20
001650 000000G                               .WORD   EGD.16
001652 000000G                               .WORD   EMS.30
001654 013700 000000G                        MOV     L$LUN,R0                        ;3229
001660 112760 000011  000000G                MOVB    #11.DUR(R0)
001666 104451                                TRAP    51                              ;3230
001670 000414                                BR      26$                             ;3214
001672 052776 020000  000010           25$:  BIS     #20000,@10(SP)                  ;3234
001700 013716 000000G                        MOV     CCTLR,(SP)                      ;3235
001704 012746 000126                         MOV     #126,-(SP)
001710 004737 000000G                        JSR     PC,BL$MUL
001714 105260 000005G                        INCB    CST+5(R0)
001720 005726                                TST     (SP)+
001722 022626                          26$:  CMP     (SP)+,(SP)+                     ;3233
001724 013716 000000G                  27$:  MOV     RP.INDX,(SP)                    ;3094
001730 004737 000000G                        JSR     PC,PUT.RETPKT                   ;3240
001734 062706 000006                   28$:  ADD     #6,SP                           ;3004
```

M8


```
001740  000207                                  RTS     PC

; Routine Size:  497 words,     Routine Base:  $CODE$ + 5230
; Maximum stack depth per invocation:  13 words
```

N8


```
;    3244  1     GLOBAL routine DR_ERR : novalue =
;    3245  1
;    3246  1     !+
;    3247  1     !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;    3248  1     !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;    3249  1     !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;    3250  1     !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;    3251  1     !       AS AN INITIALIZATION ERROR.
;    3252  1     !
;    3253  1     !       IMPLICIT INPUTS:
;    3254  1     !               RP_ADDR - ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;    3255  1     !                       (I.E., CONNECTION ID = CID_DRIVER)
;    3256  1     !-
;    3257  1
;    3258  2         begin
;    3259  2
;    3260  2         local
;    3261  2             REASON : word initial (DU_TIME);        ! ASSUME COMMAND TIMEOUT
;    3262  2
;    3263  2         if .RP_ADDR [MESTYP] eql MT_FATAL           ! IF FATAL DEVICE ERROR
;    3264  2         then
;    3265  2
;    3266  2         DROP_CTLR (.CCTLR, .REASON);                ! DROP ALL UNITS ON CONTROLLER
;    3267  1         end;


                                        .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                  DR.ERR::MOV     R1,-(SP)                        ;
000002  012701  000012                  MOV     #12,R1                          ; *,REASON                      3244
000006  013700  000000G                 MOV     RP.ADDR,R0                      ;                               3258
000012  116000  000002                  MOVB    2(R0),R0                                                        3263
000016  042700  177417                  BIC     #177417,R0
000022  020027  000060                  CMP     R0,#60
000026  001006                          BNE     1$
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                               3266
000034  010146                          MOV     R1,-(SP)                        ; REASON,*
000036  004737  000000G                 JSR     PC,DROP.CTLR
000042  022626                          CMP     (SP)+,(SP)+
000044  012601                  1$:     MOV     (SP)+,R1                        ;                               3244
000046  000207                          RTS     PC

; Routine Size:  20 words,      Routine Base:  $CODE$ + 7172
; Maximum stack depth per invocation:  4 words
```

```
;    3268  1     routine ACCESS : novalue =
;    3269  1
;    3270  1     !+
;    3271  1     !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
;    3272  1     !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;    3273  1     !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
;    3274  1     !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;    3275  1     !
;    3276  1     !       IMPLICIT INPUTS:
;    3277  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3278  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    3279  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    3280  1     !-
;    3281  1
;    3282  2         begin
;    3283  2
;    3284  2     local
;    3285  2             RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
;    3286  2             LBN : word,
;    3287  2             PASS : word initial (1);                ! LOOP PASS COUNT
;    3288  2
;    3289  2         ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB-CODE
;    3290  2     !ZZZ    LBN = (((.MAX_LBN [.L$LUN] + 1) ^ -1) and %o'77777') - 1;
;    3291  2             LBN = 0;                                !TRY LBN 0 FIRST                    !ZZZ
;    3292  2
;    3293  2         do
;    3294  3             begin                                   ! LOOP STARTS HERE
;    3295  3             P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
;    3296  3             MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
;    3297  3             MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
;    3298  3             MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
;    3299  3             MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
;    3300  3             MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD;    ! NON-SEQUENTIAL COMMAND
;    3301  3
;    3302  3             if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;    3303  3             then
;    3304  4                 begin
;    3305  4                 PUT_PKT (.P_INDEX);                 ! RETURN PACKET TO POOL
;    3306  4                 PASS = 2;                           ! NO MORE TRIES
;    3307  4                 end
;    3308  3             else
;    3309  4                 begin                               ! IF SEND WAS SUCCESSFUL
;    3310  4
;    3311  4                 do
;    3312  5                     begin
;    3313  5                     WAIT ();                        ! WAIT FOR RESPONSE
;    3314  5                     RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
;    3315  5                     RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;    3316  5
;    3317  5                     if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    3318  5                     then
;    3319  5                         PUT_RETPKT (.RP_INDX);
;    3320  5
```

C9


```
                    end
                until (.RP_ADDR [CONID] eql CID_DRIVER) or
                      ((.RP_ADDR [MESTYP] eql MT_SEQ) and
                       ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));

                if .RP_ADDR [CONID] eql CID_DRIVER  ! IF RETPKT CAME FROM "DRIVER"
                then
                    PASS = 2                        ! NO MORE TRIES
                else

                    if .RP_ADDR [ENDCOD] neq (OP_ACC or OP_END)
                    then
                        begin
                        PRINTF (DBM29);             ! "RETPKT HAS BAD ENDCODE"
!ZZZ                    EMSCMD ();
                        end
                    else
                        begin                       ! RETPKT HAS CORRECT ENDCODE
                        ST_CODE = .RP_ADDR [STSCOD];! GET STATUS CODE FROM PACKET
                        SB_CODE = .RP_ADDR [SUBCOD];! GET SUB-CODE FROM PACKET

                        if .ST_CODE eql ST_SUC      ! IF STATUS CODE INDICATES SUCCESS
                        then
                            begin
                            RESULT = SUCCESS;
                            PASS = 2;               ! NO NEED TO TRY AGAIN
                            end;

                        end;                        ! IF RETPKT HAS CORRECT ENDCODE

                PUT_RETPKT (.RP_INDX);
                end;                                ! IF SEND WAS SUCCESSFUL

            LBN = .LBN + 100;                       !TRY ANOTHER ONE                    !ZZZ
            PASS = .PASS + 1;                       ! SECOND PASS
            end                                     ! END OF PASS LOOP
        until .PASS gequ 3;

        if .RESULT eql FAILURE
        then
            begin
            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
            CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
            ERRDF (17, EGD_17, EMS_30);                 ! ACCESS FAILED
            DUR [.L$LUN] = DU_ACCESS;               ! SET REASON TO DROP UNIT
            DODU (.L$LUN);                          ! DROP UNIT
            end;                                    ! IF ACCESS FAILED

        end;                                        ! ROUTINE ACCESS
```

```
                                .SBTTL  ACCESS INITIALIZATION TEST ROUTINES
000000  004137  000000G     ACCESS: JSR     R1,$SAVE4                   ;                    3268
```

```
000004 005003                          CLR     R3                      ; RESULT                        3282
000006 012702 000001                   MOV     #1,R2                   ; *,PASS
000012 005037 000000G                  CLR     SB.CODE                 ;                               3289
000016 005037 000000G                  CLR     ST.CODE
000022 005004                          CLR     R4                      ; LBN                           3291
000024 013746 000000G          1$:     MOV     CCTLR,-(SP)             ;                               3295
000030 004737 000000G                  JSR     PC,GET.PKT
000034 010037 000000G                  MOV     R0,P.INDEX
000040 010016                          MOV     R0,(SP)                 ; P.INDEX,*                     3296
000042 012746 000106                   MOV     #106,-(SP)
000046 004737 000000G                  JSR     PC,BL$MUL
000052 013760 000000G 000016G          MOV     CDISK,MSCP.PKT+16(R0)
000060 112760 000020 000022G           MOVB    #20,MSCP.PKT+22(R0)     ;                               3297
000066 012760 001000 000026G           MOV     #1000,MSCP.PKT+26(R0)   ;                               3298
000074 010460 000046G                  MOV     R4,MSCP.PKT+46(R0)      ; LBN,*                         3299
000100 112760 000002 000004G           MOVB    #2,MSCP.PKT+4(R0)       ;                               3300
000106 013716 000000G                  MOV     P.INDEX,(SP)            ;                               3302
000112 004737 000000G                  JSR     PC,SEND
000116 005700                          TST     R0
000120 001007                          BNE     2$
000122 013716 000000G                  MOV     P.INDEX,(SP)            ;                               3305
000126 004737 000000G                  JSR     PC,PUT.PKT
000132 012702 000002                   MOV     #2,R2                   ; *,PASS                        3306
000136 000522                          BR      9$                      ;                               3302
000140 004737 000000G          2$:     JSR     PC,WAIT                 ;                               3313
000144 004737 000000G                  JSR     PC,OUT.IODQ             ;                               3314
000150 010037 000000G                  MOV     R0,RP.INDX
000154 010016                          MOV     R0,(SP)                 ; RP.INDX,*                     3315
000156 012746 000054                   MOV     #54,-(SP)
000162 004737 000000G                  JSR     PC,BL$MUL
000166 062700 000000G                  ADD     #RETPKT,R0
000172 010037 000000G                  MOV     R0,RP.ADDR
000176 132760 000360 000002            BITB    #360,2(R0)              ;                               3317
000204 001404                          BEQ     3$
000206 013716 000000G                  MOV     RP.INDX,(SP)            ;                               3319
000212 004737 000000G                  JSR     PC,PUT.RETPKT
000216 005726                  3$:     TST     (SP)+                   ;                               3312
000220 013701 000000G                  MOV     RP.ADDR,R1              ;                               3322
000224 005000                          CLR     R0
000226 126127 000003 000003            CMPB    3(R1),#3
000234 001002                          BNE     4$
000236 005200                          INC     R0
000240 000407                          BR      5$
000242 132761 000360 000002    4$:     BITB    #360,2(R1)              ;                               3323
000250 001333                          BNE     2$
000252 105761 000014                   TSTB    14(R1)                  ;                               3324
000256 100330                          BPL     2$
000260 006000                  5$:     ROR     R0                      ;                               3326
000262 103442                          BLO     7$                      ;                               3328
000264 126127 000014 000220            CMPB    14(R1),#220             ;                               3331
000272 001410                          BEQ     6$
000274 012716 000000G                  MOV     #DBM29,(SP)             ;                               3334
000300 012746 000001                   MOV     #1,-(SP)
```

E9


```
000304  010600                          MOV     SP,R0                   ; SP,*
000306  104417                          TRAP    17
000310  005726                          TST     (SP)+                   ;                       3333
000312  000430                          BR      8$                      ;                       3331
000314  116137  000016  000000G   6$:   MOVB    16(R1),ST.CODE          ;                       3339
000322  042737  177740  000000G       BIC     #177740,ST.CODE
000330  016100  000016                  MOV     16(R1),R0               ;                       3340
000334  006200                          ASR     R0
000336  006200                          ASR     R0
000340  006200                          ASR     R0
000342  006200                          ASR     R0
000344  006200                          ASR     R0
000346  042700  174000                  BIC     #174000,R0
000352  010037  000000G                 MOV     R0,SB.CODE
000356  005737  000000G                 TST     ST.CODE
000362  001004                          BNE     8$                      ;                       3342
000364  012703  000001                  MOV     #1,R3
000370  012702  000002           7$:    MOV     #2,R2                   ; *,RESULT              3345
000374  013716  000000G           8$:    MOV     RP.INDX,(SP)            ; *,PASS                3346
000400  004737  000000G                 JSR     PC,PUT.RETPKT           ;                       3351
000404  062704  000144           9$:    ADD     #144,R4                 ; *,LBN                 3354
000410  005202                          INC     R2                      ; PASS                  3355
000412  022626                          CMP     (SP)+,(SP)+             ;                       3294
000414  020227  000003                  CMP     R2,#3                   ; PASS,*                3357
000420  103601                          BLO     1$
000422  005703                          TST     R3                      ; RESULT                3359
000424  001025                          BNE     10$
000426  013700  000000G                 MOV     T.ADDR,R0               ;                       3362
000432  105260  000050                  INCB    50(R0)
000436  013700  000000G                 MOV     CUOFF,R0                ;                       3363
000442  006300                          ASL     R0
000444  063700  000000G                 ADD     CST.ADDR,R0
000450  052710  010000                  BIS     #10000,(R0)
000454  104455                          TRAP    55
000456  000021                          .WORD   21                      ;                       3364
000460  000000G                         .WORD   EGD.17
000462  000000G                         .WORD   EMS.30
000464  013700  000000G                 MOV     L$LUN,R0
000470  112760  000010  000000G         MOVB    #10,DUR(R0)             ;                       3365
000476  104451                          TRAP    51
000500  000207                  10$:    RTS     PC                      ;                       3366
                                                                        ;                       3268

; Routine Size:  161 words,    Routine Base:  $CODE$ + 7242
; Maximum stack depth per invocation:  10 words
```

F9


```
;    3370  1     %sbttl 'MULTI-DRIVE TEST ROUTINES'
;    3371  1
;    3372  1
;    3373  1     GLOBAL routine MULTI_DRIVE : novalue =
;    3374  1
;    3375  1
;    3376  1     !+
;    3377  1     !       THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;    3378  1     !       MULTI-DRIVE TEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;    3379  1     !       USER AN INDICATION OF HOW ONE OR SEVERAL RDRX DRIVES WOULD PERFORM IN
;    3380  1     !       AN OPERATING SYSTEM ENVIRONMENT.
;    3381  1     !
;    3382  1     !       THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
;    3383  1     !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE TEST DATA, THIS ROUTINE
;    3384  1     !       ENTERS A LOOP WHICH ISSUES QIOs TO ALL ACTIVE CONTROLLERS AND PROCESSES
;    3385  1     !       ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;    3386  1     !       DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;    3387  1     !       LOOP OCCURS WHEN QIOs ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;    3388  1     !       QIOS HAVE COMPLETED.
;    3389  1     !-
;    3390  1
;    3391  1
;    3392  2         begin
;    3393  2
;    3394  2         local
;    3395  2             CUR_PRIORITY : word;
;    3396  2
;    3397  2         label
;    3398  2             SEND_COMMANDS;
;    3399  2
;    3400  2         MD_INIT ();                                          ! INIT MULTI-DRIVE TEST DATA
;    3401  2         INIT_OCCURED = TRUE;                                 !
;    3402  2
;    3403  2
;    3404  3         do  begin                                            ! START OF EXECUTIVE LOOP
;    3405  3
;    3406  3             incr CTLR from 0 to (MAX_CTLR - 1) do            ! FOR EACH CONTROLLER
;    3407  4                 begin
;    3408  4                 SET_CPAR (.CTLR);                            ! SET UP CURRENT CONTROLLER PARAMETERS
;    3409  4                 GETPRI (CUR_PRIORITY);                       !
;    3410  4     !ZZZ        SETPRI (PRI04);                              ! NO INTERRUPTS WHEN EXAMINING SA
;    3411  4                 SETPRI (.BRLEVEL);                           ! NO INTERRUPTS WHEN EXAMINING SA          ZZZ
;    3412  4                 ICTLR = .CCTLR;                              ! FAKE INTERRUPTING CONTROLLER'S NUMBER
;    3413  4                 ICST_ADDR = .CST_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S CST ADDR
;    3414  4                 IDCT_ADDR = .DCT_ADDR;                       ! FAKE INTERRUPTING CONTROLLER'S DCT ADDR
;    3415  4                 IRDRX_ADDR = .ICST_ADDR [IP_ADDR];           ! FAKE INTERRUPTING CONTROLLER'S ADDRESS
;    3416  4                 IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];  ! CONTENTS OF THE SA REGISTER
;    3417  4
;    3418  5                 if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)     ! IF SA SHOWS AN ERROR
;    3419  4                 then
;    3420  5                     begin
;    3421  5                     FATAL_ERROR ();                          ! DECLARE FATAL ERROR
;    3422  5                     SETPRI (.CUR_PRIORITY);                  ! LOWER PRIORITY
```

```
;    3423  5                exitloop;                                          ! QUIT
;    3424  5                end
;    3425  5
;    3426  4            else
;    3427  4                SETPRI (.CUR_PRIORITY);                            ! IF NO ERROR, CONTINUE
;    3428  4
;    3429  4            if QIO_OK ()                                           ! IF O.K. TO ISSUE QIO(S) TO CONTROLLER
;    3430  4            then
;    3431  4                SEND_COMMANDS:
;    3432  5                    begin
;    3433  5                    QIO_GEN ();                                    ! GENERATE 1 OR 2 QIOs
;    3434  5
;    3435  5                    if (.MX1 geq 0) and                            ! IF SUCCESS ON FIRST QIO
;    3436  6                       (not .EOP_FLAG)
;    3437  5                    then
;    3438  5
;    3439  5                        if SEND (.MX1) eql SUCCESS                 ! ATTEMPT TO SEND IT. IF SUCCESS
;    3440  5                        then
;    3441  6                        BEGIN                                      !ZZZ
;    3442  6                            QIO [.CTLR] = .QIO [.CTLR] + 1;        ! INCR OUTSTANDING QIO COUNT ZZZ
;    3443  6                        END                                        !ZZZ
;    3444  6
;    3445  5                        else
;    3446  6                            begin
;    3447  6                            PUT_PKT (.MX1);                        ! RETURN PACKET TO POOL
;    3448  6                            leave SEND_COMMANDS;
;    3449  5                            end;
;    3450  5
;    3451  5
;    3452  5                    if (.MX2 geq 0) and                            ! IF SUCCESS ON SECOND QIO
;    3453  6                       (not .EOP_FLAG)
;    3454  5                    then
;    3455  6                        begin
;    3456  6
;    3457  6                        do
;    3458  6                            BREAK
;    3459  6                        until (.DCT_ADDR [CRING_CNT] lssu CRING_LEN);  ! WAIT TILL 1 MORE SLOT AVAILABLE IN CRING
;    3460  6                                                                   !
;    3461  6                        if SEND (.MX2) eql SUCCESS                 ! ATTEMPT TO SEND IT.
;    3462  6                        then
;    3463  7                        BEGIN                                      !ZZZ
;    3464  7                            QIO [.CTLR] = .QIO [.CTLR] + 1;        ! IF SUCCESS, INCR OUTSTANDING QIO COUNT
;    3465  7                        END                                        !ZZZ
;    3466  7
;    3467  6                        else
;    3468  7                            begin
;    3469  7                            PRINTF (DBM121, .CRN_HIGH, .CRN_LOW);
;    3470  7                            COMPARE_DATA = FALSE;                  ! NO SENSE IN COMPARING WRITE DATA
;    3471  7                            PUT_PKT (.MX2);                        ! RETURN PACKET TO POOL
;    3472  6                            end;
;    3473  6
;    3474  5                    end;
;    3475  5
```

```
;    3476  4                    end;
;    3477  3                end;
;    3478  3
;    3479  3
;    3480  3                BREAK;
;    3481  3                PROC_RETPKT ();
;    3482  3
;    3483  3                end
;    3484  3            until ((not QIO_OUT ()) or
;    3485  4                   ((.DCT_ADDR [CRING_CNT] eql 0) and
;    3486  2                    (.EOP_FLAG)));
;    3487  2
;    3488  2
;    3489  2            DCT_ADDR [IG_INT] = TRUE;
;    3490  2
;    3491  2
;    3492  1            end;
```

```
                                .SBTTL  MULTI.DRIVE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       MULTI.DRIVE::
                        JSR     R1,$SAVE3                       ;                       3373
000004  005746                  TST     -(SP)
000006  004737  000000V        JSR     PC,MD.INIT              ;                       3400
000012  112737  000001  000000G MOVB    #1,INIT.OCCURED         ;                       3401
000020  005001                1$:     CLR     R1                      ; CTLR          3406
000022  010146                2$:     MOV     R1,-(SP)                ; CTLR,*        3408
000024  004737  000000G        JSR     PC,SET.CPAR
000030  104440                  TRAP    40                      ;                       3409
000032  010003                  MOV     R0,R3                   ; *,CUR.PRIORITY
000034  013700  000000G        MOV     BRLEVEL,R0              ;                       3411
000040  104441                  TRAP    41
000042  013737  000000G 000104' MOV     CCTLR,ICTLR             ;                       3412
000050  013737  000000G 000076' MOV     CST.ADDR,ICST.ADDR      ;                       3413
000056  013737  000000G 000100' MOV     DCT.ADDR,IDCT.ADDR      ;                       3414
000064  017737  000076' 000000G MOV     @ICST.ADDR,IRDRX.ADDR   ;                       3415
000072  013700  000100'         MOV     IDCT.ADDR,R0            ;                       3416
000076  013702  000000G         MOV     IRDRX.ADDR,R2
000102  016266  000002  000002  MOV     2(R2),2(SP)             ; *,RC.REG
000110  016660  000002  000002  MOV     2(SP),2(R0)             ; RC.REG,*
000116  016600  000002          MOV     2(SP),R0                ;                       3418
000122  042700  077777          BIC     #77777,R0
000126  020027  100000          CMP     R0,#-100000
000132  001006                  BNE     3$
000134  004737  000000V        JSR     PC,FATAL.ERROR          ;                       3421
000140  010300                  MOV     R3,R0                   ; CUR.PRIORITY,*        3422
000142  104441                  TRAP    41
000144  005726                  TST     (SP)+                   ;                       3420
000146  000511                  BR      9$
000150  010300                3$:     MOV     R3,R0                   ; CUR.PRIORITY,*        3427
000152  104441                  TRAP    41
000154  004737  000000V        JSR     PC,QIO.OK               ;                       3429
000160  006000                  ROR     R0
```

```
000162  103077                          BCC     8$
000164  004737  000000V                 JSR     PC,QIO.GEN                      ;                       3433
000170  013700  000110'                 MOV     MX1,R0                          ;                       3435
000174  002422                          BLT     5$
000176  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       3436
000204  001016                          BNE     5$
000206  010016                          MOV     R0,(SP)                         ;                       3439
000210  004737  000000G                 JSR     PC,SEND
000214  020027  000001                  CMP     R0,#1
000220  001003                          BNE     4$
000222  105261  000000G                 INCB    QIO(R1)                         ; *(CTLR)               3442
000226  000405                          BR      5$                              ;                       3439
000230  013716  000110'          4$:    MOV     MX1,(SP)                        ;                       3447
000234  004737  000000G                 JSR     PC,PUT.PKT
000240  000450                          BR      8$                              ;                       3446
000242  005737  000112'          5$:    TST     MX2                             ;                       3452
000246  002445                          BLT     8$
000250  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       3453
000256  001041                          BNE     8$
000260  104422                  6$:     TRAP    22                              ;                       3457
000262  127727  000000G 000004          CMPB    @DCT.ADDR,#4                    ;                       3459
000270  103373                          BHIS    6$
000272  013716  000112'                 MOV     MX2,(SP)                        ;                       3461
000276  004737  000000G                 JSR     PC,SEND
000302  020027  000001                  CMP     R0,#1
000306  001003                          BNE     7$
000310  105261  000000G                 INCB    QIO(R1)                         ; *(CTLR)               3464
000314  000422                          BR      8$                              ;                       3461
000316  013716  000000G          7$:    MOV     CRN.LOW,(SP)                    ;                       3469
000322  013746  000000G                 MOV     CRN.HIGH,-(SP)
000326  012746  000000G                 MOV     #DBM121,-(SP)
000332  012746  000003                  MOV     #3,-(SP)
000336  010600                          MOV     SP,R0                           ; SP,*
000340  104417                          TRAP    17
000342  105037  001264'                 CLRB    COMPARE.DATA
000346  013716  000112'                 MOV     MX2,(SP)                        ;                       3470
000352  004737  000000G                 JSR     PC,PUT.PKT                      ;                       3471
000356  062706  000006                  ADD     #6,SP
000362  005726                  8$:     TST     (SP)+                           ;                       3468
000364  005201                          INC     R1                              ;                       3407
000366  000243                          .WORD CLV!CLC                           ; CTLR                  3406

000370  003614                          BLE     2$
000372  104422                  9$:     TRAP    22                              ;                       3477
000374  004737  000000V                 JSR     PC,PROC.RETPKT                  ;                       3481
000400  004737  000000V                 JSR     PC,QIO.OUT                      ;                       3484
000404  006000                          ROR     R0
000406  103011                          BCC     10$
000410  105777  000000G                 TSTB    @DCT.ADDR                       ;                       3485
000414  001201                          BNE     1$
000416  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       3486
000424  001002                          BNE     10$
000426  000137  007764'                 JMP     1$
```

J9


```
000432  052777  040000  000000G        10$:    BIS     #40000,@DCT.ADDR                ;                       3489
000440  005726                                 TST     (SP)+                           ;                       3373
000442  000207                                 RTS     PC

; Routine Size:  146 words,    Routine Base:  $CODE$ + 7744
; Maximum stack depth per invocation:  11 words

;   3493  1
```

K9


```
;    3494  1     GLOBAL routine MD_INIT : novalue =
;    3495  1
;    3496  1     !+
;    3497  1     !       THIS ROUTINE IS CALLED BY ROUTINE MULTI_DRIVE TO INITIALIZE DATA ITEMS
;    3498  1     !       USED BY THE MULTI-DRIVE TEST.
;    3499  1     !-
;    3500  1
;    3501  2         begin
;    3502  2
;    3503  2     !!ZZZ    local
;    3504  2     !!ZZZ    AVG_XFER_SIZE : word,                          ! SIZE (BYTES) OF AN AVERAGE I/O XFER
;    3505  2     !!ZZZ    QUICK_PASS_CNT : word;                         ! AVG NO. OF I/O OPERATIONS IN A QUICK PASS
;    3506  2
;    3507  2         if not .INIT_OCCURED                               ! IF THIS IS A START
;    3508  2         then
;    3509  2             INIT_IO_BUFF ();                               ! PARTITION FREE MEMORY INTO I/O BUFFERS
;    3510  2
;    3511  2         if (.ENTRY_REASON neq CONT) and                    ! IF START, RESTART, OR PWR FAIL
;    3512  3            (.ENTRY_REASON neq NEW_PASS)
;    3513  2         then
;    3514  2
;    3515  2             incr CTLR from 0 to (MAX_CTLR - 1) do
;    3516  3                 begin
;    3517  3                 SET_CPAR (.CTLR);
;    3518  3
;    3519  4                 INCR DISK FROM (0 + OF_UN) TO (3 * UNIT_SIZE                    !ZZZ
;    3520  3                     + OF_UN) BY UNIT_SIZE DO                                    !ZZZ
;    3521  4                     BEGIN                                                       !ZZZ
;    3522  4                     SET_UPAR (.DISK);                                           !ZZZ
;    3523  4                     DPST [.L$LUN] = DP_CNT;        !INIT DATA PTRN SEQ TABLE    !ZZZ
;    3524  3                     END;                                                        !ZZZ
;    3525  3
;    3526  2                 END;                                                            !ZZZ
;    3527  2             INCR COUNT FROM 0 TO (QIO_PER_CTLR * MAX_CTLR - 1) DO   !INIT       !ZZZ
;    3528  2                 BUFF_OWN [.COUNT] = -1;                 !I/O BUFF ALLOC TABLE   !ZZZ
;    3529  1             END;                                        !END MD_INIT            !ZZZ


                                    .SBTTL  MD.INIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G     MD.INIT::
                            JSR     R1,$SAVE2                       ;                       3494
000004  132737  000001  000000G    BITB    #1,INIT.OCCURED         ;                       3507
000012  001002              BNE     1$
000014  004737  000000V     JSR     PC,INIT.IO.BUFF                 ;                       3509
000020  123727  000000G 000003  1$: CMPB    ENTRY.REASON,#3         ;                       3511
000026  001433              BEQ     4$
000030  123727  000000G 000005    CMPB    ENTRY.REASON,#5         ;                       3512
000036  001427              BEQ     4$
000040  005002              CLR     R2                              ; CTLR                  3515
000042  010246          2$: MOV     R2,-(SP)                        ; CTLR,*                3517
000044  004737  000000G     JSR     PC,SET.CPAR
000050  012701  000003      MOV     #3,R1                           ; *,DISK                3519
000054  010116          3$: MOV     R1,(SP)                         ; DISK,*                3522
```

L9


```
000056  004737  000000G                        JSR     PC,SET.UPAR
000062  013700  000000G                        MOV     L$LUN,R0                                        ;                                       3523
000066  112760  000025  000050'                MOVB    #25,DPST(R0)
000074  062701  000012                         ADD     #12,R1                                          ; *,DISK                                3519
000100  020127  000041                         CMP     R1,#41                                          ; DISK,*
000104  003763                                 BLE     3$
000106  005726                                 TST     (SP)+                                           ;                                       3516
000110  005202                                 INC     R2                                              ; CTLR                                  3515
000112  000243                                 .WORD CLV!CLC

000114  003752                                 BLE     2$
000116  005000                         4$:     CLR     R0                                              ; COUNT                                 3527
000120  112760  000377  000000G         5$:     MOVB    #377,BUFF.OWN(R0)                               ; *,*(COUNT)                            3528
000126  005200                                 INC     R0                                              ; COUNT                                 3527
000130  020027  000007                         CMP     R0,#7                                           ; COUNT,*
000134  003771                                 BLE     5$
000136  000207                                 RTS     PC                                              ;                                       3494

; Routine Size:  48 words,      Routine Base:  $CODE$ + 10410
; Maximum stack depth per invocation:  5 words

;   3530  1
```

M9


```
;    3531  1     GLOBAL routine INIT_IO_BUFF : novalue =
;    3532  1
;    3533  1     !+
;    3534  1     !       THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE TEST IS FIRST
;    3535  1     !       STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;    3536  1     !       COLLECTION OF I/O BUFFERS. THE SIZE OF EACH I/O BUFFER IS DETERMINED
;    3537  1     !       BY A NUMBER OF FACTORS, INCLUDING THE NUMBER OF UNITS, THE NUMBER OF
;    3538  1     !       CONTROLLERS, AND THE SIZE OF FREE MEMORY.
;    3539  1     !
;    3540  1     !       ONCE THE BUFFER SIZE IS DETERMINED, THE NUMBER OF I/O BUFFERS IS
;    3541  1     !       CALCULATED. FINALLY, THE BUFFER ADDRESS (BUFF_ADDR) TABLE IS LOADED
;    3542  1     !       WITH FIXED BUFFER DESCRIPTORS THAT ARE USED IN THE ALLOCATION AND
;    3543  1     !       DEALLOCATION PROCESS.
;    3544  1     !
;    3545  1     !       IMPLICIT INPUTS:
;    3546  1     !               CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
;    3547  1     !               L$UNIT - THE NUMBER OF UNITS AVAILABLE FOR TESTING
;    3548  1     !               FREE_MEM_ADDR - START OF FREE MEMORY
;    3549  1     !-
;    3550  1
;    3551  2         begin
;    3552  2         BUFF_ADDR [0] = (.FREE_MEM_ADDR + 2 + 1) and %o'177776';                ! START OF READ/WRITE BUFFERS
;    3553  2
;    3554  2         while (.BUFF_ADDR [0] and %o'37') neq 0 do                              ! FORCE FIRST I/O BUFFER TO START
;    3555  2             BUFF_ADDR [0] = .BUFF_ADDR [0] + 2;                                 ! ON EVEN BOUNDARY
;    3556  2
;    3557  2         BYTS_PER_QIO = ((.DRS_START - .BUFF_ADDR [0]) / (QIO_PER_CTLR * MAX_CTLR)) and %o'177740';
;    3558  2                                                                                 ! MAX TRANSFER SIZE
;    3559  2
;    3560  2         if .BYTS_PER_QIO gtru MAX_XFER_SIZE
;    3561  2         then
;    3562  2             BYTS_PER_QIO = MAX_XFER_SIZE;                                       ! ADJUST TRANSFER SIZE LOWER
;    3563  2
;    3564  2         if .BYTS_PER_QIO lssu 32
;    3565  2         then
;    3566  3             begin
;    3567  3             ERRSF (2, EGS_02, 0);                                               ! ERROR IF NOT ENOUGH MEMORY
;    3568  3             DOCLN;
;    3569  2             end;
;    3570  2
;    3571  2         if (QIO_PER_CTLR * MAX_CTLR) gtru 1
;    3572  2         then
;    3573  2
;    3574  2             incr index from 1 to (QIO_PER_CTLR * MAX_CTLR - 1) do               ! INIT REMAINING TABLE ENTRIES
;    3575  2                 BUFF_ADDR [.index] = .BUFF_ADDR [.index - 1] + .BYTS_PER_QIO;   ! FIXED BUFFER ADDRESS
;    3576  2
;    3577  1         end;                                                                    ! ROUTINE INIT_IO_BUFF


                                        .SBTTL  INIT.IO.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       INIT.IO.BUFF::
                                        JSR     R1,$SAVE3                       ;                       3531
000004  013700  000000G                 MOV     FREE.MEM.ADDR,R0                ;                       3552
```

N9


```
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$:     BIT     #37,BUFF.ADDR                   ;                       3554
000034  001404                               BEQ     2$
000036  062737  000002  000000G               ADD     #2,BUFF.ADDR                    ;                       3555
000044  000770                               BR      1$                              ;                       3554
000046  013746  001252'              2$:     MOV     DRS.START,-(SP)                 ;                       3557
000052  163716  000000G                       SUB     BUFF.ADDR,(SP)
000056  012746  000010                       MOV     #10.,-(SP)
000062  004737  000000G                       JSR     PC,BL$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              ;                       3560
000106  101403                               BLOS    3$
000110  012737  001400  000000G               MOV     #1400,BYTS.PER.QIO              ;                       3562
000116  023727  000000G 000040       3$:     CMP     BYTS.PER.QIO,#40                ;                       3564
000124  103005                               BHIS    4$
000126  104454                               TRAP    54                              ;                       3567
000130  000002                               .WORD   2
000132  000000G                              .WORD   EGS.02
000134  000000                               .WORD   0
000136  104444                               TRAP    44
000140  012702  000001              4$:     MOV     #1,R2                           ; *,INDEX               3571
000144  010200                      5$:     MOV     R2,R0                           ; INDEX,*               3575
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                 3571
000172  020227  000007                       CMP     R2,#7                           ; INDEX,*
000176  003762                               BLE     5$
000200  022626                               CMP     (SP)+,(SP)+                     ;                       3551
000202  000207                               RTS     PC                              ;                       3531

; Routine Size:  66 words,    Routine Base:  $CODE$ + 10550
; Maximum stack depth per invocation:  8 words
```

```
;  3578  1     GLOBAL routine QIO_OK =
;  3579  1
;  3580  1
;  3581  1     !+
;  3582  1     !       THIS ROUTINE IS CALLED BY THE MULTI DRIVE "EXECUTIVE" IN ORDER TO
;  3583  1     !       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
;  3584  1     !       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
;  3585  1     !       THE CONTROLLER MEETS 3 REQUIREMENTS:
;  3586  1     !
;  3587  1     !               A. THE CONTROLLER IS ONLINE;
;  3588  1     !               B. THE NUMBER OF OUTSTANDING QIOs IS AT LEAST 2 LESS THAN THE
;  3589  1     !                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
;  3590  1     !               C. THERE IS AT LEAST ONE DISK ONLINE TO THE CONTROLLER.
;  3591  1     !
;  3592  1     !       IF ANY OF THESE TEST FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
;  3593  1     !
;  3594  1     !       IMPLICIT INPUTS:
;  3595  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  3596  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  3597  1     !-
;  3598  1
;  3599  1
;  3600  2         begin
;  3601  2
;  3602  2         local                                                   !MMM
;  3603  2
;  3604  2             MILLISECOND : WORD,                                 !MMM
;  3605  2             DUTY_CYCLE : WORD;                                  !MMM
;  3606  2
;  3607  2
;  3608  2         MILLISECOND = .CLK_TICKS mod 60;                        !MMM
;  3609  2
;  3610  2         if (.CST_ADDR [STATE] eql ONLINE) and                   ! IF CONTROLLER IS ONLINE
;  3611  2            (not .EOP_FLAG) and
!MMM    3612  2         ((.QIO [.CCTLR] + 2) lequ QIO_PER_CTLR) and         ! IF OUTSTANDING QIO COUNT IS O.K.
;  3613  2            (.QIO [.CCTLR] lequ (CRING_LEN-2)) and               ! IF OUTSTANDING QIO COUNT IS O.K.    MMM
;  3614  2            (.CST_ADDR [U_CNT] neq 0) and                        ! IF THERE IS VALID UNIT
;  3615  2            ((not .EL_FLUSH [.CCTLR]) or (.QIO [.CCTLR] eqlu 0))    ! MMM
;  3616  2
;  3617  2         then
;  3618  3             begin                                               ! MMM
!;  3619  3              if .CSR_MEM and .TST_PAR                           ! MMM
!;  3620  3              then                                               ! MMM
!;  3621  3                  if not .PAR_TSD                                ! MMM
!;  3622  3                  then
!;  3623  3                      if .QIO [.CCTLR] geq 1                     ! MMM PROCESS ONE COMMAND AT A TIME UNTIL
!;  3624  3                      then                                       ! PARITY TEST IS COMPLETE
!;  3625  3                          return FALSE;
;  3626  3             EL_FLUSH [.CCTLR] = 0;                             ! MMM
!;  3627  3              if (((not .TST_PAR) or (.TST_PAR and .PAR_TSD)) and (.SWP_XFER NEQ 0))          ! MMM
!;  3628  3             then
;  3629  3              if .MAN_TST                                        ! MMM
;  3630  3              then                                               ! MMM
```

C10


```
;  3631  3              if (DUTY_CYCLE = .MILLISECOND mod 60) gtru 15     !LOOK AT 'CURRENT SECOND' MMM
;  3632  3              then
;  3633  3                  return FALSE;                                 !FIRST 15 OF 60 TICKS (=1 SECOND) IS 25% DUTY CYCLE MMM
;  3634  3          return TRUE;                                            ! "TRUE" EXIT POINT
;  3635  3          end
;  3636  2      else
;  3637  2          return FALSE;                                           ! "FALSE" EXIT POINT
;  3638  2
;  3639  1      end;                                                        !MMM


                                        .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          QIO.OK::JSR     R1,$SAVE2                       ;                       3578
000004  013746  000000G                  MOV     CLK.TICKS,-(SP)                 ;                       3608
000010  012746  000074                   MOV     #74,-(SP)
000014  004737  000000G                  JSR     PC,BL$MOD
000020  013701  000000G                  MOV     CST.ADDR,R1                     ;                       3610
000024  005761  000002                   TST     2(R1)
000030  100061                           BPL     3$
000032  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                       3611
000040  001055                           BNE     3$
000042  013701  000000G                  MOV     CCTLR,R1                        ;                       3613
000046  126127  000000G 000002          CMPB    QIO(R1),#2
000054  101047                           BHI     3$
000056  013701  000000G                  MOV     CST.ADDR,R1                     ;                       3614
000062  105761  000005                   TSTB    5(R1)
000066  001442                           BEQ     3$
000070  013701  000000G                  MOV     CCTLR,R1
000074  010102                           MOV     R1,R2                           ;                       3615
000076  006302                           ASL     R2
000100  032762  000001  000106'         BIT     #1,EL.FLUSH(R2)
000106  001403                           BEQ     1$
000110  105761  000000G                  TSTB    QIO(R1)
000114  001027                           BNE     3$
000116  013701  000000G          1$:     MOV     CCTLR,R1                        ;                       3626
000122  006301                           ASL     R1
000124  005061  000106'                  CLR     EL.FLUSH(R1)
000130  032737  000001  000000G         BIT     #1,MAN.TST                      ;                       3629
000136  001413                           BEQ     2$
000140  010016                           MOV     R0,(SP)                         ; MILLISECOND,*         3631
000142  012746  000074                   MOV     #74,-(SP)
000146  004737  000000G                  JSR     PC,BL$MOD
000152  005726                           TST     (SP)+
000154  020027  000017                   CMP     R0,#17                          ; DUTY.CYCLE,*
000160  101402                           BLOS    2$
000162  022626                           CMP     (SP)+,(SP)+                     ;                       3633
000164  000406                           BR      5$
000166  012700  000001          2$:     MOV     #1,R0                           ;                       3637
000172  000401                           BR      4$
000174  005000                  3$:     CLR     R0
000176  022626                  4$:     CMP     (SP)+,(SP)+                     ;                       3610
000200  000207                           RTS     PC                              ;                       3600
000202  005000                  5$:     CLR     R0                              ;                       3578
```

```
000204  000207                                  RTS     PC

; Routine Size:  67 words,     Routine Base:  $CODE$ + 10754
; Maximum stack depth per invocation:  7 words

;   3640  1
```

E10


```
;    3641  1     GLOBAL routine QIO_OUT =
;    3642  1
;    3643  1     !+
;    3644  1     !       THIS ROUTINE IS CALLED BY THE MULTI DRIVE EXECUTIVE FOR DETERMINING THE
;    3645  1     !       END OF THE MULTI-DRIVE TEST. ITS PURPOSE IS TO EXAMINE THE QIO VECTOR
;    3646  1     !       FOR ANY OUTSTANDING QIOs ON ANY CONTROLLER. A VALUE OF "TRUE" IS
;    3647  1     !       RETURNED IF THERE IS AT LEAST ONE QIO OUTSTANDING ON ANY CONTROLLER.
;    3648  1     !       OTHERWISE, "FALSE" IS RETURNED INDICATING NO OUTSTANDING QIOs.
;    3649  1     !-
;    3650  1
;    3651  2         begin
;    3652  2
;    3653  2         incr CTLR from 0 to (MAX_CTLR - 1) do
;    3654  3             begin
;    3655  3             SET_CPAR (.CTLR);                       ! SET UP CURRENT CONTROLLER PARAMETERS
;    3656  3
;    3657  3             if .CST_ADDR [STATE] eql ONLINE         ! IF CONTROLLER IS ONLINE
;    3658  3             then
;    3659  3                 return TRUE;
;    3660  3
;    3661  2             end;
;    3662  2
;    3663  2         return FALSE;                               ! EXIT - NO CONTROLLERS ONLINE
;    3664  1         end;


                                        .SBTTL  QIO.OUT MULTI-DRIVE TEST ROUTINES
000000  010146                  QIO.OUT::
                                        MOV     R1,-(SP)                        ;                       3641
000002  005001                          CLR     R1                              ; CTLR                  3653
000004  010146                  1$:     MOV     R1,-(SP)                        ; CTLR,*                3655
000006  004737  000000G                 JSR     PC,SET.CPAR
000012  013700  000000G                 MOV     CST.ADDR,R0                     ;                       3657
000016  005760  000002                  TST     2(R0)
000022  100004                          BPL     2$
000024  005726                          TST     (SP)+                           ;                       3659
000026  012700  000001                  MOV     #1,R0
000032  000405                          BR      3$
000034  005726                  2$:     TST     (SP)+                           ;                       3654
000036  005201                          INC     R1                              ; CTLR                  3653
000040  000243                          .WORD CLV!CLC

000042  003760                          BLE     1$
000044  005000                          CLR     R0
000046  012601                  3$:     MOV     (SP)+,R1                        ;                       3651
000050  000207                          RTS     PC                              ;                       3641

; Routine Size:  21 words,      Routine Base:  $CODE$ + 11162
; Maximum stack depth per invocation:  3 words
```

F10


```
;    3665  1     GLOBAL routine QIO_GEN : novalue =
;    3666  1
;    3667  1     !+
;    3668  1     !       THIS ROUTINE IS CALLED BY THE MULTI DRIVE EXECUTIVE FOR AN ONLINE
;    3669  1     !       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
;    3670  1     !       RESPONSIBLE FOR SECURING ONE OR TWO MSCP PACKETS AND LOADING THEM
;    3671  1     !       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
;    3672  1     !       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
;    3673  1     !       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
;    3674  1     !
;    3675  1     !       EACH FIELD OF THE PACKET(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
;    3676  1     !       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
;    3677  1     !       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
;    3678  1     !
;    3679  1     !       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
;    3680  1     !       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
;    3681  1     !       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
;    3682  1     !       GENERATED WITH THE SAME LBN AND BYTE COUNT.
;    3683  1     !
;    3684  1     !       AFTER THE PACKET(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
;    3685  1     !       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
;    3686  1     !       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
;    3687  1     !       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
;    3688  1     !       CONTAIN VALID MSCP PACKET INDECES, OR -1.
;    3689  1     !
;    3690  1     !       IMPLICIT INPUTS:
;    3691  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3692  1     !-
;    3693  1
;    3694  2         begin
;    3695  2         MX2 = -1;                                   ! ASSUME FAILURE IN SECURING 2ND PACKET
;    3696  2
;    3697  2         if (MX1 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 1ST PACKET. IF FAILURE
;    3698  2         then
;    3699  2             return;                                 ! NO POINT IN CONTINUING
;    3700  2
;    3701  2         if (MX2 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 2ND PACKET. IF FAILURE
;    3702  2         then
;    3703  3             begin
;    3704  3             PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
;    3705  3             MX1 = -1;                               ! INDICATE FAILURE
;    3706  3             return;                                 ! DONE
;    3707  2             end;
;    3708  2
;    3709  2         MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);     ! CALCULATE STARTING ADDRESSES
;    3710  2         MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);     !       OF BOTH PACKETS
;    3711  2         GET_RANDOM ();                              ! GENERATE A SET OF RANDOM NUMBERS
;    3712  2         QIO_UNIT ();                                ! LOAD RANDOM UNIT NUMBER INTO PACKETS
;    3713  2
;    3714  2         if .EOP_FLAG                                ! RETURN IF NO UNIT ONLINE
;    3715  2         then
;    3716  2             return;
;    3717  2
```

```
;    3718  2          QIO_FUNC ();                                ! LOAD RANDOM FUNCTION CODE (OPCODE)
;    3719  2          QIO_LBN ();                                 ! LOAD LBN (RANDOM OR SEQUENTIAL)
;    3720  2          QIO_SIZE ();                                ! LOAD RANDOM BYTE COUNT
;    3721  2          GET_IO_BUFF (MAD1 [BUF_0]);                 ! TRY TO GET AN I/O BUFFER
;    3722  2
;    3723  2          if .MX2 geq 0                               ! IF TWO QIOs ARE TO BE ISSUED
;    3724  2          then
;    3725  3              begin
;    3726  3              GET_IO_BUFF (MAD2 [BUF_0]);             ! TRY TO GET 2ND I/O BUFFER
;    3727  3
;    3728  3              if .MAD2 [BUF_0] eqla 0                 ! IF 2ND BUFFER ALLOCATION FAILED
;    3729  3              then
;    3730  4                  begin
;    3731  4
;    3732  4                  if .MAD1 [BUF_0] neqa 0             ! IF 1ST I/O BUFFER WAS ALLOCATED
;    3733  4                  then
;    3734  5                      begin
;    3735  5                      PUT_IO_BUFF (MAD1 [BUF_0]);     ! RETURN 1ST I/O BUFFER TO POOL
;    3736  5                      MAD1 [BUF_0] = 0;               ! MARK IT AS FAILED
;    3737  4                      end;
;    3738  4
;    3739  4                  PUT_PKT (.MX2);                     ! RETURN 2ND PACKET TO POOL
;    3740  4                  MX2 = -1;                           ! INDICATE FAILURE
;    3741  3                  end;                                ! IF 2ND I/O BUFFER ALLOCATION FAILED
;    3742  3
;    3743  2              end;                                    ! IF TWO QIOs ARE TO BE ISSUED
;    3744  2
;    3745  2          if .MAD1 [BUF_0] eqla 0                     ! IF 1ST I/O BUFFER ALLOCATION FAILED
;    3746  2          then
;    3747  3              begin
;    3748  3              PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
;    3749  3              MX1 = -1;                               ! INDICATE FAILURE
;    3750  3              end
;    3751  2          else
;    3752  2
;    3753  2              if .MAD1 [OPCODE] eql OP_WRT            ! OTHERWISE, IF 1ST OPCODE IS A WRITE  (ALL IS O.K.)
;    3754  2              then
;    3755  2                  FILL_BUFF ();                       ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;    3756  2
;    3757  1          end;                                        ! ROUTINE QIO_GEN

                                        .SBTTL  QIO.GEN MULTI-DRIVE TEST ROUTINES
000000  012737  177777  000112'  QIO.GEN::
                                        MOV     #-1,MX2                         ;                       3695
000006  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3697
000012  004737  000000G                 JSR     PC,GET.PKT
000016  010037  000110'                 MOV     R0,MX1
000022  005726                          TST     (SP)+
000024  005700                          TST     R0                              ; MX1
000026  002563                          BLT     6$                              ;
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3699
000034  004737  000000G                 JSR     PC,GET.PKT                      ;                       3701
```

H10


```
000040  010037  000112'                        MOV     R0,MX2
000044  005726                                 TST     (SP)+
000046  005700                                 TST     R0                      ; MX2
000050  002011                                 BGE     1$
000052  013746  000110'                        MOV     MX1,-(SP)               ;                       3704
000056  004737  000000G                       JSR     PC,PUT.PKT
000062  012737  177777  000110'               MOV     #-1,MX1                 ;                       3705
000070  005726                                 TST     (SP)+                   ;                       3706
000072  000207                                 RTS     PC                      ;                       3703
000074  013746  000110'                1$:     MOV     MX1,-(SP)               ;                       3709
000100  012746  000106                        MOV     #106.,-(SP)
000104  004737  000000G                       JSR     PC,BL$MUL
000110  062700  000000G                       ADD     #MSCP.PKT,R0
000114  010037  000114'                       MOV     R0,MAD1
000120  013716  000112'                       MOV     MX2,(SP)
000124  012746  000106                        MOV     #106.,-(SP)             ;                       3710
000130  004737  000000G                       JSR     PC,BL$MUL
000134  062700  000000G                       ADD     #MSCP.PKT,R0
000140  010037  000116'                       MOV     R0,MAD2
000144  004737  000000V                       JSR     PC,GET.RANDOM
000150  004737  000000V                       JSR     PC,QIO.UNIT             ;                       3711
000154  132737  000001  000000G               BITB    #1,EOP.FLAG             ;                       3712
000162  001103                                 BNE     5$                      ;                       3714
000164  004737  000000V                       JSR     PC,QIO.FUNC             ;                       3665
000170  004737  000000V                       JSR     PC,QIO.LBN              ;                       3718
000174  004737  000000V                       JSR     PC,QIO.SIZE             ;                       3719
000200  013716  000114'                       MOV     MAD1,(SP)               ;                       3720
000204  062716  000032                        ADD     #32,(SP)                ;                       3721
000210  004737  000000G                       JSR     PC,GET.IO.BUFF
000214  005737  000112'                       TST     MX2
000220  002437                                 BLT     3$                      ;                       3723
000222  013716  000116'                       MOV     MAD2,(SP)               ;                       3726
000226  062716  000032                        ADD     #32,(SP)
000232  004737  000000G                       JSR     PC,GET.IO.BUFF
000236  013700  000116'                       MOV     MAD2,R0
000242  005760  000032                        TST     32(R0)                  ;                       3728
000246  001024                                 BNE     3$
000250  013700  000114'                       MOV     MAD1,R0                 ;                       3732
000254  062700  000032                        ADD     #32,R0
000260  005710                                 TST     (R0)
000262  001407                                 BEQ     2$
000264  010016                                 MOV     R0,(SP)                 ;                       3735
000266  004737  000000G                       JSR     PC,PUT.IO.BUFF
000272  013700  000114'                       MOV     MAD1,R0                 ;                       3736
000276  005060  000032                        CLR     32(R0)
000302  013716  000112'                2$:     MOV     MX2,(SP)                ;                       3739
000306  004737  000000G                       JSR     PC,PUT.PKT
000312  012737  177777  000112'               MOV     #-1,MX2                 ;                       3740
000320  013700  000114'                3$:     MOV     MAD1,R0                 ;                       3745
000324  005760  000032                        TST     32(R0)
000330  001010                                 BNE     4$
000332  013716  000110'                       MOV     MX1,(SP)                ;                       3748
000336  004737  000000G                       JSR     PC,PUT.PKT
```

I10


```
000342  012737  177777  000110'                  MOV     #-1,MX1                 ;                       3749
000350  000410                                   BR      5$                      ;                       3745
000352  013700  000114'                  4$:     MOV     MAD1,R0                 ;                       3753
000356  126027  000022  000042                   CMPB    22(R0),#42
000364  001002                                   BNE     5$
000366  004737  000000V                          JSR     PC,FILL.BUFF            ;                       3755
000372  062706  000006                   5$:     ADD     #6,SP                   ;                       3694
000376  000207                           6$:     RTS     PC                      ;                       3665

; Routine Size:  128 words,     Routine Base:  $CODE$ + 11234
; Maximum stack depth per invocation:  4 words
```

```
;   3758  1     GLOBAL routine GET_RANDOM : novalue =
;   3759  1
;   3760  1     !+
;   3761  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;   3762  1     !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;   3763  1     !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;   3764  1     !       OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;   3765  1     !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;   3766  1     !-
;   3767  1
;   3768  2         begin
;   3769  2
;   3770  2         own
;   3771  2             SEED : word initial (173),
;   3772  2             NEXT_RANDNUM : word initial (245);
;   3773  2
;   3774  2         incr COUNT from 0 to (RDM_LEN - 1) do
;   3775  3             begin
;   3776  3             SEED = (.SEED + .NEXT_RANDNUM + 1) * 4;
;   3777  3             NEXT_RANDNUM = (.NEXT_RANDNUM / 4) + .SEED;
;   3778  3             RANDOM [.COUNT] = .NEXT_RANDNUM;
;   3779  2             end;
;   3780  2
;   3781  1         end;


001274                                  .PSECT  $GGG$,  RO
001274  000255                  SEED:   .WORD   255
001276  000365                  NEXT.RANDNUM:
                                        .WORD   365


                                        .SBTTL  GET.RANDOM MULTI-DRIVE TEST ROUTINES
011634                                  .PSECT  $CODE$,  RO

000000  004137  000000G         GET.RANDOM::
                                        JSR     R1,$SAVE3                       ;                       3758
000004  013703  001274'                 MOV     SEED,R3                         ;                       3776
000010  013702  001276'                 MOV     NEXT.RANDNUM,R2
000014  005001                          CLR     R1                              ; COUNT                 3774
000016  010200                  1$:     MOV     R2,R0                           ;                       3776
000020  060300                          ADD     R3,R0
000022  006300                          ASL     R0
000024  006300                          ASL     R0
000026  010037  001274'                 MOV     R0,SEED
000032  062737  000004  001274'         ADD     #4,SEED
000040  010246                          MOV     R2,-(SP)                        ;                       3777
000042  012746  000004                  MOV     #4,-(SP)
000046  004737  000000G                 JSR     PC,BL$DIV
000052  013703  001274'                 MOV     SEED,R3
000056  060300                          ADD     R3,R0
000060  010037  001276'                 MOV     R0,NEXT.RANDNUM
```

K10


```
000064  010002                          MOV     R0,R2                   ; NEXT.RANDNUM,*                3778
000066  010261  000000G                 MOV     R2,RANDOM(R1)           ; *,*(COUNT)
000072  022626                          CMP     (SP)+,(SP)+             ;                               3775
000074  062701  000002                  ADD     #2,R1                   ; *,COUNT                       3774
000100  020127  000036                  CMP     R1,#36                  ; COUNT,*
000104  003744                          BLE     1$
000106  000207                          RTS     PC                      ;                               3758

; Routine Size:  36 words,      Routine Base:  $CODE$ + 11634
; Maximum stack depth per invocation:  7 words
```

```
;    3782  1
;    3783  1     GLOBAL routine RANDY : novalue =
;    3784  1
;    3785  1     !+
;    3786  1     !       THIS ROUTINE GENERATES A 32-BIT RANDOM NUMBER. THE LOW 16 BITS
;    3787  1     !       ARE OUTPUT IN "RNDY0". THE HIGH 16 BITS ARE OUTPUT IN "RNDY1".
;    3788  1     !
;    3789  1     !       THE LOW 3 BITS OF CLK_TICKS SELECTS A WORD FROM 'RNDYIN'. THIS
;    3790  1     !       IS 'R_STRING'. FRAME_CNT (0-9) SELECTS A 7-BIT FRAME OF THIS
;    3791  1     !       WORD. BITS OF THIS FRAME ARE USED AS FOLLOWS:
;    3792  1     !
;    3793  1     !            BITS 0-2 ... SELECT A PATTERN FOR LOW WORD.
;    3794  1     !            BITS 1-3 ... SELECT A PATTERN FOR HIGH WORD.
;    3795  1     !            BIT   4  ... IF 1, SHIFT PATTERN LEFT.
;    3796  1     !            BITS 4-6 ... SELECTS MASKS FOR FINAL OUTPUT.
;    3797  1     !
;    3798  1     !-
;    3799  1
;    3800  2         begin
;    3801  2
;    3802  2         local
;    3803  2           PAT_LO: WORD,                                  !LO WORD OF PATTERN
;    3804  2           PAT_HI: WORD,                                  !HI WORD OF PATTERN
;    3805  2           SHIFT : WORD,                                  !LEFT-SHIFT BIT
;    3806  2           MSKNO : WORD;                                  !WHICH MASK TO USE
;    3807  2
;    3808  2
;    3809  2         IF .FRAME_CNT EQLU 0                             !IF IT'S TIME TO SAMPLE CLOCK AGAIN
;    3810  2         THEN
;    3811  3            BEGIN
;    3812  3            R_STRING = .RNDYIN [.CLK_TICKS AND 7]         !CLOCK BITS SELECT 16 BIT STRING
;    3813  3            END;
;    3814  2
;    3815  2
;    3816  2         PAT_LO = .RNDYIN [(.R_STRING + -.FRAME_CNT) AND 7];      !BITS 0-2 OF FRAME SELECT LO WD OF PATTERN
;    3817  2         PAT_HI = .RNDYIN [(.R_STRING +(-1 -.FRAME_CNT)) AND 7];  !BITS 1-3 OF FRAME SELECT HI WD OF PATTERN
;    3818  2
;    3819  2
;    3820  2         SHIFT =  (.R_STRING + (-4 -.FRAME_CNT)) AND 1;           !BIT 4 OF FRAME IS SHIFTER.
;    3821  2         PAT_LO =  .PAT_LO + .SHIFT;                              !SHIFT PATTERN IF SHIFTER = 1
;    3822  2         PAT_HI = (.PAT_HI + .SHIFT) + .SHIFT;                    !SHIFT PATTERN AND ADD 1 IF SHIFTER = 1
;    3823  2
;    3824  2
;    3825  2         MSKNO = (.R_STRING + (-4 -.FRAME_CNT)) AND 7;            !GET MASK INDEX
;    3826  2         RNDY0 = .PAT_LO AND (.RNDMS0 [.MSKNO]);                  !MASK LO WORD
;    3827  2         RNDY1 = .PAT_HI AND (.RNDMS1 [.MSKNO]);                  !MASK HI WORD
;    3828  2
;    3829  2
;    3830  2         FRAME_CNT = .FRAME_CNT + 1;                              !SHIFT FRAME LEFT ONE BIT
;    3831  2         IF .FRAME_CNT GTRU 9                                     !IF DONE TEN RANDOM 32-BIT NUMBERS
;    3832  2         THEN
;    3833  2            FRAME_CNT = 0;                                        !ZERO IT, SO WE'LL READ CLOCK NEXT TIME
;    3834  2
```

M10


```
;   3835  1     END;


                                        .SBTTL  RANDY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          RANDY:: JSR     R1,$SAVE4                       ;
000004  013702  000132'                  MOV     FRAME.CNT,R2                    ;                       3783
000010  001010                          BNE     1$                              ;                       3809
000012  013700  000000G                  MOV     CLK.TICKS,R0                    ;
000016  042700  177770                  BIC     #177770,R0                      ;                       3812
000022  006300                          ASL     R0
000024  016037  000136' 000134'         MOV     RNDYIN(R0),R.STRING             ;
000032  013701  000134'          1$:     MOV     R.STRING,R1                     ;                       3811
000036  010146                          MOV     R1,-(SP)                        ;                       3816
000040  010246                          MOV     R2,-(SP)
000042  005416                          NEG     (SP)
000044  004737  000000G                  JSR     PC,BL$SHF
000050  042700  177770                  BIC     #177770,R0
000054  006300                          ASL     R0
000056  016004  000136'                  MOV     RNDYIN(R0),R4                   ; *,PAT.LO
000062  010116                          MOV     R1,(SP)                         ;                       3817
000064  012746  177777                  MOV     #-1,-(SP)
000070  160216                          SUB     R2,(SP)
000072  004737  000000G                  JSR     PC,BL$SHF
000076  042700  177770                  BIC     #177770,R0
000102  006300                          ASL     R0
000104  016003  000136'                  MOV     RNDYIN(R0),R3                   ; *,PAT.HI
000110  010116                          MOV     R1,(SP)                         ;                       3820
000112  012746  177774                  MOV     #-4,-(SP)
000116  160216                          SUB     R2,(SP)
000120  004737  000000G                  JSR     PC,BL$SHF
000124  010001                          MOV     R0,R1
000126  010102                          MOV     R1,R2
000130  042702  177776                  BIC     #177776,R2                      ; *,SHIFT
000134  010416                          MOV     R4,(SP)                         ; *,SHIFT
000136  010246                          MOV     R2,-(SP)                        ; PAT.LO,*              3821
000140  004737  000000G                  JSR     PC,BL$SHF                       ; SHIFT,*
000144  010004                          MOV     R0,R4
000146  010316                          MOV     R3,(SP)                         ; *,PAT.LO
000150  010246                          MOV     R2,-(SP)                        ; PAT.HI,*              3822
000152  004737  000000G                  JSR     PC,BL$SHF                       ; SHIFT,*
000156  060200                          ADD     R2,R0
000160  010003                          MOV     R0,R3                           ; SHIFT,*
000162  010102                          MOV     R1,R2                           ; *,PAT.HI
000164  042702  177770                  BIC     #177770,R2                      ; *,MSKNO               3825
000170  010200                          MOV     R2,R0                           ; *,MSKNO
000172  006300                          ASL     R0                              ; MSKNO,*               3826
000174  016037  000160' 000126'         MOV     RNDMS0(R0),RNDY0
000202  005104                          COM     R4
000204  040437  000126'                  BIC     R4,RNDY0
000210  010200                          MOV     R2,R0                           ; MSKNO,*               3827
000212  006300                          ASL     R0
000214  016037  000200' 000130'         MOV     RNDMS1(R0),RNDY1
000222  005103                          COM     R3
```

N10


```
000224  040337  000130'                        BIC     R3,RNDY1
000230  005237  000132'                        INC     FRAME.CNT
000234  023727  000132' 000011                 CMP     FRAME.CNT,#11                   ;                       3830
000242  101402                                 BLOS    2$                              ;                       3831
000244  005037  000132'                        CLR     FRAME.CNT
000250  062706  000014                 2$:     ADD     #14,SP                          ;                       3833
000254  000207                                 RTS     PC                              ;                       3800
                                                                                     ;                       3783

; Routine Size:  87 words,      Routine Base:  $CODE$ + 11744
; Maximum stack depth per invocation:  12 words
```

```
;  3836  1     GLOBAL routine QIO_UNIT : novalue =
;  3837  1
;  3838  1     !+
;  3839  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;  3840  1     !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;  3841  1     !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;  3842  1     !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;  3843  1     !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;  3844  1     !
;  3845  1     !       IMPLICIT INPUTS:
;  3846  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  3847  1     !
;  3848  1     !       IMPLICIT OUTPUTS:
;  3849  1     !               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
;  3850  1     !                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
;  3851  1     !-
;  3852  1
;  3853  2         begin
;  3854  2
;  3855  2         local
;  3856  2             MOD_COUNT : byte,
;  3857  2             TBL_COUNT : byte,
;  3858  2             SELECT_RD : byte initial (byte (TRUE)),
;  3859  2     !ZZZ    RD_COUNT : word initial (0),
;  3860  2             RX_COUNT : word initial (0);
;  3861  2
;  3862  2     !
;  3863  2     ! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
;  3864  2     ! SELECTED VIA THE SOFTWARE PARAMETERS
;  3865  2     !
;  3866  2     ! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
;  3867  2     ! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
;  3868  2     ! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
;  3869  2     ! MODE
;  3870  2     !
;  3871  2
;  3872  2             RD_COUNT = 0;                                                   !ZZZ
;  3873  2             RX_COUNT = 0;                                                   !ZZZ
;  3874  2
;  3875  2             incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  3876  2
;  3877  2                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  3878  2                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  3879  2                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  3880  2                 then
;  3881  2
;  3882  2                     if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;  3883  2                     then
;  3884  2                         RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
;  3885  2                     else
;  3886  2                         RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;  3887  2
;  3888  2
```

```
;    3889  2
;    3890  2          if (not BIT_TST (SWP_FLAGS, SWF_RDM)) and          ! NOT RANDOM MODE
;    3891  2             (not BIT_TST (SWP_FLAGS, SWF_SEQ))              ! NOT RANDOM SEQUEUNTIAL MODE
;    3892  2          then
;    3893  2
;    3894  2              if (.BST_CNT neq 0) and
;    3895  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3896  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3897  2                (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3898  2              then
;    3899  2                  begin                                      ! ALREADY WITHIN DEVICE
;    3900  2                  BST_CNT = .BST_CNT - 1;
;    3901  2                  SET_UPAR (.BST_DEV);
;    3902  2                  MAD1 [DK_NUM] = .CDISK;
;    3903  2                  MAD2 [DK_NUM] = .CDISK;
;    3904  2                  return;
;    3905  2                  end
;    3906  2              else
;    3907  2                  begin                                      ! GET NEW DEVICE
;    3908  2
;    3909  2      !ZZZ        incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3910  2      !ZZZ
;    3911  2      !ZZZ            if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3912  2      !ZZZ               (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3913  2      !ZZZ               (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3914  2      !ZZZ            then
;    3915  3      !ZZZ
;    3916  3      !ZZZ                if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3917  3      !ZZZ                then
;    3918  3      !ZZZ                    RD_COUNT = .RD_COUNT + 1       ! NUMBER OF RD51/52s UNDER TEST
;    3919  3      !ZZZ                else
;    3920  3      !ZZZ                    RX_COUNT = .RX_COUNT + 1;      ! NUMBER OF RX50s UNDER TEST
;    3921  3
;    3922  3              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3923  4                  begin
;    3924  4
;    3925  4                  if (.BST_DEV eql 0) or
;    3926  5                     (.BST_DEV eql ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN))
;    3927  4                  then
;    3928  4                      BST_DEV = OF_UN
;    3929  4                  else
;    3930  4                      BST_DEV = .BST_DEV + UNIT_SIZE;
;    3931  4
;    3932  4                  if (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3933  4                     (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3934  5                     (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3935  4                  then
;    3936  5                      begin
;    3937  5
;    3938  5                      if .CST_ADDR [.BST_DEV + OF_DATA, D_TYPE] eql REMOVABLE
;    3939  5                      then
;    3940  5                          BST_CNT = .RX_MAX_SEQ_CNT / .RX_COUNT
;    3941  5                      else
```

```
;    3942  5                        BST_CNT = .RD_MAX_SEQ_CNT / .RD_COUNT;
;    3943  5
;    3944  5                    if .BST_CNT eql 0
;    3945  5                    then
;    3946  5                        BST_CNT = 1;
;    3947  5
;    3948  5                    SET_UPAR (.BST_DEV);
;    3949  5                    MAD1 [DK_NUM] = .CDISK;
;    3950  5                    MAD2 [DK_NUM] = .CDISK;
;    3951  5                    return;
;    3952  4                    end;
;    3953  4
;    3954  3                end;
;    3955  3
;    3956  2            end;
;    3957  2
;    3958  2        !
;    3959  2        ! RANDOM SELECTION OF DRIVES
;    3960  2        !
;    3961  2        ! DETERMINE IF RD51/52s ARE TO BE SELECTED
;    3962  2        !
;    3963  2
;    3964  2            if ((.RANDOM [RDM_LEN - 1] and %o'077777') mod 100) gequ .SWP_RAT
;    3965  2            then
;    3966  2                SELECT_RD = FALSE;
;    3967  2
;    3968  2        !
;    3969  2        ! IF RD51/52s SELECTED
;    3970  2        !
;    3971  2        ! COUNT NUMBER OF RD51/52s AVAILABLE
;    3972  2        !
;    3973  2
;    3974  2            if .SELECT_RD
;    3975  2            then
;    3976  2                begin
;    3977  3                MOD_COUNT = 0;                          ! COUNT THE NUMBER OF RDs UNDER TEST
;    3978  3
;    3979  3                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3980  3
;    3981  3                    if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3982  3                       (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3983  3                       (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED) and
;    3984  4                       (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3985  3                    then
;    3986  4                        begin
;    3987  4                        STORAGE [.MOD_COUNT] = .OFFSET;
;    3988  4                        MOD_COUNT = .MOD_COUNT + 1;
;    3989  4                        end;
;    3990  3
;    3991  3        !
;    3992  3        ! SELECT ON OF THE RD51/52s
;    3993  3        !
;    3994  3
```

```
;    3995  3
;    3996  3                if .MOD_COUNT neq 0                              ! IF AT LEAST ONE RD51/52 PRESENT
;    3997  3                then
;    3998  4                    begin
;    3999  4                    TBL_COUNT = 0;
;    4000  4
;    4001  4                    do
;    4002  5                        begin
;    4003  5                        SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    4004  5                        TBL_COUNT = .TBL_COUNT + 1;
;    4005  5                        end
;    4006  5                    until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    4007  5                           (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    4008  4                           (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    4009  4                          (.TBL_COUNT eql RDM_LEN);
;    4010  4
;    4011  4                    MAD1 [DK_NUM] = .CDISK;
;    4012  4                    MAD2 [DK_NUM] = .CDISK;
;    4013  4                    return;
;    4014  3                    end;
;    4015  3
;    4016  2                end;
;    4017  2
;    4018  2            !
;    4019  2            ! IF NO RD51/52 SELECTED, SELECT AN RX50
;    4020  2            !
;    4021  2            ! COUNT THE NUMBER OF RX50s
;    4022  2            !
;    4023  2
;    4024  2                MOD_COUNT = 0;
;    4025  2
;    4026  2                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    4027  2
;    4028  2                    if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    4029  2                       (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    4030  2                       (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) and
;    4031  3                       (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    4032  2                    then
;    4033  3                        begin
;    4034  3                        STORAGE [.MOD_COUNT] = .OFFSET;
;    4035  3                        MOD_COUNT = .MOD_COUNT + 1;
;    4036  2                        end;
;    4037  2
;    4038  2            !
;    4039  2            ! AND CHOOSE ONE!
;    4040  2            !
;    4041  2
;    4042  2                if .MOD_COUNT neq 0
;    4043  3                then
;    4044  3                    begin
;    4045  3                    TBL_COUNT = 0;
;    4046  3
;    4047  3                    do
```

F11


```
;  4048  4                 begin
;  4049  4                 SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;  4050  4                 TBL_COUNT = .TBL_COUNT + 1;
;  4051  4                 end
;  4052  4             until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;  4053  4                    (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;  4054  3                    (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;  4055  3                   (.TBL_COUNT eql RDM_LEN);
;  4056  3
;  4057  3             MAD1 [DK_NUM] = .CDISK;
;  4058  3             MAD2 [DK_NUM] = .CDISK;
;  4059  3             return;
;  4060  2             end;
;  4061  2
;  4062  2     !
;  4063  2     ! IF NO UNIT SELECTED SO FAR BY ABOVE METHOD, SELECT ANY ONE AT RANDOM
;  4064  2     !
;  4065  2     ! COUNT ALL UNITS AVAILABLE
;  4066  2     !
;  4067  2
;  4068  2         MOD_COUNT = 0;
;  4069  2
;  4070  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  4071  2
;  4072  2             if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;  4073  2                (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;  4074  3                (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;  4075  2             then
;  4076  3                 begin
;  4077  3                 STORAGE [.MOD_COUNT] = .OFFSET;
;  4078  3                 MOD_COUNT = .MOD_COUNT + 1;
;  4079  2                 end;
;  4080  2
;  4081  2     !
;  4082  2     ! SELECT ANY ONE ONE UNIT AT RANDOM
;  4083  2     !
;  4084  2         if .MOD_COUNT neq 0
;  4085  2         then
;  4086  3             begin
;  4087  3             TBL_COUNT = 0;
;  4088  3
;  4089  3             do
;  4090  4                 begin
;  4091  4                 SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;  4092  4                 TBL_COUNT = .TBL_COUNT + 1;
;  4093  4                 end
;  4094  4             until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;  4095  4                    (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;  4096  3                    (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;  4097  3                   (.TBL_COUNT eql RDM_LEN);
;  4098  3
;  4099  3             MAD1 [DK_NUM] = .CDISK;
;  4100  3             MAD2 [DK_NUM] = .CDISK;
```

```
;  4101  3          return
;  4102  3          end
;  4103  3
;  4104  3      !
;  4105  3      ! DECLARE END-OF-PASS IF NO UNIT ONLINE
;  4106  3      !
;  4107  3
;  4108  2          else
;  4109  2              EOP_FLAG = TRUE;
;  4110  2
;  4111  1          end;                                    ! ROUTINE QIO_UNIT


                                .SBTTL  QIO.UNIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G    QIO.UNIT::
                            JSR     R1,$SAVE4               ;
000004  112704  000001      MOVB    #1,R4                   ; *,SELECT.RD                   3836
000010  005003              CLR     R3                      ; RX.COUNT                      3853
000012  005037  000000G     CLR     RD.COUNT                ;                               3872
000016  013702  000000G     MOV     CST.ADDR,R2             ;                               3877
000022  012701  000006      MOV     #6,R1                   ; *,OFFSET                      3875
000026  010100          1$: MOV     R1,R0                   ; OFFSET,*                      3877
000030  060200              ADD     R2,R0
000032  032710  040000      BIT     #40000,(R0)
000036  001415              BEQ     3$
000040  032710  020000      BIT     #20000,(R0)             ;                               3878
000044  001412              BEQ     3$
000046  032710  010000      BIT     #10000,(R0)             ;                               3879
000052  001007              BNE     3$
000054  132710  000020      BITB    #20,(R0)                ;                               3882
000060  001403              BEQ     2$
000062  005237  000000G     INC     RD.COUNT                ;                               3884
000066  000401              BR      3$                      ;                               3882
000070  005203          2$: INC     R3                      ; RX.COUNT                      3886
000072  062701  000024  3$: ADD     #24,R1                  ; *,OFFSET                      3875
000076  020127  000102      CMP     R1,#102                 ; OFFSET,*
000102  003751              BLE     1$
000104  032737  000002  000000G BIT  #2,SWP.FLAGS           ;                               3890
000112  001163              BNE     13$
000114  032737  001000  000000G BIT  #1000,SWP.FLAGS        ;                               3891
000122  001157              BNE     13$
000124  005737  001242'      TST     BST.CNT                 ;                               3894
000130  001447              BEQ     4$
000132  013700  001244'      MOV     BST.DEV,R0              ;                               3895
000136  006300              ASL     R0
000140  060200              ADD     R2,R0
000142  032710  040000      BIT     #40000,(R0)
000146  001440              BEQ     4$
000150  013700  001244'      MOV     BST.DEV,R0              ;                               3896
000154  006300              ASL     R0
000156  060200              ADD     R2,R0
000160  032710  020000      BIT     #20000,(R0)
000164  001431              BEQ     4$
```

```
000166  013700  001244'                        MOV     BST.DEV,R0                      ;                       3897
000172  006300                                  ASL     R0
000174  060200                                  ADD     R2,R0
000176  032710  010000                          BIT     #10000,(R0)
000202  001022                                  BNE     4$
000204  005337  001242'                        DEC     BST.CNT
000210  013746  001244'                        MOV     BST.DEV,-(SP)                   ;                       3900
000214  004737  000000G                       JSR     PC,SET.UPAR                     ;                       3901
000220  013700  000114'                        MOV     MAD1,R0
000224  013760  000000G 000016                MOV     CDISK,16(R0)                    ;                       3902
000232  013700  000116'                        MOV     MAD2,R0
000236  013760  000000G 000016                MOV     CDISK,16(R0)                    ;                       3903
000244  005726                                  TST     (SP)+                           ;                       3904
000246  000207                                  RTS     PC                              ;                       3899
000250  012702  000003              4$:         MOV     #3,R2                           ; *,OFFSET              3922
000254  013700  001244'              5$:         MOV     BST.DEV,R0                      ;                       3925
000260  001403                                  BEQ     6$
000262  020027  000041                          CMP     R0,#41                          ;                       3926
000266  001004                                  BNE     7$
000270  012737  000003  001244'      6$:         MOV     #3,BST.DEV                      ;                       3928
000276  000403                                  BR      8$                              ;                       3925
000300  062737  000012  001244'      7$:         ADD     #12,BST.DEV                     ;                       3930
000306  013700  001244'              8$:         MOV     BST.DEV,R0                      ;                       3932
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)                     ;                       3933
000332  001446                                  BEQ     12$
000334  032710  010000                          BIT     #10000,(R0)                     ;                       3934
000340  001043                                  BNE     12$
000342  132710  000020                          BITB    #20,(R0)                        ;                       3938
000346  001004                                  BNE     9$
000350  013746  001272'                        MOV     RX.MAX.SEQ.CNT,-(SP)            ;                       3940
000354  010346                                  MOV     R3,-(SP)                        ; RX.COUNT,*
000356  000404                                  BR      10$
000360  013746  001270'              9$:         MOV     RD.MAX.SEQ.CNT,-(SP)            ;                       3942
000364  013746  000000G                       MOV     RD.COUNT,-(SP)
000370  004737  000000G              10$:        JSR     PC,BL$DIV
000374  010037  001242'                        MOV     R0,BST.CNT
000400  001003                                  BNE     11$
000402  012737  000001  001242'              MOV     #1,BST.CNT                      ;                       3944
000410  013716  001244'              11$:        MOV     BST.DEV,(SP)                    ;                       3946
000414  004737  000000G                       JSR     PC,SET.UPAR                     ;                       3948
000420  013700  000114'                        MOV     MAD1,R0
000424  013760  000000G 000016                MOV     CDISK,16(R0)                    ;                       3949
000432  013700  000116'                        MOV     MAD2,R0
000436  013760  000000G 000016                MOV     CDISK,16(R0)                    ;                       3950
000444  022626                                  CMP     (SP)+,(SP)+
000446  000207                                  RTS     PC                              ;                       3951
000450  062702  000012              12$:        ADD     #12,R2                          ;                       3936
000454  020227  000041                          CMP     R2,#41                          ; *,OFFSET              3922
000460  003675                                  BLE     5$                              ; OFFSET,*
```

```
000462  013746  000036G        13$:    MOV     RANDOM+36,-(SP)                 ;                               3965
000466  042716  100000                  BIC     #100000,(SP)
000472  012746  000144                  MOV     #144,-(SP)
000476  004737  000000G                 JSR     PC,BL$MOD
000502  022626                          CMP     (SP)+,(SP)+
000504  020037  000000G                 CMP     R0,SWP.RAT
000510  103401                          BLO     14$
000512  105004                          CLRB    R4                              ; SELECT.RD                     3967
000514  006004                  14$:    ROR     R4                              ; SELECT.RD                     3975
000516  103105                          BCC     19$
000520  105003                          CLRB    R3                              ; MOD.COUNT                     3978
000522  012701  000003                  MOV     #3,R1                           ; *,OFFSET                      3980
000526  010100                  15$:    MOV     R1,R0                           ; OFFSET,*                      3982
000530  006300                          ASL     R0
000532  063700  000000G                 ADD     CST.ADDR,R0
000536  032710  040000                  BIT     #40000,(R0)
000542  001417                          BEQ     16$
000544  032710  020000                  BIT     #20000,(R0)                     ;                               3983
000550  001414                          BEQ     16$
000552  132710  000020                  BITB    #20,(R0)                        ;                               3984
000556  001411                          BEQ     16$
000560  032710  010000                  BIT     #10000,(R0)                     ;                               3985
000564  001006                          BNE     16$
000566  005000                          CLR     R0                              ;                               3988
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                     3989
000602  062701  000012          16$:    ADD     #12,R1                          ; *,OFFSET                      3980
000606  020127  000041                  CMP     R1,#41                          ; OFFSET,*
000612  003745                          BLE     15$
000614  105703                          TSTB    R3                              ; MOD.COUNT                     3996
000616  001445                          BEQ     19$
000620  105002                          CLRB    R2                              ; TBL.COUNT                     3999
000622  005000                  17$:    CLR     R0                              ;                               4003
000624  150200                          BISB    R2,R0                           ; TBL.COUNT,*
000626  006300                          ASL     R0
000630  016046  000000G                 MOV     RANDOM(R0),-(SP)
000634  042716  100000                  BIC     #100000,(SP)
000640  005046                          CLR     -(SP)
000642  110316                          MOVB    R3,(SP)                         ; MOD.COUNT,*
000644  004737  000000G                 JSR     PC,BL$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                     4004
000664  022626                          CMP     (SP)+,(SP)+                     ;                               4002
000666  013700  000000G                 MOV     CUOFF,R0                        ;                               4006
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)                     ;                               4007
```

J11


```
000712  001403                          BEQ     18$
000714  032710  010000                  BIT     #10000,(R0)                     ;                               4008
000720  001510                          BEQ     24$
000722  120227  000020          18$:    CMPB    R2,#20                          ; TBL.COUNT,*                   4009
000726  001335                          BNE     17$
000730  000504                          BR      24$
000732  105003                          CLRB    R3                              ;                               4011
000734  012701  000003          19$:    MOV     #3,R1                           ; MOD.COUNT                     4024
000740  010100                          MOV     R1,R0                           ; *,OFFSET                      4026
000742  006300                  20$:    ASL     R0                              ; OFFSET,*                      4028
000744  063700  000000G                 ADD     CST.ADDR,R0
000750  032710  040000                  BIT     #40000,(R0)
000754  001417                          BEQ     21$
000756  032710  020000                  BIT     #20000,(R0)                     ;                               4029
000762  001414                          BEQ     21$
000764  132710  000020                  BITB    #20,(R0)                        ;                               4030
000770  001011                          BNE     21$
000772  032710  010000                  BIT     #10000,(R0)                     ;                               4031
000776  001006                          BNE     21$
001000  005000                          CLR     R0                              ;                               4034
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                     4035
001014  062701  000012          21$:    ADD     #12,R1                          ; *,OFFSET                      4026
001020  020127  000041                  CMP     R1,#41                          ; OFFSET,*
001024  003745                          BLE     20$
001026  105703                          TSTB    R3                              ; MOD.COUNT                     4042
001030  001445                          BEQ     25$
001032  105002                          CLRB    R2                              ; TBL.COUNT                     4045
001034  005000                  22$:    CLR     R0                              ;                               4049
001036  150200                          BISB    R2,R0                           ; TBL.COUNT,*
001040  006300                          ASL     R0
001042  016046  000000G                 MOV     RANDOM(R0),-(SP)
001046  042716  100000                  BIC     #100000,(SP)
001052  005046                          CLR     -(SP)
001054  110316                          MOVB    R3,(SP)                         ; MOD.COUNT,*
001056  004737  000000G                 JSR     PC,BL$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                     4050
001076  022626                          CMP     (SP)+,(SP)+                     ;                               4048
001100  013700  000000G                 MOV     CUOFF,R0                        ;                               4052
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)                     ;                               4053
001124  001403                          BEQ     23$
001126  032710  010000                  BIT     #10000,(R0)                     ;                               4054
001132  001505                          BEQ     30$
001134  120227  000020          23$:    CMPB    R2,#20                          ; TBL.COUNT,*                   4055
```

K11


```
001140  001335                          BNE     22$
001142  000501                  24$:    BR      30$
001144  105003                  25$:    CLRB    R3                              ;                               4057
001146  012701  000003                  MOV     #3,R1                           ; MOD.COUNT                     4068
001152  010100                  26$:    MOV     R1,R0                           ; *,OFFSET                      4070
001154  006300                          ASL     R0                              ; OFFSET,*                      4072
001156  063700  000000G                 ADD     CST.ADDR,R0
001162  032710  040000                  BIT     #40000,(R0)
001166  001414                          BEQ     27$
001170  032710  020000                  BIT     #20000,(R0)                     ;                               4073
001174  001411                          BEQ     27$
001176  032710  010000                  BIT     #10000,(R0)                     ;                               4074
001202  001006                          BNE     27$
001204  005000                          CLR     R0                              ;                               4077
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                     4078
001220  062701  000012          27$:    ADD     #12,R1                          ; *,OFFSET                      4070
001224  020127  000041                  CMP     R1,#41                          ; OFFSET,*
001230  003750                          BLE     26$
001232  105703                          TSTB    R3                              ; MOD.COUNT                     4084
001234  001457                          BEQ     31$
001236  105002                          CLRB    R2                              ; TBL.COUNT                     4087
001240  005000                  28$:    CLR     R0                              ;                               4091
001242  150200                          BISB    R2,R0                           ; TBL.COUNT,*
001244  006300                          ASL     R0
001246  016046  000000G                 MOV     RANDOM(R0),-(SP)
001252  042716  100000                  BIC     #100000,(SP)
001256  005046                          CLR     -(SP)
001260  110316                          MOVB    R3,(SP)                         ; MOD.COUNT,*
001262  004737  000000G                 JSR     PC,BL$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                     4092
001302  022626                          CMP     (SP)+,(SP)+                     ;                               4090
001304  013700  000000G                 MOV     CUOFF,R0                        ;                               4094
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)                     ;                               4095
001330  001403                          BEQ     29$
001332  032710  010000                  BIT     #10000,(R0)                     ;                               4096
001336  001403                          BEQ     30$
001340  120227  000020          29$:    CMPB    R2,#20                          ; TBL.COUNT,*                   4097
001344  001335                          BNE     28$
001346  013700  000114'          30$:    MOV     MAD1,R0                         ;                               4099
001352  013760  000000G 000016          MOV     CDISK,16(R0)
001360  013700  000116'                 MOV     MAD2,R0                         ;                               4100
001364  013760  000000G 000016          MOV     CDISK,16(R0)
001372  000207                          RTS     PC                              ;                               4086
```

L11


```
001374  112737  000001  000000G        31$:    MOVB    #1,EOP.FLAG                 ;                                    4109
001402  000207                                 RTS     PC                          ;                                    3836

; Routine Size:  386 words,     Routine Base:  $CODE$ + 12222
; Maximum stack depth per invocation:  8 words
```

```
;  4112  1     GLOBAL routine QIO_FUNC : novalue =
;  4113  1
;  4114  1     !+
;  4115  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;  4116  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;  4117  1     !       BY THE FOLLWING ALGORITHM:
;  4118  1     !
;  4119  1     !               IF THE CHOSEN UNIT IS PROTECTED
;  4120  1     !               THEN
;  4121  1     !                       FUNCTION = READ
;  4122  1     !               ELSE (UNPROTECTED)
;  4123  1     !                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;  4124  1     !                       NUMBER
;  4125  1     !
;  4126  1     !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
;  4127  1     !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;  4128  1     !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;  4129  1     !       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
;  4130  1     !
;  4131  1     !
;  4132  1     !       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT   ZZZ
;  4133  1     !       BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN     ZZZ
;  4134  1     !       DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED     ZZZ
;  4135  1     !       WITH THE REGULAR MSCP TESTING OF THE LBNS.                       ZZZ
;  4136  1     !
;  4137  1     !       TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY       ZZZ
;  4138  1     !       EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O's HAVE BEEN DONE.  ZZZ
;  4139  1     !       THE DUMBER OF DUP I/O's IS 'DUPROUND'. THIS GIVES US A 25 TO 1   ZZZ
;  4140  1     !       INTERLEAVE.                                                      ZZZ
;  4141  1     !
;  4142  1     !       THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS   ZZZ
;  4143  1     !       TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON-      ZZZ
;  4144  1     !       TROLLER IS REINITIALIZED, AND QIO_FUNC ROUTINE CONTINUES FROM    ZZZ
;  4145  1     !       WHERE IT LEFT OFF.                                               ZZZ
;  4146  1     !
;  4147  1     !
;  4148  1     !       IMPLICIT INPUTS:
;  4149  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  4150  1     !               CUOFF - CURRENT UNIT CST OFFSET
;  4151  1     !
;  4152  1     !       IMPLICIT OUTPUTS:
;  4153  1     !               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
;  4154  1     !-
;  4155  1
;  4156  2         begin
;  4157  2
;  4158  2         local
;  4159  2             FUNC : word,                                                 ! OPCODE (READ OR WRITE)
;  4160  2             INDEX : word;                                                ! UNIT NO.                ZZZ
;  4161  2
;  4162  2     DUOFF = .CUOFF;                                          !SAVE IN CASE OTHER CMDS ZZZ
;  4163  2                                                  !LEFT IN QUEUE                  ZZZ
;  4164  3     IF ((.CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] LEQ 0) AND !MSCP CNT=0           ZZZ
```

```
;  4165  3          (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND          !FIXED DISK          ZZZ
;  4166  3          (.CST_ADDR [.DUOFF + OF_DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN          ZZZ
;  4167  3                                                                                  !ZZZ
;  4168  2          THEN                                                                     !ZZZ
;  4169  3              BEGIN                                                                !ZZZ
;  4170  3              PUT_PKT (.MX2);                          !RETURN 2ND ENVLOPE        !ZZZ
;  4171  3              MX2 = -1;                                !INDICATE FAILURE           ZZZ
;  4172  3              DUP ();                                  !DO DUP TEST                ZZZ
;  4173  3              CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =   !REINIT MSCP FUN-          ZZZ
;  4174  3                      (25 * .DUPROUND);                !CTION COUNTER              ZZZ
;  4175  3                                                                                  !ZZZ
;  4176  2          !       THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER    ZZZ
;  4177  2          !       CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.              ZZZ
;  4178  2          !                                                                        ZZZ
;  4179  2             DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;       !SET DUP INIT FLAG         ZZZ
;  4180  2             INIT_TEST ();                             !REINIT CONTROLLER         ZZZ
;  4181  2             DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT); !CLR DUP INIT DLAG        ZZZ
;  4182  2                                                                                  !ZZZ
;  4183  2             MX2 = -1;                                 !ASSUME NO 2ND ENVELOPE    ZZZ
;  4184  2             IF (MX1 = GET_PKT (.CCTLR)) LSS 0         !TRY FOR 1ST ENVELOPE      ZZZ
;  4185  4                 OR (.EOP_FLAG)                        !IF FAILURE                ZZZ
;  4186  3             THEN    RETURN;                           !NO POINT TO GO ON         ZZZ
;  4187  3             IF (MX2 = GET_PKT (.CCTLR)) LSS 0         !TRY FOR 2ND ENVELOPE      ZZZ
;  4188  4                 OR (.EOP_FLAG)                        !IF FAILURE                ZZZ
;  4189  4             THEN    BEGIN                                                        !ZZZ
;  4190  4                     PUT_PKT (.MX1);                   !PUT 1ST BACK IN POOL      ZZZ
;  4191  4                     MX1 = -1;                         !INDICATE FAILURE          ZZZ
;  4192  4                     RETURN;                           !DONE                      ZZZ
;  4193  3                     END;                                                         !ZZZ
;  4194  3                                                                                  !ZZZ
;  4195  3                     MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2); !CALC START ADDR     ZZZ
;  4196  3                     MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2); !OF BOTH ENVELOPES   ZZZ
;  4197  3                     GET_RANDOM ();                    !GET SET OF RANDOM NOS     ZZZ
;  4198  3                     QIO_UNIT ();                      !PUT RAND UNIT NO IN       ZZZ
;  4199  2                     END;                              !ENVELOPES                 ZZZ
;  4200  2                                                                                  !ZZZ
;  4201  2
;  4202  2          !       MSCP CODE STARTS HERE                                            ZZZ
;  4203  2          !                                                                        ZZZ
;  4204  2
;  4205  2          INDEX = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];                       !POINT TO TYPER + TYPEW FOR THIS UNIT   ZZZ
;  4206  2
;  4207  2          CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =          !                       ZZZ
;  4208  2              .CST_ADDR [.CUOFF + OF_COUNT,
;  4209  2          D_COUNT] -1;    !DECR MSCP FUNCTION CNTR ZZZ
;  4210  2
;  4211  2              MAD2 [OPCODE] = 0;                                                  ! ASSUME 2ND PACKET NOT NEEDED
;  4212  2
;  4213  2              if (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED) and       ! IF "FORCED ERROR" SET IN LAST READ,
;  4214  2                 (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and             !       REWRITE SAME BLOCK
;  4215  3                 (.FORCED_ERROR)                                                  !
;  4216  2              then
;  4217  2                  FUNC = OP_WRT
```

```
;  4218  2          else
;  4219  2
;  4220  2              if .CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql PROTECTED          ! IF UNIT IS PROTECTED
;  4221  2              then
;  4222  2                  FUNC = OP_RD                                           ! SET FUNCTION TO READ
;  4223  2              else
;  4224  2
;  4225  2      !MMM        if (.RANDOM [1] and 1)                                 ! USE 2ND RANDOM NUMBER TO SELECT
;  4226  2      !MMM        then
;  4227  2      !MMM            FUNC = OP_RD                                       !     READ
;  4228  2      !MMM        else
;  4229  2      !MMM            FUNC = OP_WRT;                                     !     WRITE
;  4230  2
;  4231  2              IF NOT .BAL_IN_PROGRESS [.INDEX]                           ! MMM
;  4232  2              THEN                                                       ! MMM
;  4233  3                 IF (.RANDOM [1] AND 1)                                  ! MMM
;  4234  3                 THEN                                                    ! MMM
;  4235  3                    FUNC = OP_RD                                         ! MMM
;  4236  3                 ELSE                                                    ! MMM
;  4237  3                    FUNC = OP_WRT                                        ! MMM
;  4238  3               ELSE                                                      ! MMM
;  4239  3                  IF .FORCE_WR [.INDEX]                                  ! MMM
;  4240  3                  THEN                                                   ! MMM
;  4241  3                     FUNC = OP_WRT                                       ! MMM
;  4242  3                  ELSE                                                   ! MMM
;  4243  3                     FUNC = OP_RD;                                       ! MMM
;  4244  3
;  4245  2
;  4246  2          if (MAD1 [OPCODE] = .FUNC) eql OP_WRT                          ! LOAD CHOSEN OPCODE. IF WRITE
;  4247  2          then
;  4248  3              begin
;  4249  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                             ! NON-SEQUENTIAL COMMAND
;  4250  3
;  4251  4              if BIT_TST (SWP_FLAGS, SWF_CWC)                            ! IF CONTROLLER DOES WRITE-COMPARES
;  4252  3              then
;  4253  3                  MAD1 [MODIFY] = MD_CMP                                 ! ADD COMPARE MODIFIER
;  4254  3              else
;  4255  3
;  4256  4                  if BIT_TST (SWP_FLAGS, SWF_HWC)                        ! IF HOST DOES WRITE-COMPARES
;  4257  3                  then
;  4258  4                      begin
;  4259  4                      MAD1 [MODIFY] = MD_EXP;                            ! SET WRITE AS AN EXPRESS REQUEST
;  4260  4                      MAD2 [OPCODE] = OP_RD;                             ! SET READ OPCODE INTO 2ND MSCP PACKET
;  4261  4                      MAD2 [MODIFY] = MD_EXP;                            ! SET READ AS AN EXPRESS REQUEST TOO
;  4262  4                      MAD2 [CMD_TYPE] = NON_SEQ_CMD;                     ! NON-SEQUENTIAL COMMAND
;  4263  3                      end;
;  4264  3              end
;  4265  2          else
;  4266  3              begin
;  4267  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                             ! NON-SEQUENTIAL COMMAND
;  4268  3
;  4269  4              if BIT_TST (SWP_FLAGS, SWF_CRC)                            ! IF READ-COMPARES - FUNCTION IS READ
;  4270  3              then
```

C12


```
;  4271  3                MAD1 [MODIFY] = MD_CMP;                          ! ADD COMPARE MODIFIER
;  4272  3
;  4273  2            end;
;  4274  2
;  4275  2            if .MAD2 [OPCODE] eql 0                              ! IF NO OPCODE IN 2ND PACKET
;  4276  2            then
;  4277  3                begin
;  4278  3                PUT_PKT (.MX2);                                  ! RETURN 2ND PACKET TO POOL
;  4279  3                MX2 = -1;                                        ! MARK IT UNUSED
;  4280  2                end;
;  4281  2
;  4282  1            end;                                                 ! ROUTINE QIO_FUNC


                                        .SBTTL  QIO.FUNC MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          QIO.FUNC::
                                        JSR     R1,$SAVE4                       ;                       4112
000004  013737  000000G 001250'         MOV     CUOFF,DUOFF                     ;                       4162
000012  013702  000000G                 MOV     CST.ADDR,R2                     ;                       4164
000016  013701  001250'                 MOV     DUOFF,R1
000022  010100                          MOV     R1,R0
000024  006300                          ASL     R0
000026  060200                          ADD     R2,R0
000030  005760  000022                  TST     22(R0)
000034  003146                          BGT     4$
000036  010100                          MOV     R1,R0                           ;                       4165
000040  006300                          ASL     R0
000042  060200                          ADD     R2,R0
000044  132710  000020                  BITB    #20,(R0)
000050  001540                          BEQ     4$
000052  010100                          MOV     R1,R0                           ;                       4166
000054  006300                          ASL     R0
000056  060200                          ADD     R2,R0
000060  005760  000020                  TST     20(R0)
000064  100532                          BMI     4$
000066  013746  000112'                 MOV     MX2,-(SP)                       ;                       4170
000072  004737  000000G                 JSR     PC,PUT.PKT
000076  012737  177777  000112'         MOV     #-1,MX2                         ;                       4171
000104  004737  000000V                 JSR     PC,DUP                          ;                       4172
000110  013701  001250'                 MOV     DUOFF,R1                        ;                       4173
000114  006301                          ASL     R1
000116  063701  000000G                 ADD     CST.ADDR,R1
000122  013716  000000G                 MOV     DUPROUND,(SP)                   ;                       4174
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                    ;                       4179
000150  004737  001306'                 JSR     PC,INIT.TEST                    ;                       4180
000154  042737  000002  000000G         BIC     #2,DUP.FLAGS                    ;                       4181
000162  012737  177777  000112'         MOV     #-1,MX2                         ;                       4183
000170  013716  000000G                 MOV     CCTLR,(SP)                      ;                       4184
000174  004737  000000G                 JSR     PC,GET.PKT
000200  010037  000110'                 MOV     R0,MX1
```

D12


```
000204  002426                          BLT     2$
000206  132737  000001  000000G         BITB    #1,EOP.FLAG                             ;                       4185
000214  001022                          BNE     2$                                      ;                       4112
000216  013716  000000G                 MOV     CCTLR,(SP)                              ;                       4187
000222  004737  000000G                 JSR     PC,GET.PKT
000226  010037  000112'                 MOV     R0,MX2
000232  002404                          BLT     1$
000234  132737  000001  000000G         BITB    #1,EOP.FLAG                             ;                       4188
000242  001411                          BEQ     3$
000244  013716  000110'         1$:     MOV     MX1,(SP)                                ;                       4190
000250  004737  000000G                 JSR     PC,PUT.PKT
000254  012737  177777  000110'         MOV     #-1,MX1                                 ;                       4191
000262  022626                  2$:     CMP     (SP)+,(SP)+                             ;                       4192
000264  000207                          RTS     PC                                      ;                       4189
000266  013716  000110'         3$:     MOV     MX1,(SP)                                ;                       4195
000272  012746  000106                  MOV     #106,-(SP)
000276  004737  000000G                 JSR     PC,BL$MUL
000302  062700  000000G                 ADD     #MSCP.PKT,R0
000306  010037  000114'                 MOV     R0,MAD1
000312  013716  000112'                 MOV     MX2,(SP)                                ;                       4196
000316  012746  000106                  MOV     #106,-(SP)
000322  004737  000000G                 JSR     PC,BL$MUL
000326  062700  000000G                 ADD     #MSCP.PKT,R0
000332  010037  000116'                 MOV     R0,MAD2
000336  004737  011634'                 JSR     PC,GET.RANDOM                           ;                       4197
000342  004737  012222'                 JSR     PC,QIO.UNIT                             ;                       4198
000346  062706  000010                  ADD     #10,SP                                  ;                       4169
000352  013702  000000G         4$:     MOV     CUOFF,R2                                ;                       4205
000356  006302                          ASL     R2
000360  063702  000000G                 ADD     CST.ADDR,R2
000364  111200                          MOVB    (R2),R0
000366  042700  177760                  BIC     #177760,R0                              ; *,INDEX
000372  013701  000000G                 MOV     CUOFF,R1                                ; *,INDEX
000376  006301                          ASL     R1                                      ;                       4207
000400  063701  000000G                 ADD     CST.ADDR,R1
000404  005361  000022                  DEC     22(R1)
000410  013701  000116'                 MOV     MAD2,R1                                 ;                       4209
000414  012704  000022                  MOV     #22,R4                                  ;                       4211
000420  060104                          ADD     R1,R4
000422  105014                          CLRB    (R4)
000424  005712                          TST     (R2)
000426  100007                          BPL     5$                                      ;                       4213
000430  132712  000020                  BITB    #20,(R2)                                ;                       4214
000434  001404                          BEQ     5$
000436  132737  000001  000000G         BITB    #1,FORCED.ERROR                         ;                       4215
000444  001021                          BNE     7$                                      ;                       4217
000446  032712  100000          5$:     BIT     #100000,(R2)                            ;                       4220
000452  001421                          BEQ     8$                                      ;                       4222
000454  006300                          ASL     R0                                      ;                       4231
000456  032760  000001  000000G         BIT     #1,BAL.IN.PROGRESS(R0)
000464  001005                          BNE     6$
000466  032737  000001  000002G         BIT     #1,RANDOM+2                             ;                       4233
000474  001405                          BEQ     7$
```

E12


```
000476  000407                                  BR      8$                                  ;                                       4235
000500  032760  000001  000000G          6$:    BIT     #1,FORCE.WR(R0)                     ;                                       4239
000506  001403                                  BEQ     8$
000510  012702  000042                   7$:    MOV     #42,R2                              ; *,FUNC                                4241
000514  000402                                  BR      9$                                  ;                                       4239
000516  012702  000041                   8$:    MOV     #41,R2                              ; *,FUNC                                4243
000522  013700  000114'                  9$:    MOV     MAD1,R0                             ;                                       4246
000526  013703  000000G                         MOV     SWP.FLAGS,R3                        ;                                       4251
000532  110260  000022                          MOVB    R2,22(R0)                           ; FUNC,*                                4246
000536  020227  000042                          CMP     R2,#42                              ; FUNC,*
000542  001025                                  BNE     10$
000544  112760  000002  000004                  MOVB    #2,4(R0)                            ;                                       4249
000552  032703  000020                          BIT     #20,R3                              ;                                       4251
000556  001025                                  BNE     11$                                 ;                                       4253
000560  032703  000040                          BIT     #40,R3                              ;                                       4256
000564  001425                                  BEQ     12$
000566  012760  100000  000024                  MOV     #-100000,24(R0)                     ;                                       4259
000574  112714  000041                          MOVB    #41,(R4)                            ;                                       4260
000600  012761  100000  000024                  MOV     #-100000,24(R1)                     ;                                       4261
000606  112761  000002  000004                  MOVB    #2,4(R1)                            ;                                       4262
000614  000411                                  BR      12$                                 ;                                       4246
000616  112760  000002  000004          10$:    MOVB    #2,4(R0)                            ;                                       4267
000624  032703  000004                          BIT     #4,R3                               ;                                       4269
000630  001403                                  BEQ     12$
000632  012760  040000  000024          11$:    MOV     #40000,24(R0)                       ;                                       4271
000640  105714                          12$:    TSTB    (R4)                                ;                                       4275
000642  001010                                  BNE     13$
000644  013746  000112'                         MOV     MX2,-(SP)                           ;                                       4278
000650  004737  000000G                         JSR     PC,PUT.PKT
000654  012737  177777  000112'                 MOV     #-1,MX2                             ;                                       4279
000662  005726                                  TST     (SP)+                               ;                                       4277
000664  000207                          13$:    RTS     PC                                  ;                                       4112

; Routine Size:  219 words,     Routine Base:  $CODE$ + 13626
; Maximum stack depth per invocation:  10 words
```

F12


```
GLOBAL ROUTINE DUP : NOVALUE =                                                  !ZZZ
!+                                                                          !ZZZ
!       THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS.    ZZZ
!       THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,      ZZZ
!       SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                    ZZZ
!                                                                           ZZZ
!       THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ-     ZZZ
!       COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL         ZZZ
!       RECORD THE STATISTICS IN THE TALLY TABLES.                          ZZZ
!                                                                           ZZZ
!       THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ        ZZZ
!       DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE          ZZZ
!       SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.         ZZZ
!       THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN          ZZZ
!       BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR    ZZZ
!       SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE     ZZZ
!       PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE    ZZZ
!       DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST-    ZZZ
!       ED IN THE RCT TABLES.                                               ZZZ
!                                                                           ZZZ
!       AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES     ZZZ
!       WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE        ZZZ
!       AS BEFORE.                                                          ZZZ
!                                                                           ZZZ
!       IMPLICIT INPUTS:                                                    ZZZ
!                       CCTLR - CURRENT CONTROLLER NUMBER                   ZZZ
!                       CST_ADDRS - CONTAINS THE CURRENT CONTROLLER         ZZZ
!                                   STATUS TABLE                            ZZZ
!                       CUOFF - CURRENT OFFSET IN CST TABLE FOR             ZZZ
!                               PARTICULAR DRIVE                            ZZZ
!                                                                           ZZZ
!       IMPLICIT OUTPUTS:                                                   ZZZ
!                       S_PATTERN - PATTERN BEING WRITTEN TO DBNS           ZZZ
!                                                                           ZZZ
!                                                                          !ZZZ
!-                                                                         !ZZZ
```

```
;   4321  1
;   4322  1
;   4323  1
;   4324  2      BEGIN
;   4325  2      OWN
;   4326  2            TEMP : WORD;
;   4327  2
;   4328  2      !PRINTX (DBM110);
;   4329  2      !PRINTX (DER10);
;   4330  2
;   4331  2      until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or        ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;   4332  2           (.EOP_FLAG eql true) do                      ! Make sure all MSCP commands are completed
;   4333  3          begin
;   4334  3          BREAK;                                        ! BREAK FOR ACT
;   4335  3          PROC_RETPKT();                                ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;   4336  3          RP_INDX = .RP_INDX + 1;                       ! INCREMENT RP_INDX
;   4337  3          if .RP_INDX geq RP_CNT then (RP_INDX = 0;);   ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
;   4338  3          RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);   ! CALCULATE RETPKT ADDRESS
;   4339  2          end;
;   4340  2
;   4341  2
;   4342  2      S_PATTERN = .RANDOM [1];                                  !OTHER UNIT VARIABLES
;   4343  2
;   4344  2      IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .duproud) GEQ 144      ! TEST TO SEE IF NEXT DBN'S TO LARGE
;   4345  2      THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);             ! CIRCLE ARROUND IF DBN TO LARGE
;   4346  2
;   4347  2      DUPIDLE ();                                               ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
;   4348  2      IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN
;   4349  2
;   4350  2      TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
;nd 4351  2      INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .duproud) DO                  ! INCREMENT FROM RELATIVE DBN TO DBN + duprou
;   4352  3            BEGIN
; WR4353DBN'S          IF .CST_ADDR [.DUOFF + OF_DBN, DUPWRITE]                                  ! IF WRITE FLAG SET IN CST TABLE THEN
;   4354  3            THEN
;   4355  4                  BEGIN
;E  4356  4                  DUPIDLE ();                                          ! MAKE SURE THE CONTROLLER IS IN AN IDLE STAT
;   4357  4                  DUPWRTDBN ();                                        ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
;   4358  3                  END;
;   4359  3
;   4360  3            DUPIDLE ();                                                ! MAKE SURE CONTROLLER IN IDLE STATE
;   4361  3            DUPREDDBN ();                                              ! CALL ROUTINE TO HANDLE READING DBN'S
;   4362  3
;   4363  3            CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
;  Z4364  3            IF .CST_ADDR [.DUOFF + OF_DBN, D_DBN]  GTRU  MAX_DBN                        !BUT NOT MORE THAN MAX NUMBER
;  Z4365  3            THEN                                                                        !IF BIGGER THAN MAX
;  Z4366  3              CST_ADDR [.DUOFF + OF_DBN, D_DBN]  =  0;                                  !MAKE IT ZERO
;   4367  3
;   4368  3
;   4369  3
;   4370  3            IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1                   ! ERROR IN DUP REINITIALIZE
;   4371  3            THEN RETURN;                                                     ! AND RETURN
;   4372  2            END;
;   4373  1      END;
```

H12


```
001300
001300                                  TEMP:   .PSECT  $GGG$,  RO
                                                .BLKW   1


                                                .SBTTL  DUP MULTI-DRIVE TEST ROUTINES
014514                                          .PSECT  $CODE$,  RO

000000  004137  000000G                DUP::   JSR     R1,$SAVE3
000004  013700  000000G                1$:     MOV     RP.ADDR,RO              ;                       4284
000010  023760  000000G 000004                CMP     CRN.LOW,4(RO)           ;                       4331
000016  001433                                BEQ     3$
000020  123727  000000G 000001                CMPB    EOP.FLAG,#1             ;                       4332
000026  001427                                BEQ     3$
000030  104422                                TRAP    22                      ;                       4333
000032  004737  000000V                       JSR     PC,PROC.RETPKT          ;                       4335
000036  005237  000000G                       INC     RP.INDX                 ;                       4336
000042  023727  000000G 000010                CMP     RP.INDX,#10             ;                       4337
000050  002402                                BLT     2$
000052  005037  000000G                       CLR     RP.INDX
000056  013746  000000G                2$:     MOV     RP.INDX,-(SP)           ;                       4338
000062  012746  000054                        MOV     #54,-(SP)
000066  004737  000000G                       JSR     PC,BL$MUL
000072  062700  000000G                       ADD     #RETPKT,RO
000076  010037  000000G                       MOV     RO,RP.ADDR
000102  022626                                CMP     (SP)+,(SP)+
000104  000737                                BR      1$                      ;                       4333
000106  013737  000002G 000000G        3$:     MOV     RANDOM+2,S.PATTERN      ;                       4331
000114  013700  001250'                       MOV     DUOFF,RO                ;                       4342
000120  006300                                ASL     RO                      ;                       4344
000122  063700  000000G                       ADD     CST.ADDR,RO
000126  005001                                CLR     R1
000130  156001  000020                        BISB    20(RO),R1
000134  063701  000000G                       ADD     DUPROUND,R1
000140  020127  000220                        CMP     R1,#220
000144  002402                                BLT     4$
000146  105060  000020                        CLRB    20(RO)
000152  004737  000000V                4$:     JSR     PC,DUPIDLE              ;                       4345
000156  013700  001250'                       MOV     DUOFF,RO                ;                       4347
000162  006300                                ASL     RO                      ;                       4348
000164  063700  000000G                       ADD     CST.ADDR,RO
000170  005760  000020                        TST     20(RO)
000174  100462                                BMI     9$
000176  116037  000020  001300'               MOVB    20(RO),TEMP             ;                       4350
000204  105037  001301'                       CLRB    TEMP+1
000210  013703  001300'                       MOV     TEMP,R3                 ;                       4351
000214  063703  000000G                       ADD     DUPROUND,R3
000220  013700  001250'                       MOV     DUOFF,RO                ;                       4353
000224  006300                                ASL     RO
000226  063700  000000G                       ADD     CST.ADDR,RO
000232  010001                                MOV     RO,R1
```

I12


```
000234  062701  000020                      ADD     #20,R1
000240  013702  001300'                     MOV     TEMP,R2                 ; *,DBNCNT                      4351
000244  000433                              BR      8$
000246  032711  010000              5$:     BIT     #10000,(R1)             ;                               4353
000252  001404                              BEQ     6$
000254  004737  000000V                     JSR     PC,DUPIDLE              ;                               4356
000260  004737  000000V                     JSR     PC,DUPWRTDBN            ;                               4357
000264  004737  000000V             6$:     JSR     PC,DUPIDLE              ;                               4360
000270  004737  000000V                     JSR     PC,DUPREDDBN            ;                               4361
000274  013700  001250'                     MOV     DUOFF,R0                ;                               4363
000300  006300                              ASL     R0
000302  063700  000000G                     ADD     CST.ADDR,R0
000306  010001                              MOV     R0,R1
000310  062701  000020                      ADD     #20,R1
000314  105211                              INCB    (R1)
000316  121127  000077                      CMPB    (R1),#77                ;                               4364
000322  101401                              BLOS    7$
000324  105011                              CLRB    (R1)                    ;                               4366
000326  032711  040000              7$:     BIT     #40000,(R1)             ;                               4370
000332  001003                              BNE     9$                      ;                               4371
000334  005202                      8$:     INC     R2                      ; DBNCNT                        4351
000336  020203                              CMP     R2,R3                   ; DBNCNT,*
000340  003742                              BLE     5$
000342  000207                      9$:     RTS     PC                      ;                               4284

; Routine Size:  114 words,    Routine Base:  $CODE$ + 14514
; Maximum stack depth per invocation:  7 words


;  4374  1
```

```
1   GLOBAL ROUTINE DUPWRTDBN : NOVALUE =
1
1   !+
1   ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
1   ! "WRTDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
1   ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM WRITES TO A DIAGONOSTIC BLOCK (DBN)
1   ! THE WORD IN "S_PATTERN" IS WRITTEN TO THE 256 WORDS IN THE DBN.  IF AN ERROR OCCURS
1   ! WHILE RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
1   ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
1   !
1   !       IMPLICIT INPUTS:
1   !               CST_ADDRS  - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
1   !               DUOFF      - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
1   !-              S_PATTERN  - CONTAINS PATTERN WORD!-
2   BEGIN
2   LOCAL
2       TRYNUM : WORD,                                  !                                                ZZZ
2       MAX_TRY_COUNT : word initial (9);               ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
2                                                       !                                                ZZZ
2   LABEL                                               !                                                ZZZ
2       DUP_WLOOP;                                      !START OF DUP WRITE RETRY LOOP                   ZZZ
2
2   !PRINTX (DER11);
2   T_ADDR [T_DBN_WT] = .T_ADDR [T_DBN_WT] + 1;         ! INCREMENT # OF WRITES GIVEN
2
2   TRYNUM = 0;                                         !ZERO TRY COUNTER                                ZZZ
2   DUP_WLOOP:                                          !LABEL FOR LOOP EXCAPE ON GOOD WRITE             ZZZ
3   BEGIN                                               !BEGIN DUP_WLOOP                                 ZZZ
3   INCR TRIES FROM 1 TO 10 DO                          !START TRYING DUP WRITES                         ZZZ
4   BEGIN                                               !BEGIN LARGE DO LOOP                             ZZZ
4                                                       !                                                ZZZ
4
4   MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                   ! PACKET SIZE                                    EXECUTE LOCAL PROGRAM WRT DBN
4   MSCP_PKT [.MX1, OPCODE] = OP_ELP;                   ! OPCODE = EXECUTE LOCAL PROGRAM
4   MSCP_PKT [.MX1, L1] = %ascii'W';                    ! FILL IN PROGRAM NAME WITH ASCII LETTERS
4   MSCP_PKT [.MX1, L2] = %ascii'R';
4   MSCP_PKT [.MX1, L3] = %ascii'T';
4   MSCP_PKT [.MX1, L4] = %ascii'D';
4   MSCP_PKT [.MX1, L5] = %ascii'B';
4   MSCP_PKT [.MX1, L6] = %ascii'N';
4   MSCP_PKT [.MX1, MODIFY] = 1;                        ! STANDALONE MODIFIER
4   !ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;             ! CALL IT IMMEDIATE
4   MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                   ! CALL ALL DUP CMDS SEQUENTIAL.                  ZZZ
4   DUPCOMMAND ();                                      ! SENDS AND RECEIVES THE COMMAND
4
4   IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1      !status error
4   THEN RETURN;                                        ! AND RETURN
4
5   DO (MX1 = GET_PKT (.CCTLR))
4   UNTIL (.MX1 GEQ 0);                         ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
4
4           MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                RECIEVE DATA
4           MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
```

K12


```
;ec 4428  4              MSCP_PKT [.MX1, BC_LO] = 80;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 2 ***see pg 26 of DUP sp
;    4429  4              MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;    4430  4              MSCP_PKT [.MX1, MODIFY] = 0;                    !
;    4431  4  !ZZZ        MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;    4432  4              MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;               ! CALL ALL DUP CMDS SEQUENTIAL.                          ZZZ
;    4433  4              DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    4434  4
;    4435  4              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;    4436  4                 (.DUPPKT [DUPTYPE] NEQU 1) OR                !dup type error
;    4437  5                 (.DUPPKT [DUPMESG] NEQU 6)
;    4438  4              THEN
;    4439  5                      (D_FAIL = 1;                            !TELL HARD_ERROR IT WAS A DUP PROBLEM            ZZZ
;    4440  5                       HARD_ERROR ();                         !
;    4441  5                       D_FAIL = 0;                            !                                                ZZZ
;    4442  5                       CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    4443  4                       RETURN;);                              ! NO POINT IN CONTINUING
;    4444  4
;    4445  5              DO (MX1 = GET_PKT (.CCTLR))
;    4446  4              UNTIL (.MX1 GEQ 0);                             ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4447  4
;    4448  4              MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;               ! PACKET SIZE                                  SEND DATA
;    4449  4              MSCP_PKT [.MX1, OPCODE] = OP_SDD;               ! OPCODE = SEND DATA
;    4450  4              MSCP_PKT [.MX1, BC_LO] = 6;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 6
;    4451  4              MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;    4452  4              DUPPRT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; !LOAD UNIT NUMBER (RDRX)
;    4453  4              DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;    4454  4              DUPPKT [DUPBF2] = .S_PATTERN;                   ! LOAD PATTERN
;    4455  4              MSCP_PKT [.MX1, MODIFY] = 0;                    !
;    4456  4  !ZZZ        MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;    4457  4              MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;               ! CALL ALL DUP CMDS SEQUENTIAL.                          ZZZ
;    4458  4              DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    4459  4
;    4460  4              IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  ! status error
;    4461  4              THEN RETURN;
;    4462  4
;    4463  5              DO (MX1 = GET_PKT (.CCTLR))
;    4464  4              UNTIL (.MX1 GEQ 0);                             ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4465  4
;    4466  4              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                  RECEIVE DATA
;    4467  4              MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
;    4468  4              MSCP_PKT [.MX1, BC_LO] = 4;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;    4469  4              MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;    4470  4              MSCP_PKT [.MX1, MODIFY] = 0;                    !
;    4471  4  !ZZZ        MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;    4472  4              MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;               ! CALL ALL DUP CMDS SEQUENTIAL.                          ZZZ
;    4473  4              DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    4474  4
;    4475  4
;    4476  4              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                        ZZZ
;    4477  4                 (.DUPPKT [DUPTYPE] EQL 3) AND                !NO dup type error                               ZZZ
;    4478  4                 (.DUPPKT [DUPMESG] EQL 3) AND                !                                                ZZZ
;    4479  5                 (.DUPPKT [DUPBF1]  EQL 0)                    !AND A successful write code                     ZZZ
;    4480  5                                                              !                                                ZZZ
```

```
;   4481  4          THEN                                          !THEN                                  ZZZ
;   4482  4             LEAVE DUP_WLOOP                            !I/O OK. EXIT RETRY LOOP.              ZZZ
;   4483  4
;   4484  4   !ZZZ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   4485  4   !        ELSE
;   4486  4   !        IF (.DUPPKT [DUPBF0] EQL 50003) AND        !IF DBN  IS OUT OF RANGE                !ZZZ
;   4487  4   !           (.DUPPKT [DUPBF1]  EQL  2)              !WD 0 = 2 FOR OUT OF RANGE STATUS.      ZZZ
;   4488  4   !        THEN                                      !THEN                                  ZZZ
;   4489  4   !           LEAVE DUP_WLOOP                         !EXIT RETRY LOOP.                       ZZZ
;   4490  4   !ZZZ ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;   4491  4
;   4492  4          ELSE                                          !                                      ZZZ
;   4493  5             BEGIN                                      !                                      ZZZ
;   4494  5             TRYNUM = .TRYNUM + 1;                      !INCR ATTEMPT COUNT                    ZZZ
;   4495  5             IF .TRYNUM  EQL .MAX_TRY_COUNT             !IF IT FAILED ALL RETRIES, THEN        ZZZ
;   4496  5             THEN                                      !REPORT THE ERROR.                     ZZZ
;   4497  6                   (D_FAIL = 1;                         !TELL HARD_ERROR IT WAS A DUP PROBLEM  ZZZ
;   4498  6                    HARD_ERROR ();                      !                                      ZZZ
;   4499  6                    D_FAIL = 0;                         !                                      ZZZ
;   4500  6                    CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;   ! SET FLAG                     ZZZ
;   4501  5                    RETURN;);                           ! NO POINT IN CONTINUING               ZZZ
;   4502  4             END;                                      !                                      ZZZ
;   4503  3    END;                                               !END LARGE DO LOOP                     ZZZ
;   4504  3                                                       !                                      ZZZ
;   4505  2   END;                                                !END DUP_WLOOP                         ZZZ
;   4506  2
;   4507  2
;   4508  3   DO (MX1 = GET_PKT (.CCTLR))
;   4509  2   UNTIL (.MX1 GEQ 0);                                 ! TRY TO GET AN ENVELOPE.
;   4510  2
;   4511  2   T_ADDR [T_BLK_WT] = .T_ADDR [T_BLK_WT] + 1;        !INCREMENT COUNTER IF A SUCCESS
;   4512  2
;   4513  1   END;


                                                .SBTTL  DUPWRTDBN MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 DUPWRTDBN::
                                                JSR     R1,$SAVE4                       ;                       4375
000004  012704  000011                          MOV     #11,R4                          ; *,MAX.TRY.COUNT       4389
000010  013700  000000G                         MOV     T.ADDR,R0                       ;                       4398
000014  005260  000060                          INC     60(R0)
000020  005002                                  CLR     R2                              ; TRYNUM                4400
000022  012703  000012                          MOV     #12,R3                          ; *,TRIES               4403
000026  013746  000110'                 1$:     MOV     MX1,-(SP)                       ;                       4407
000032  012746  000106                          MOV     #106,-(SP)
000036  004737  000000G                         JSR     PC,BL$MUL
000042  012760  000022  000006G                 MOV     #22,MSCP.PKT+6(R0)
000050  112760  000003  000022G                 MOVB    #3,MSCP.PKT+22(R0)              ;                       4408
000056  112760  000127  000026G                 MOVB    #127,MSCP.PKT+26(R0)            ;                       4409
000064  112760  000122  000027G                 MOVB    #122,MSCP.PKT+27(R0)            ;                       4410
000072  112760  000124  000030G                 MOVB    #124,MSCP.PKT+30(R0)            ;                       4411
000100  112760  000104  000031G                 MOVB    #104,MSCP.PKT+31(R0)            ;                       4412
000106  112760  000102  000032G                 MOVB    #102,MSCP.PKT+32(R0)            ;                       4413
```

M12


```
000114  112760  000116  000033G                 MOVB    #116,MSCP.PKT+33(R0)            ;                       4414
000122  012760  000001  000024G                 MOV     #1,MSCP.PKT+24(R0)              ;                       4414
000130  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                       4415
000136  004737  000000V                         JSR     PC,DUPCOMMAND                   ;                       4417
000142  013700  001250'                         MOV     DUOFF,R0                        ;                       4418
000146  006300                                  ASL     R0                              ;                       4420
000150  063700  000000G                         ADD     CST.ADDR,R0
000154  032760  040000  000020                 BIT     #40000,20(R0)
000162  001402                                  BEQ     2$
000164  022626                                  CMP     (SP)+,(SP)+
000166  000207                                  RTS     PC                              ;                       4375
000170  013716  000000G                 2$:     MOV     CCTLR,(SP)                      ;                       4421
000174  004737  000000G                         JSR     PC,GET.PKT                      ;                       4423
000200  010037  000110'                         MOV     R0,MX1
000204  002771                                  BLT     2$
000206  010016                                  MOV     R0,(SP)                         ;                       4424
000210  012746  000106                          MOV     #106,-(SP)                      ;                       4426
000214  004737  000000G                         JSR     PC,BL$MUL
000220  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000226  112760  000005  000022G                 MOVB    #5,MSCP.PKT+22(R0)
000234  012760  000120  000026G                 MOV     #120,MSCP.PKT+26(R0)            ;                       4427
000242  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4428
000250  005060  000024G                         CLR     MSCP.PKT+24(R0)                 ;                       4429
000254  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                       4430
000262  004737  000000V                         JSR     PC,DUPCOMMAND                   ;                       4432
000266  013700  001250'                         MOV     DUOFF,R0                        ;                       4433
000272  006300                                  ASL     R0                              ;                       4435
000274  063700  000000G                         ADD     CST.ADDR,R0
000300  032760  040000  000020                 BIT     #40000,20(R0)
000306  001004                                  BNE     3$
000310  023727  000000G 010006                  CMP     DUPPKT,#10006
000316  001422                                  BEQ     4$                              ;                       4436
000320  112737  000001  000000G         3$:     MOVB    #1,D.FAIL                       ;                       4439
000326  004737  000000V                         JSR     PC,HARD.ERROR                   ;                       4440
000332  105037  000000G                         CLRB    D.FAIL                          ;                       4441
000336  013700  001250'                         MOV     DUOFF,R0                        ;                       4442
000342  006300                                  ASL     R0
000344  063700  000000G                         ADD     CST.ADDR,R0
000350  052760  040000  000020                 BIS     #40000,20(R0)
000356  062706  000006                          ADD     #6,SP
000362  000207                                  RTS     PC                              ;                       4443
000364  013716  000000G                 4$:     MOV     CCTLR,(SP)                      ;                       4439
000370  004737  000000G                         JSR     PC,GET.PKT                      ;                       4445
000374  010037  000110'                         MOV     R0,MX1
000400  002771                                  BLT     4$
000402  010016                                  MOV     R0,(SP)                         ;                       4446
000404  012746  000106                          MOV     #106,-(SP)                      ;                       4448
000410  004737  000000G                         JSR     PC,BL$MUL
000414  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000422  112760  000004  000022G                 MOVB    #4,MSCP.PKT+22(R0)
000430  012760  000006  000026G                 MOV     #6,MSCP.PKT+26(R0)              ;                       4449
000436  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4450
000444  013701  001250'                         MOV     DUOFF,R1                        ;                       4451
                                                                                        ;                       4452
```

N12


```
000450  006301                          ASL     R1
000452  063701  000000G                 ADD     CST.ADDR,R1
000456  111137  000000G                 MOVB    (R1),DUPPKT
000462  042737  177760  000000G         BIC     #177760,DUPPKT
000470  013701  001250'                 MOV     DUOFF,R1                ;                       4453
000474  006301                          ASL     R1
000476  063701  000000G                 ADD     CST.ADDR,R1
000502  116137  000020  000002G         MOVB    20(R1),DUPPKT+2
000510  105037  000003G                 CLRB    DUPPKT+3
000514  013737  000000G  000004G        MOV     S.PATTERN,DUPPKT+4      ;                       4454
000522  005060  000024G                 CLR     MSCP.PKT+24(R0)         ;                       4455
000526  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)    ;                       4457
000534  004737  000000V                 JSR     PC,DUPCOMMAND           ;                       4458
000540  013700  001250'                 MOV     DUOFF,R0                ;                       4460
000544  006300                          ASL     R0
000546  063700  000000G                 ADD     CST.ADDR,R0
000552  032760  040000  000020         BIT     #40000,20(R0)
000560  001403                          BEQ     5$
000562  062706  000010                  ADD     #10,SP
000566  000207                          RTS     PC                      ;                       4375
000570  013716  000000G          5$:    MOV     CCTLR,(SP)              ;                       4461
000574  004737  000000G                 JSR     PC,GET.PKT              ;                       4463
000600  010037  000110'                 MOV     R0,MX1
000604  002771                          BLT     5$
000606  010016                          MOV     R0,(SP)                 ;                       4464
000610  012746  000106                  MOV     #106,-(SP)              ;                       4466
000614  004737  000000G                 JSR     PC,BL$MUL
000620  012760  000034  000006G         MOV     #34,MSCP.PKT+6(R0)
000626  112760  000005  000022G         MOVB    #5,MSCP.PKT+22(R0)
000634  012760  000004  000026G         MOV     #4,MSCP.PKT+26(R0)      ;                       4467
000642  012760  000000G  000032G         MOV     #DUPPKT,MSCP.PKT+32(R0) ;                       4468
000650  005060  000024G                 CLR     MSCP.PKT+24(R0)         ;                       4469
000654  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)    ;                       4470
000662  004737  000000V                 JSR     PC,DUPCOMMAND           ;                       4472
000666  013700  001250'                 MOV     DUOFF,R0                ;                       4473
000672  006300                          ASL     R0                      ;                       4476
000674  063700  000000G                 ADD     CST.ADDR,R0
000700  032760  040000  000020         BIT     #40000,20(R0)
000706  001012                          BNE     6$
000710  023727  000000G  030003         CMP     DUPPKT,#30003           ;                       4477
000716  001006                          BNE     6$
000720  005737  000002G                 TST     DUPPKT+2                ;                       4479
000724  001003                          BNE     6$
000726  062706  000012                  ADD     #12,SP                  ;                       4482
000732  000433                          BR      8$
000734  005202                  6$:     INC     R2                      ; TRYNUM                4494
000736  020204                          CMP     R2,R4                   ; TRYNUM,MAX.TRY.COUNT  4495
000740  001022                          BNE     7$
000742  112737  000001  000000G         MOVB    #1,D.FAIL               ;                       4497
000750  004737  000000V                 JSR     PC,HARD.ERROR           ;                       4498
000754  105037  000000G                 CLRB    D.FAIL                  ;                       4499
000760  013700  001250'                 MOV     DUOFF,R0                ;                       4500
000764  006300                          ASL     R0
```

B13


```
000766  063700  000000G                        ADD     CST.ADDR,R0
000772  052760  040000  000020                BIS     #40000,20(R0)
001000  062706  000012                        ADD     #12,SP                          ;                       4501
001004  000207                                RTS     PC                              ;                       4497
001006  062706  000012                7$:     ADD     #12,SP                          ;                       4404
001012  005303                                DEC     R3                              ; TRIES                 4403
001014  001402                                BEQ     8$
001016  000137  015106'                        JMP     1$
001022  013746  000000G               8$:     MOV     CCTLR,-(SP)                     ;                       4508
001026  004737  000000G                        JSR     PC,GET.PKT
001032  010037  000110'                        MOV     R0,MX1
001036  005726                                TST     (SP)+
001040  005700                                TST     R0                              ; MX1                   4509
001042  002767                                BLT     8$
001044  013700  000000G                        MOV     T.ADDR,R0                       ;                       4511
001050  005260  000056                        INC     56(R0)
001054  000207                                RTS     PC                              ;                       4375

; Routine Size:  279 words,    Routine Base:  $CODE$ + 15060
; Maximum stack depth per invocation:  11 words
```

```
1   GLOBAL ROUTINE DUPREDDBN : NOVALUE =
1
1   !+
1   ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
1   ! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
1   ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
1   ! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
1   ! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
1   ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
1   !
1   !
1   !       IMPLICIT INPUTS:
1   !               CST_ADDR    - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
1   !               DUOFF       - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
1   !-
2   BEGIN
2   LOCAL
2       TRYNUM : WORD,                                  !                                       ZZZ
2       MAX_TRY_COUNT : word initial (9);               ! MAXIMUM NUMBER OF RETRIES BEFORE ERROR ZZZ
2                                                       !                                       ZZZ
2   LABEL                                               !                                       ZZZ
2       DUP_RLOOP;                                      !START OR DUP READ RETRY LOOP           ZZZ
2
2
2   !PRINTX (DER12);
2   T_ADDR [T_DBN_RD] = .T_ADDR [T_DBN_RD] + 1;         ! INCREMENT # OF READS GIVEN
2
2   TRYNUM = 0;                                         !ZERO TRY COUNTER                       ZZZ
2   DUP_RLOOP:                                          !LABEL FOR LOOP EXCAPE ON GOOD READ     ZZZ
3   BEGIN                                               !BEGIN DUP_RLOOP                        ZZZ
3   INCR TRIES FROM 1 TO 10 DO                          !START TRYING DUP READS                 ZZZ
4   BEGIN                                               !BEGIN LARGE DO LOOP                    ZZZ
4                                                       !                                       ZZZ
4
4    MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                  ! PACKET SIZE                          EXECUTE REDDBN PROGRAM
4    MSCP_PKT [.MX1, OPCODE] = OP_ELP;                  ! OPCODE = EXECUTE LOCAL PROGRAM
4    MSCP_PKT [.MX1, L1] = %ascii'R';                   ! FILL IN PROGRAM NAME WITH ASCII LETTERS
4    MSCP_PKT [.MX1, L2] = %ascii'E';
4    MSCP_PKT [.MX1, L3] = %ascii'D';
4    MSCP_PKT [.MX1, L4] = %ascii'D';
4    MSCP_PKT [.MX1, L5] = %ascii'B';
4    MSCP_PKT [.MX1, L6] = %ascii'N';
4    MSCP_PKT [.MX1, MODIFY] = 1;                       ! STANDALONE MODIFIER
4   !ZZZ  MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                    ! CALL IT IMMEDIATE
4    MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                  ! CALL ALL DUP CMDS SEQUENTIAL.                ZZZ
4    DUPCOMMAND ();                                     ! SENDS AND RECEIVES THE COMMAND
4
4    IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          !status error
4    THEN RETURN;
4
5    DO (MX1 = GET_PKT (.CCTLR))
4    UNTIL (.MX1 GEQ 0);                        ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
4
```

```
;  4567  4                MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                                    RECIEVE DATA
;  4568  4                MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = RECEIVE DATA
;  4569  4                MSCP_PKT [.MX1, BC_LO] = 80;                     ! BYTE COUNT TO BE TRANSFERED EQUALS 2 *****see pg 26 DUP spe
;  4570  4                MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4571  4                MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4572  4      !ZZZ      MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4573  4                MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.                  ZZZ
;  4574  4                DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4575  4
;  4576  4                IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;  4577  4                   (.DUPPKT [DUPTYPE] NEQU 1) OR                 !dup type error
;  4578  5                   (.DUPPKT [DUPMESG] NEQU 5)
;  4579  4                THEN
;  4580  5                        (D_FAIL = 1;                             !TELL HARD_ERROR IT WAS A DUP PROBLEM            ZZZ
;  4581  5                         HARD_ERROR (.);
;  4582  5                         D_FAIL = 0;                             !                                                ZZZ
;  4583  5                         CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;       ! SET FLAG
;  4584  4                         RETURN;);                               ! NO POINT IN CONTINUING
;  4585  4
;  4586  5                DO (MX1 = GET_PKT (.CCTLR))
;  4587  4                UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4588  4
;  4589  4                MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;                ! PACKET SIZE                                    SEND DATA
;  4590  4                MSCP_PKT [.MX1, OPCODE] = OP_SDD;                ! OPCODE = SEND DATA
;  4591  4                MSCP_PKT [.MX1, BC_LO] = 4;                      ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;  4592  4                MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4593  4                DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; ! LOAD UNIT NUMBER (RDRX)
;  4594  4                DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;  4595  4                MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4596  4      !ZZZ      MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4597  4                MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.                  ZZZ
;  4598  4                DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4599  4
;  4600  4                 IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1          !status error
;  4601  4                 THEN RETURN;
;  4602  4
;  4603  5                DO (MX1 = GET_PKT (.CCTLR))
;  4604  4                UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;  4605  4
;  4606  4                MSCP_PKT [.MX1, MSGLEN] = SZ_REC;                ! PACKET SIZE                                    RECEIVE DATA
;  4607  4                MSCP_PKT [.MX1, OPCODE] = OP_RCD;                ! OPCODE = GET DUST STATUS
;  4608  4                MSCP_PKT [.MX1, BC_LO] = 514;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 512
;  4609  4                MSCP_PKT [.MX1, BUF_0] = DUPPKT;                 ! LOAD DESCRIBTOR BUFFER
;  4610  4                MSCP_PKT [.MX1, MODIFY] = 0;                     !
;  4611  4      !ZZZ      MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;               ! CALL IT sequential
;  4612  4                MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                ! CALL ALL DUP CMDS SEQUENTIAL.                  ZZZ
;  4613  4                DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;  4614  4
;  4615  4                IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 0) AND    !IF status OK AND                ZZZ
;  4616  4                   (.DUPPKT [DUPTYPE] EQL 6) AND                 !NO dup type error               ZZZ
;  4617  5                   (.DUPPKT [DUPMESG] EQL 2)                     !                                ZZZ
;  4618  4                THEN                                             !THEN                            ZZZ
;  4619  4                   LEAVE DUP_RLOOP                               !I/O OK. EXIT RETRY LOOP.        ZZZ
```

E17


```
;  4620  4
;  4621  4     !ZZZ ;;;
```

F13


```
000136  004737  000000V                        JSR     PC,DUPCOMMAND
000142  013700  001250'                        MOV     DUOFF,R0                ;                       4559
000146  006300                                  ASL     R0                      ;                       4561
000150  063700  000000G                        ADD     CST.ADDR,R0
000154  032760  040000  000020                 BIT     #40000,20(R0)
000162  001402                                  BEQ     2$
000164  022626                                  CMP     (SP)+,(SP)+
000166  000207                                  RTS     PC                      ;                       4514
000170  013716  000000G                2$:     MOV     CCTLR,(SP)              ;                       4562
000174  004737  000000G                        JSR     PC,GET.PKT              ;                       4564
000200  010037  000110'                        MOV     R0,MX1
000204  002771                                  BLT     2$
000206  010016                                  MOV     R0,(SP)                 ;                       4565
000210  012746  000106                        MOV     #106,-(SP)              ;                       4567
000214  004737  000000G                        JSR     PC,BL$MUL
000220  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000226  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)
000234  012760  000120  000026G                MOV     #120,MSCP.PKT+26(R0)    ;                       4568
000242  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0) ;                       4569
000250  005060  000024G                        CLR     MSCP.PKT+24(R0)         ;                       4570
000254  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)    ;                       4571
000262  004737  000000V                        JSR     PC,DUPCOMMAND           ;                       4573
000266  013700  001250'                        MOV     DUOFF,R0                ;                       4574
000272  006300                                  ASL     R0                      ;                       4576
000274  063700  000000G                        ADD     CST.ADDR,R0
000300  032760  040000  000020                 BIT     #40000,20(R0)
000306  001004                                  BNE     3$
000310  023727  000000G 010005                 CMP     DUPPKT,#10005
000316  001422                                  BEQ     4$                      ;                       4577
000320  112737  000001  000000G        3$:     MOVB    #1,D.FAIL
000326  004737  000000V                        JSR     PC,HARD.ERROR           ;                       4580
000332  105037  000000G                        CLRB    D.FAIL                  ;                       4581
000336  013700  001250'                        MOV     DUOFF,R0                ;                       4582
000342  006300                                  ASL     R0                      ;                       4583
000344  063700  000000G                        ADD     CST.ADDR,R0
000350  052760  040000  000020                 BIS     #40000,20(R0)
000356  062706  000006                        ADD     #6,SP
000362  000207                                  RTS     PC                      ;                       4584
000364  013716  000000G                4$:     MOV     CCTLR,(SP)              ;                       4580
000370  004737  000000G                        JSR     PC,GET.PKT              ;                       4586
000374  010037  000110'                        MOV     R0,MX1
000400  002771                                  BLT     4$
000402  010016                                  MOV     R0,(SP)                 ;                       4587
000404  012746  000106                        MOV     #106,-(SP)              ;                       4589
000410  004737  000000G                        JSR     PC,BL$MUL
000414  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000422  112760  000004  000022G                MOVB    #4,MSCP.PKT+22(R0)
000430  012760  000004  000026G                MOV     #4,MSCP.PKT+26(R0)      ;                       4590
000436  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0) ;                       4591
000444  013701  001250'                        MOV     DUOFF,R1                ;                       4592
000450  006301                                  ASL     R1                      ;                       4593
000452  063701  000000G                        ADD     CST.ADDR,R1
000456  111137  000000G                        MOVB    (R1),DUPPKT
```

```
000462  042737  177760  000000G                 BIC     #177760,DUPPKT
000470  013701  001250'                         MOV     DUOFF,R1
000474  006301                                  ASL     R1                              ;                       4594
000476  063701  000000G                         ADD     CST.ADDR,R1
000502  116137  000020  000002G                 MOVB    20(R1),DUPPKT+2
000510  105037  000003G                         CLRB    DUPPKT+3
000514  005060  000024G                         CLR     MSCP.PKT+24(R0)
000520  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                       4595
000526  004737  000000V                         JSR     PC,DUPCOMMAND                   ;                       4597
000532  013700  001250'                         MOV     DUOFF,R0                        ;                       4598
000536  006300                                  ASL     R0                              ;                       4600
000540  063700  000000G                         ADD     CST.ADDR,R0
000544  032760  040000  000020                  BIT     #40000,20(R0)
000552  001403                                  BEQ     5$
000554  062706  000010                          ADD     #10,SP
000560  000207                                  RTS     PC                              ;                       4514
000562  013716  000000G                 5$:     MOV     CCTLR,(SP)                      ;                       4601
000566  004737  000000G                         JSR     PC,GET.PKT                      ;                       4603
000572  010037  000110'                         MOV     R0,MX1
000576  002771                                  BLT     5$
000600  010016                                  MOV     R0,(SP)                         ;                       4604
000602  012746  000106                          MOV     #106,-(SP)                      ;                       4606
000606  004737  000000G                         JSR     PC,BL$MUL
000612  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000620  112760  000005  000022G                 MOVB    #5,MSCP.PKT+22(R0)
000626  012760  001002  000026G                 MOV     #1002,MSCP.PKT+26(R0)           ;                       4607
000634  012760  000000G 000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4608
000642  005060  000024G                         CLR     MSCP.PKT+24(R0)                 ;                       4609
000646  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)            ;                       4610
000654  004737  000000V                         JSR     PC,DUPCOMMAND                   ;                       4612
000660  013700  001250'                         MOV     DUOFF,R0                        ;                       4613
000664  006300                                  ASL     R0                              ;                       4615
000666  063700  000000G                         ADD     CST.ADDR,R0
000672  032760  040000  000020                  BIT     #40000,20(R0)
000700  001007                                  BNE     6$
000702  023727  000000G 060002                  CMP     DUPPKT,#60002                   ;                       4616
000710  001003                                  BNE     6$
000712  062706  000012                          ADD     #12,SP                          ;                       4619
000716  000433                                  BR      8$
000720  005202                          6$:     INC     R2                              ; TRYNUM                4631
000722  020204                                  CMP     R2,R4                           ; TRYNUM,MAX.TRY.COUNT  4632
000724  001022                                  BNE     7$
000726  112737  000001  000000G                 MOVB    #1,D.FAIL                       ;                       4634
000734  004737  000000V                         JSR     PC,HARD.ERROR                   ;                       4635
000740  105037  000000G                         CLRB    D.FAIL                          ;                       4636
000744  013700  001250'                         MOV     DUOFF,R0                        ;                       4637
000750  006300                                  ASL     R0
000752  063700  000000G                         ADD     CST.ADDR,R0
000756  052760  040000  000020                  BIS     #40000,20(R0)
000764  062706  000012                          ADD     #12,SP                          ;                       4638
000770  000207                                  RTS     PC                              ;                       4634
000772  062706  000012                  7$:     ADD     #12,SP                          ;                       4545
000776  005303                                  DEC     R3                              ; TRIES                 4544
```

H13


```
001000  001402                          BEQ     8$
001002  000137  016164'                 JMP     1$
001006  013746  000000G         8$:     MOV     CCTLR,-(SP)             ;                       4644
001012  004737  000000G                 JSR     PC,GET.PKT
001016  010037  000110'                 MOV     R0,MX1
001022  005726                          TST     (SP)+
001024  005700                          TST     R0                      ; MX1                   4645
001026  002767                          BLT     8$
001030  013700  000000G                 MOV     T.ADDR,R0               ;                       4647
001034  005260  000062                  INC     62(R0)
001040  000207                          RTS     PC                      ;                       4514

; Routine Size:  273 words,    Routine Base:  $CODE$ + 16136
; Maximum stack depth per invocation:  11 words

;   4650  1
```

```
;  4651  1
;  4652  1     GLOBAL ROUTINE DUPCOMMAND : NOVALUE =
;  4653  1
;  4654  1     !+
;  4655  1     ! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
;  4656  1     ! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
;  4657  1     ! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
;  4658  1     ! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
;  4659  1     !-
;  4660  2     BEGIN
;  4661  2     !PRINTX (DER13);
;  4662  2
;  4663  2     MSCP_PKT [.MX1, CREDITS] = 0;                   ! DUP DOES NOT USE THE CREDIT SYSTEM
;  4664  2     MSCP_PKT [.MX1, CONNID] = CID_DUP;              ! MAKE PACKAGE EQUAL A DUP COMMAND
;  4665  2     MSCP_PKT [.MX1, DK_NUM] = 0;                    ! DISK NUMBER (NOT APPLICABLE)
;  4666  2
;  4667  2     IF SEND (.MX1) EQLU FAILURE                     ! ATTEMPT SEND; IF CTLR IS OFFLINE
;  4668  2     THEN
;  4669  3             BEGIN
;  4670  3             PUT_PKT (.MX1);
;  4671  3             MX1 = -1;                               ! RETURN ENVELOPE TO POOL
;  4672  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;
;  4673  3     !       PRINTF (DBM112);                        ! "DUP: PKT NOT AVAILABLE"  ZZZ
;  4674  3             END
;  4675  3
;  4676  2     ELSE
;  4677  2             do
;  4678  3                     begin
;  4679  3                     BREAK;                          ! BREAK FOR ACT
;  4680  3                     PROC_RETPKT ();                         ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;  4681  3                     end
;  4682  2             until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;  4683  2             (.EOP_FLAG eql true);                   ! or end of pass caused by error
;  4684  1     END;


                                .SBTTL  DUPCOMMAND MULTI-DRIVE TEST ROUTINES
000000  013746  000110'    DUPCOMMAND::
                            MOV     MX1,-(SP)                       ;                       4663
000004  012746  000106      MOV     #106,-(SP)
000010  004737  000000G     JSR     PC,BL$MUL
000014  142760  000017  000010G   BICB    #17,MSCP.PKT+10(R0)
000022  112760  000002  000011G   MOVB    #2,MSCP.PKT+11(R0)      ;                       4664
000030  005060  000016G     CLR     MSCP.PKT+16(R0)                 ;                       4665
000034  013716  000110'     MOV     MX1,(SP)                        ;                       4667
000040  004737  000000G     JSR     PC,SEND
000044  005700              TST     R0
000046  001020              BNE     1$
000050  013716  000110'     MOV     MX1,(SP)                        ;                       4670
000054  004737  000000G     JSR     PC,PUT.PKT
000060  012737  177777  000110'   MOV     #-1,MX1                 ;                       4671
000066  013700  001250'     MOV     DUOFF,R0                        ;                       4672
000072  006300              ASL     R0
```

J13


```
000074  063700  000000G                        ADD     CST.ADDR,R0
000100  052760  040000  000020                 BIS     #40000,20(R0)
000106  000415                                 BR      2$
000110  104422                          1$:    TRAP    22                      ;                       4667
000112  004737  000000V                        JSR     PC,PROC.RETPKT          ;                       4678
000116  013700  000000G                        MOV     RP.ADDR,R0              ;                       4680
000122  023760  000000G 000004                 CMP     CRN.LOW,4(R0)           ;                       4682
000130  001404                                 BEQ     2$
000132  123727  000000G 000001                 CMPB    EOP.FLAG,#1             ;
000140  001363                                 BNE     1$                                              4683
000142  022626                          2$:    CMP     (SP)+,(SP)+             ;
000144  000207                                 RTS     PC                      ;                       4660
                                                                             ;                       4652

; Routine Size:  51 words,      Routine Base:  $CODE$ + 17200
; Maximum stack depth per invocation:  4 words
```

K13


```
;  4685  1
;  4686  1     GLOBAL ROUTINE DUPIDLE : NOVALUE =
;  4687  1     !+
;  4688  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO INSURE THAT THE CONTROLLER
;  4689  1     ! IS NOT IN A ACTIVE STATE. IF CALLED AND THE CONTROLLER IS IN AN ACTIVE
;  4690  1     ! STATE THE CONTROLLER WILL GIVE AN ABORT COMMAND WHICH SHOULD KILL THE
;  4691  1     ! CURRENT JOB OR LOCAL PROGRAM.
;  4692  1     !-
;  4693  2     BEGIN
;  4694  2     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 0;                !CLEAR DUP ERROR FLAG;
;  4695  2
;  4696  2     MSCP_PKT [.MX1, MSGLEN] = SZ_GDS;                 ! PACKET SIZE                          GET DUST STATUS
;  4697  2     MSCP_PKT [.MX1, OPCODE] = OP_GDS;                 ! OPCODE = GET DUST STATUS
;  4698  2     MSCP_PKT [.MX1, MODIFY] = 0;                      !
;  4699  2     !ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;           ! CALL IT IMMEDIATE
;  4700  2     MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                 ! CALL ALL DUP CMDS SEQUENTIAL.                ZZZ
;  4701  2     DUPCOMMAND ();                                    ! SENDS AND RECEIVES THE COMMAND
;  4702  2                                                       ! GDS ONLY RETURNS SUCCESS or it don't return
;  4703  2
;  4704  3     DO (MX1 = GET_PKT (.CCTLR))
;  4705  2     UNTIL (.MX1 GEQ 0);                               ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;  4706  2
;  4707  2     if .CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] neq IDLE        ! if not in idle state then abort the program
;  4708  2     then
;  4709  3         begin
;  4710  3         MSCP_PKT [.MX1, MSGLEN] = SZ_ABT;                 ! PACKET SIZE                          ABORT CMD
;  4711  3         MSCP_PKT [.MX1, OPCODE] = OP_ABT;                 ! OPCODE = ABORT PROGRAM
;  4712  3         MSCP_PKT [.MX1, MODIFY] = 0;                      !
;  4713  3     !ZZZ MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;               ! CALL IT IMMEDIATE
;  4714  3         MSCP_PKT [.MX1, MSGTYP] = MT_SEQ;                 ! CALL ALL DUP CMDS SEQUENTIAL.                ZZZ
;  4715  3         DUPCOMMAND ();                                    ! SENDS AND RECEIVES THE COMMAND
;  4716  3                                                           !ONLY ERROR IS already in idle state
;  4717  4         DO (MX1 = GET_PKT (.CCTLR))
;  4718  3         UNTIL (.MX1 GEQ 0);                           ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;  4719  2         end;
;  4720  1     end;


                                        .SBTTL  DUPIDLE MULTI-DRIVE TEST ROUTINES
000000  010146                  DUPIDLE::
                                        MOV     R1,-(SP)                        ;                                  4686
000002  013700  001250'                 MOV     DUOFF,R0                        ;                                  4694
000006  006300                          ASL     R0
000010  063700  000000G                 ADD     CST.ADDR,R0
000014  042760  040000  000020          BIC     #40000,20(R0)
000022  013746  000110'                 MOV     MX1,-(SP)                       ;                                  4696
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)              ;                                  4697
000052  005060  000024G                 CLR     MSCP.PKT+24(R0)                 ;                                  4698
000056  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)            ;                                  4700
000064  004737  017200'                 JSR     PC,DUPCOMMAND                   ;                                  4701
```

L13


```
000070  013716  000000G                  1$:     MOV     CCTLR,(SP)                      ;                                       4704
000074  004737  000000G                          JSR     PC,GET.PKT
000100  010037  000110'                          MOV     R0,MX1
000104  010001                                   MOV     R0,R1                           ; MX1,*                                 4705
000106  002770                                   BLT     1$
000110  013700  001250'                          MOV     DUOFF,R0                        ;                                       4707
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)                         ;                                       4710
000134  012746  000106                           MOV     #106,-(SP)
000140  004737  000000G                          JSR     PC,BL$MUL
000144  012760  000014  000006G                  MOV     #14,MSCP.PKT+6(R0)
000152  112760  000006  000022G                  MOVB    #6,MSCP.PKT+22(R0)              ;                                       4711
000160  005060  000024G                          CLR     MSCP.PKT+24(R0)                 ;                                       4712
000164  142760  000360  000010G                  BICB    #360,MSCP.PKT+10(R0)            ;                                       4714
000172  004737  017200'                          JSR     PC,DUPCOMMAND                   ;                                       4715
000176  013716  000000G                  2$:     MOV     CCTLR,(SP)                      ;                                       4717
000202  004737  000000G                          JSR     PC,GET.PKT
000206  010037  000110'                          MOV     R0,MX1
000212  002771                                   BLT     2$                              ;                                       4718
000214  005726                                   TST     (SP)+                           ;                                       4709
000216  022626                          3$:     CMP     (SP)+,(SP)+                     ;                                       4693
000220  012601                                   MOV     (SP)+,R1                        ;                                       4686
000222  000207                                   RTS     PC

; Routine Size:  74 words,      Routine Base:  $CODE$ + 17346
; Maximum stack depth per invocation:  5 words
```

```
;    4721  1     GLOBAL routine QIO_LBN : novalue =
;    4722  1
;    4723  1     !+
;    4724  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
;    4725  1     !       BE USED FOR THE CURRENT QIO OR QIO PAIR.
;    4726  1     !
;    4727  1     !       IF THE OPERATOR CHOSE THE RANDOM SEEK MODE OPTION, THEN THE LBN IS
;    4728  1     !       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR THE LBN.
;    4729  1     !       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
;    4730  1     !       TABLE (BST).
;    4731  1     !
;    4732  1     !       IMPLICIT INPUTS:
;    4733  1     !               L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
;    4734  1     !
;    4735  1     !       IMPLICIT OUTPUTS:
;    4736  1     !               THE LBN IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4737  1     !-
;    4738  1
;    4739  2         begin
;    4740  2
;    4741  2         own
;ZZ4742  2             LBN0_SAVE : word initial (0),                   !LO LBN SELECTED IN PREVIOUS PASS
;ZZ4743  2             LBN1_SAVE : word initial (0);                   !HI LBN SELECTED IN PREVIOUS PASS
;    4744  2
;    4745  2         local
;ZZ4746  2             S0_TEMP : word,                                 ! TEMPORARY STORAGE FOR START LBN LO
;ZZ4747  2             S1_TEMP : word,                                 ! TEMPORARY STORAGE FOR START LBN HI
;ZZ4748  2             E0_TEMP : word,                                 ! TEMPORARY STORAGE FOR END LBN LO
;ZZ4749  2             E1_TEMP : word,                                 ! TEMPORARY STORAGE FOR END LBN HI
;ZZ4750  2             ADD0_LBN : word,                                ! TEMPORARY STORAGE USED FOR COMPUTING DESIRED LBN LO
;ZZ4751  2             ADD1_LBN : word,                                ! TEMPORARY STORAGE USED FOR COMPUTING DESIRED LBN HI
;ZZ4752  2             LBN0 : word,                                    ! LOGICAL BLOCK NUMBER LO
;ZZ4753  2             LBN1 : word,                                    ! LOGICAL BLOCK NUMBER HI
;    4754  2             WINCHESTER : byte initial (byte (TRUE));        ! FLAG TO INDICATE WINCHESTER DISK SELECTED
;    4755  2
;    4756  2         label
;    4757  2             FIND_LBN;
;    4758  2
;ZZ4759  2         S0_TEMP = .CST_ADDR [.CUOFF + OF_BEG, D_BEG0];      ! STARTING LBN LO
;ZZ4760  2         S1_TEMP = .CST_ADDR [.CUOFF + OF_BEG1, D_BEG1];     ! STARTING LBN HI
;ZZ4761  2         E0_TEMP = .CST_ADDR [.CUOFF + OF_END, D_END0];      ! ENDING LBN LO
;ZZ4762  2         E1_TEMP = .CST_ADDR [.CUOFF + OF_END1, D_END1];     ! ENDING LBN HI
;    4763  2
;    4764  2         FIND_LBN:
;ZZ4765  3             begin                                           !BEGIN A.
;    4766  3
;    4767  3             if (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and
;    4768  3                (BIT_TST (SWP_FLAGS, SWF_FER)) and
;    4769  3                (.MAD1 [OPCODE] eql OP_WRT) and
;    4770  4                (.FORCED_ERROR)
;    4771  3             then
;    4772  4                 begin
;ZZZ4773  4                 LBN0= .FER0_LBN;                            ! IF "FORCED ERROR" DETECTED, REWRITE ERROR LBN LO
```

```
;ZZZ4774  4                 LBN1= .FER1_LBN;                                    ! IF "FORCED ERROR" DETECTED, REWRITE ERROR LBN HI
;   4775  4                 leave FIND_LBN;
;   4776  3                 end;
;   4777  3
;   4778  3             if .CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql REMOVABLE
;   4779  3             then
;   4780  3                 WINCHESTER = FALSE;
;   4781  3
;   4782  4             if BIT_TST (SWP_FLAGS, SWF_RDM)                         ! IF RANDOM SEEK MODE
;   4783  4             then
;   4784  4
;   4785  4     !!!!
```

```
;  4827  4                    end;
;  4828  4                end
;  4829  4
;  4830  3            ELSE                                                    !
;  4831  4            begin                                                   ! ELSE - SEQUENTIAL LBN MODE  (BEGIN A)
;  4832  4                LBN0 = .BST [.L$LUN, LO_WRD];                       ! GET LBN FROM BST (LO WORD)
;  4833  4                LBN1 = .BST [.L$LUN, HI_WRD];                       ! GET LBN FROM BST (HI WORD)
;  4834  4
;  4835  4                IF .TRK_SGN [.L$LUN] EQLU 1                         !IF WE WANT SERIAL INCREMENT
;  4836  4                THEN                                                !
;  4837  5                BEGIN                                               !(BEGIN B)
;  4838  5                    IF .BST [.L$LUN, LO_WRD] EQLU %o'177777'        !IF OVERFLOW FROM LO WD TO HI WD
;  4839  5                    THEN                                            !
;  4840  6                    BEGIN
;  4841  6                        BST [.L$LUN, LO_WRD] = 0;                   !ZERO LO WORD
;  4842  6                        BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1;       !INCREMENT HI WORD
;  4843  6                    END                                             !
;  4844  5                    ELSE
;  4845  5                        BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] + 1;       !OTHERWISE JUST INCR LO WORD
;  4846  5                                                                    !
;  4847  5
;  4848  6                    IF (.BST [.L$LUN, HI_WRD] GTRU .E1_TEMP)        !NOW TAKE CARE OF OVERFLOW WHILE INCREMENTING
;LT 4849  7                      OR ((.BST [.L$LUN, HI_WRD] EQLU .E1_TEMP)     !IF LBN1 OVER HI LIMIT
;  4850  6                      AND (.BST [.L$LUN, LO_WRD] GTRU .E0_TEMP))    !OR LBN1 EQUALS HI LIMIT AND LBN0 IS OVER LIM
;  4851  5                    THEN                                            !
;  4852  6                        BEGIN                                       !
;  4853  6                        BST [.L$LUN, LO_WRD] = .E0_TEMP;            !THEN SET HI LIMITS
;  4854  6                        BST [.L$LUN, HI_WRD] = .E1_TEMP;            !INTO BST FOR NEXT TIME
;  4855  6                        TRK_SGN [.L$LUN] = - 1;                     !AND REVERSE DIRECTION
;  4856  5                        END;                                        !
;  4857  5            END                                                     !(END B)
;  4858  5
;  4859  5
;  4860  5
;  4861  4            ELSE                                                    !IF WE WANT SERIAL DECREMENT
;  4862  5            BEGIN                                                   !(BEGIN C)
;  4863  5                IF .BST [.L$LUN, LO_WRD] EQLU 0                     !IF NEED TO BORROW FROM HI WD
;  4864  5                THEN                                                !
;  4865  6                BEGIN
;  4866  6                    BST [.L$LUN, LO_WRD] = %o'177777';              !LO WORD
;  4867  6                    BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] - 1;       !DECREMENT HI WORD
;  4868  6                END                                                 !
;  4869  5                ELSE
;  4870  5                    BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] - 1;       !OTHERWISE JUST DECR LO WORD
;  4871  5                                                                    !
;G. 4872  5
;  4873  6                IF (.BST [.L$LUN, HI_WRD] LSS  .S1_TEMP)            !NOW TAKE CARE OF UNDERFLOW WHILE INCREMENTIN
;  4874  7                  OR ((.BST [.L$LUN, HI_WRD] EQLU .S1_TEMP)         !IF LBN1 IS BELOW LIMIT
;  4875  6                  AND (.BST [.L$LUN, LO_WRD] LSSU .S0_TEMP))        !OR LBN1 EQUALS LO LIMIT AND LBN0 IS BELOW
;  4876  5                THEN                                                !
;  4877  6                    BEGIN                                           !
;  4878  6                    BST [.L$LUN, LO_WRD] = .S0_TEMP;                !THEN SET LO LIMITS
;  4879  6                    BST [.L$LUN, HI_WRD] = .S1_TEMP;                !INTO BST FOR NEXT TIME
```

```
;  4880  6                    TRK_SGN [.L$LUN] = + 1;                                  !AND REVERSE DIRECTION
;  4881  5                    END;                                                     !
;  4882  4              END;                                                           !
;  4883  3          END;                                                               !END C.
;  4884  3                                                                             !
;  4885  2        END;                                                                 !END A.
;  4886  2
;  4887  2
;  4888  3      IF ((.S1_TEMP EQLU .E1_TEMP) AND (.S0_TEMP EQLU .E0_TEMP))             !IF START ADDR SAME AS END ADDR
;  4889  2      THEN                                                                    !JUST USE THE START ADDRESS.
;  4890  3         BEGIN                                                                !
;  4891  3         LBN0 = .S0_TEMP;                                                     !
;  4892  3         LBN1 = .S1_TEMP;                                                     !
;  4893  2         END;                                                                 !
;  4894  2
;  4895  2      MAD1 [LBN_L] = .LBN0;                                                 ! LOAD LBN INTO 1ST PACKET
;  4896  2      MAD1 [LBN_H] = .LBN1;                                                 ! LOAD LBN INTO 1ST PACKET
;  4897  2
;  4898  2      if .MX2 geq 0                                                         ! IF 2 QIOs
;  4899  2      then
;  4900  2          MAD2 [LBN_L] = .LBN0;                                             ! LOAD LBN INTO 2ND PACKET
;  4901  2          MAD2 [LBN_H] = .LBN1;                                             ! LOAD LBN INTO 2ND PACKET
;  4902  2
;  4903  2      LBN0_SAVE = .LBN0;                                                    ! SAVE FOR USE NEXT CYCLE IF NEEDED
;  4904  2      LBN1_SAVE = .LBN1;                                                    ! SAVE FOR USE NEXT CYCLE IF NEEDED
;  4905  2
;  4906  1      end;                                                                  ! ROUTINE QIO_LBN


001302                                  .PSECT  $GGG$,  R0
001302  000000                  LBN0.SAVE:
                                        .WORD   0
001304  000000                  LBN1.SAVE:
                                        .WORD   0


                                        .SBTTL  QIO.LBN MULTI-DRIVE TEST ROUTINES
017572                                  .PSECT  $CODE$,  R0

000000  004137  000000G         QIO.LBN::
                                        JSR     R1,$SAVE5                       ;                                   4721
000004  024646                          CMP     -(SP),-(SP)
000006  112746  000001                  MOVB    #1,-(SP)                        ; *,WINCHESTER                      4739
000012  013701  000000G                 MOV     CST.ADDR,R1                     ;                                   4759
000016  013702  000000G                 MOV     CUOFF,R2
000022  010200                          MOV     R2,R0
000024  006300                          ASL     R0
000026  060100                          ADD     R1,R0
000030  016005  000002                  MOV     2(R0),R5                        ; *,S0.TEMP
000034  010200                          MOV     R2,R0                           ;                                   4760
000036  006300                          ASL     R0
000040  060100                          ADD     R1,R0
```

```
000042  016003  000004                        MOV     4(R0),R3                ; *,S1.TEMP
000046  010200                                MOV     R2,R0                   ;                       4761
000050  006300                                ASL     R0
000052  060100                                ADD     R1,R0
000054  016046  000006                        MOV     6(R0),-(SP)             ; *,E0.TEMP
000060  010200                                MOV     R2,R0                   ;                       4762
000062  006300                                ASL     R0
000064  060100                                ADD     R1,R0
000066  016004  000010                        MOV     10(R0),R4               ; *,E1.TEMP
000072  006302                                ASL     R2                      ;                       4767
000074  060102                                ADD     R1,R2
000076  132712  000020                        BITB    #20,(R2)
000102  001430                                BEQ     2$
000104  032737  004000  000000G               BIT     #4000,SWP.FLAGS         ;                       4768
000112  001421                                BEQ     1$
000114  013700  000114'                       MOV     MAD1,R0                 ;                       4769
000120  126027  000022  000042                CMPB    22(R0),#42
000126  001013                                BNE     1$
000130  132737  000001  000000G               BITB    #1,FORCED.ERROR         ;                       4770
000136  001407                                BEQ     1$
000140  013766  000000G 000004                MOV     FER0.LBN,4(SP)          ; *,LBN0                4773
000146  013766  000000G 000006                MOV     FER1.LBN,6(SP)          ; *,LBN1                4774
000154  000544                                BR      16$                     ;                       4772
000156  132712  000020              1$:       BITB    #20,(R2)                ;                       4778
000162  001002                                BNE     3$
000164  105066  000002              2$:       CLRB    2(SP)                   ; WINCHESTER            4780
000170  032737  000002  000000G     3$:       BIT     #2,SWP.FLAGS            ;                       4782
000176  001447                                BEQ     8$
000200  004737  011744'                       JSR     PC,RANDY                ;                       4806
000204  023704  000130'                       CMP     RNDY1,R4                ; *,E1.TEMP             4807
000210  101004                                BHI     4$
000212  001013                                BNE     5$
000214  023716  000126'                       CMP     RNDY0,(SP)              ; *,E0.TEMP             4809
000220  101410                                BLOS    5$
000222  010400                      4$:       MOV     R4,R0                   ; E1.TEMP,*             4812
000224  005100                                COM     R0
000226  040037  000130'                       BIC     R0,RNDY1
000232  011600                                MOV     (SP),R0                 ; E0.TEMP,*             4813
000234  005100                                COM     R0
000236  040037  000126'                       BIC     R0,RNDY0
000242  023703  000130'              5$:       CMP     RNDY1,R3                ; *,S1.TEMP             4816
000246  103404                                BLO     6$
000250  001013                                BNE     7$
000252  023705  000126'                       CMP     RNDY0,R5                ; *,S0.TEMP             4818
000256  103010                                BHIS    7$
000260  010300                      6$:       MOV     R3,R0                   ; S1.TEMP,*             4821
000262  005100                                COM     R0
000264  040037  000130'                       BIC     R0,RNDY1
000270  010500                                MOV     R5,R0                   ; S0.TEMP,*             4822
000272  005100                                COM     R0
000274  040037  000126'                       BIC     R0,RNDY0
000300  013766  000126'  000004      7$:       MOV     RNDY0,4(SP)             ; *,LBN0                4825
000306  013766  000130'  000006              MOV     RNDY1,6(SP)             ; *,LBN1                4826
```

E14


```
000314  000464                          BR      16$
000316  013702  000000G         8$:     MOV     L$LUN,R2                ;                               4782
000322  010201                          MOV     R2,R1                   ;                               4832
000324  006301                          ASL     R1
000326  006301                          ASL     R1
000330  012700  000000G                 MOV     #BST,R0
000334  060100                          ADD     R1,R0
000336  011066  000004                  MOV     (R0),4(SP)              ; *,LBN0
000342  062701  000002G                 ADD     #BST+2,R1               ;
000346  011166  000006                  MOV     (R1),6(SP)              ; *,LBN1                        4833
000352  062702  000000G                 ADD     #TRK.SGN,R2             ;                               4835
000356  121227  000001                  CMPB    (R2),#1
000362  001021                          BNE     12$
000364  021027  177777                  CMP     (R0),#-1                ;                               4838
000370  001003                          BNE     9$
000372  005010                          CLR     (R0)                    ;                               4841
000374  005211                          INC     (R1)                    ;                               4842
000376  000401                          BR      10$                     ;                               4838
000400  005210          9$:             INC     (R0)                    ;                               4845
000402  021104          10$:            CMP     (R1),R4                 ; *,E1.TEMP                     4848
000404  101003                          BHI     11$
000406  001027                          BNE     16$                     ;                               4849
000410  021016                          CMP     (R0),(SP)               ; *,E0.TEMP                     4850
000412  101425                          BLOS    16$
000414  011610          11$:            MOV     (SP),(R0)               ; E0.TEMP,*                     4853
000416  010411                          MOV     R4,(R1)                 ; E1.TEMP,*                     4854
000420  112712  000377                  MOVB    #377,(R2)               ;                               4855
000424  000420                          BR      16$                     ;                               4835
000426  005710          12$:            TST     (R0)                    ;                               4863
000430  001004                          BNE     13$
000432  012710  177777                  MOV     #-1,(R0)                ;                               4866
000436  005311                          DEC     (R1)                    ;                               4867
000440  000401                          BR      14$                     ;                               4863
000442  005310          13$:            DEC     (R0)                    ;                               4870
000444  021103          14$:            CMP     (R1),R3                 ; *,S1.TEMP                     4873
000446  002403                          BLT     15$
000450  001006                          BNE     16$                     ;                               4874
000452  021005                          CMP     (R0),R5                 ; *,S0.TEMP                     4875
000454  103004                          BHIS    16$
000456  010510          15$:            MOV     R5,(R0)                 ; S0.TEMP,*                     4878
000460  010311                          MOV     R3,(R1)                 ; S1.TEMP,*                     4879
000462  112712  000001                  MOVB    #1,(R2)                 ;                               4880
000466  020304          16$:            CMP     R3,R4                   ; S1.TEMP,E1.TEMP               4888
000470  001006                          BNE     17$
000472  020516                          CMP     R5,(SP)                 ; S0.TEMP,E0.TEMP
000474  001004                          BNE     17$
000476  010566  000004                  MOV     R5,4(SP)                ; S0.TEMP,LBN0                  4891
000502  010366  000006                  MOV     R3,6(SP)                ; S1.TEMP,LBN1                  4892
000506  013700  000114'        17$:     MOV     MAD1,R0                 ;                               4895
000512  016660  000004  000046          MOV     4(SP),46(R0)            ; LBN0,*
000520  016660  000006  000050          MOV     6(SP),50(R0)            ; LBN1,*                        4896
000526  005737  000112'                 TST     MX2                     ;                               4898
000532  002405                          BLT     18$
```

F14


```
000534  013700  000116'                          MOV     MAD2,R0                         ;                                  4900
000540  016660  000004  000046                   MOV     4(SP),46(R0)                    ; LBN0,*
000546  013700  000116'                  18$:    MOV     MAD2,R0                         ;                                  4901
000552  016660  000006  000050                   MOV     6(SP),50(R0)                    ; LBN1,*
000560  016637  000004  001302'                  MOV     4(SP),LBN0.SAVE                 ; LBN0,*                           4903
000566  016637  000006  001304'                  MOV     6(SP),LBN1.SAVE                 ; LBN1,*                           4904
000574  062706  000010                           ADD     #10,SP                          ;                                  4721
000600  000207                                   RTS     PC

; Routine Size:  193 words,     Routine Base:  $CODE$ + 17572
; Maximum stack depth per invocation:  11 words

;   4907  1
```

```
;   4908  1     !!ZZZ routine QIO_SIZE : novalue =
;   4909  1     GLOBAL ROUTINE QIO_SIZE : NOVALUE =
;   4910  1
;   4911  1     !+
;   4912  1     !       THIS ROUTINE IS CALLED BY QIO GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;   4913  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
;   4914  1     !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS FALL BETWEEN 1 AND THE
;   4915  1     !       I/O BUFFER SIZE (BYTS_PER_QIO). It is assumed that BYTS_PER_QIO will
;   4916  1     !       never be larger than one binary word or  65000 bytes.
;   4917  1     !
;   4918  1     !       IMPLICIT OUTPUTS:
;   4919  1     !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;   4920  1     !-
;   4921  1
;   4922  2         begin
;   4923  2
;   4924  2         local
;   4925  2             SIZE : word,                                                    ! BYTE COUNT
;   4926  2             BLOCKS_LEFT : word;                                             ! REMAINING BLOCKS LEFT
;   4927  2
;   4928  2         SIZE = ((.RANDOM [4] and %o'077777') mod (.BYTS_PER_QIO + 1)) and %o'177760';!GET BYTE COUNT FROM RANDOM NUMBER
;   4929  2
;   4930  2         if .SIZE eql 0
;   4931  2         then
;   4932  2             SIZE = 16;
;   4933  2
;   4934  2         if .CST_ADDR [.CUOFF + 4, D_END1] gtru .MAD1 [LBN_H]
;   4935  2         then BLOCKS_LEFT = %o'177777'
;   4936  2         else BLOCKS_LEFT = .CST_ADDR [.CUOFF + 3, D_END0] - .MAD1 [LBN_L] + 1;          ! find
;   4937  2                                                                                         ! REMAINING BLOCK COUNT
;   4938  2         if ((.SIZE + BYTES_PER_SECT - 1) / BYTES_PER_SECT) gtru .BLOCKS_LEFT    ! IF BLOCK COUNT NOT ENOUGH
;   4939  2         then
;   4940  2             SIZE = .BLOCKS_LEFT * BYTES_PER_SECT;                              ! ADJUST BYTE COUNT DOWN
;   4941  2
;   4942  2         MAD1 [BC_LO] = .SIZE;                                                  ! LOAD SIZE INTO 1ST MSCP PACKET
;   4943  2
;   4944  2         if .MX2 geq 0                                                          ! IF 2 QIOS
;   4945  2         then
;   4946  2             MAD2 [BC_LO] = .SIZE;                                              ! LOAD SIZE INTO 2ND MSCP PACKET
;   4947  2
;   4948  1         end;                                                                   ! ROUTINE QIO_SIZE


                                        .SBTTL  QIO.SIZE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       QIO.SIZE::
                                        JSR     R1,$SAVE3                       ;                       4909
000004  013746  000010G               MOV     RANDOM+10,-(SP)                 ;                       4928
000010  042716  100000                BIC     #100000,(SP)
000014  013746  000000G               MOV     BYTS.PER.QIO,-(SP)
000020  005216                        INC     (SP)
000022  004737  000000G               JSR     PC,BL$MOD
000026  010003                        MOV     R0,R3                           ; *,SIZE
000030  042703  000017                BIC     #17,R3                          ; *,SIZE
```

H14


```
000034  001002                          BNE     1$                      ;                               4930
000036  012703  000020                  MOV     #20,R3                  ; *,SIZE                        4932
000042  013700  000000G         1$:     MOV     CUOFF,R0                ;                               4934
000046  006300                          ASL     R0
000050  063700  000000G                 ADD     CST.ADDR,R0
000054  013701  000114'                 MOV     MAD1,R1
000060  026061  000010  000050          CMP     10(R0),50(R1)
000066  101403                          BLOS    2$
000070  012702  177777                  MOV     #-1,R2                  ; *,BLOCKS.LEFT                 4935
000074  000413                          BR      3$                      ;                               4934
000076  013700  000000G         2$:     MOV     CUOFF,R0                ;                               4936
000102  006300                          ASL     R0
000104  063700  000000G                 ADD     CST.ADDR,R0
000110  016000  000006                  MOV     6(R0),R0
000114  166100  000046                  SUB     46(R1),R0
000120  010002                          MOV     R0,R2                   ; *,BLOCKS.LEFT
000122  005202                          INC     R2                      ; BLOCKS.LEFT
000124  010316         3$:     MOV     R3,(SP)                 ; SIZE,*                        4938
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,*                 4940
000152  000300                          SWAB    R0
000154  105000                          CLRB    R0
000156  006300                          ASL     R0
000160  010003                          MOV     R0,R3                   ; *,SIZE
000162  010361  000026         4$:     MOV     R3,26(R1)               ; SIZE,*                        4942
000166  005737  000112'                 TST     MX2                     ;                               4944
000172  002404                          BLT     5$
000174  013700  000116'                 MOV     MAD2,R0                 ;                               4946
000200  010360  000026                  MOV     R3,26(R0)               ; SIZE,*
000204  022626         5$:     CMP     (SP)+,(SP)+             ;                               4922
000206  000207                          RTS     PC                      ;                               4909

; Routine Size:  68 words,    Routine Base:  $CODE$ + 20374
; Maximum stack depth per invocation:  8 words
```

```
;  4949  1     GLOBAL routine FILL_BUFF : novalue =
;  4950  1
;  4951  1     !+
;  4952  1     !       THIS ROUTINE IS CALLED BY QIO GEN TO LOAD THE I/O BUFFER DESCRIBED IN
;  4953  1     !       THE FIRST MSCP PACKET WITH THE APPROPRIATE DATA PATTERN.
;  4954  1     !
;  4955  1     !       THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
;  4956  1     !
;  4957  1     !               IF THE OPERATOR DEFINED A DATA PATTERN
;  4958  1     !               THEN
;  4959  1     !                       SELECT IT
;  4960  1     !               ELSE
;  4961  1     !                       GET DATA PATTERN NUMBER FROM SW P-TABLE
;  4962  1     !                       IF DATA PATTERN NUMBER = 0
;  4963  1     !                       THEN
;  4964  1     !                               GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
;  4965  1     !                               IN THE DATA PATTERN SEQUENCE TABLE (DPST)
;  4966  1     !
;  4967  1     !       NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
;  4968  1     !       21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
;  4969  1     !
;  4970  1     !       IMPLICIT INPUTS:
;  4971  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;  4972  1     !-
;  4973  1
;  4974  2         begin
;  4975  2
;  4976  2         local
;  4977  2             DP_NUM : word,                          ! DATA PATTERN NUMBER SELECTED
;  4978  2             DP_ADDR,                                ! ADDR OF DATA PATTERN (LENGTH)
;  4979  2             IOB_ADDR,                               ! I/O BUFFER ADDRESS (DESTINATION)
;  4980  2             SRC_ADDR,                               ! WORKING SOURCE ADDRESS
;  4981  2             COUNT : word,                           ! NO. OF WORDS IN DATA PATTERN
;  4982  2             CUR_PRIORITY : word;                    ! MMM
;  4983  2
;  4984  3         if BIT_TST (SWP_FLAGS, SWF_UDP)             ! IF USER DEFINED A DATA PATTERN
;  4985  2         then
;  4986  2             DP_ADDR = SWP_UCNT                      ! SELECT IT
;  4987  2         else
;  4988  3             begin
;  4989  3
;  4990  3             if .SWP_DPAT neq 0                      ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;  4991  3             then
;  4992  3                 DP_NUM = .SWP_DPAT                  ! SELECT IT
;  4993  3             else
;  4994  4                 begin
;  4995  4                 DP_NUM = .DPST [.L$LUN];            ! GET PATTERN NUMBER FROM SEQUENCE TABLE
;  4996  4                 DPST [.L$LUN] = .DPST [.L$LUN] + 1; ! ADVANCE TO NEXT PATTERN NUMBER
;  4997  4
;  4998  4                 if .DPST [.L$LUN] gtru DP_CNT       ! CHECK FOR HIGH LIMIT
;  4999  4                 then
;  5000  4                     DPST [.L$LUN] = 1;
;  5001  4
```

J14


```
;  5002  3                 end;
;  5003  3
;  5004  3             DP_ADDR = .DPA_TBL [.DP_NUM - 1];              ! ADDRESS OF DATA PATTERN (COUNT)
;  5005  3
;  5006  3             if .DP_NUM gequ 17
;  5007  3             then
;  5008  3
;  5009  3                 if .DP_NUM                                 ! CHECK MACRO (IF PATTERN 17, 19, OR 21)
;  5010  3                 then
;  5011  3                     (.DP_ADDR + 2) = .MAD1 [LBN_L]         ! LOAD LBN INTO FIRST WORD OF PATTERN
;  5012  3                 else
;  5013  3                     (.DP_ADDR + 4) = .MAD1 [LBN_L];        ! LOAD LBN INTO SECOND WORD OF PATTERN
;  5014  3
;  5015  2             end;
;  5016  2
;  5017  2         IOB_ADDR = .MAD1 [BUF_0];                          ! I/O BUFFER ADDRESS
;  5018  2         COUNT = ..DP_ADDR;                                 ! NO. OF WORDS IN DATA PATTERN
;  5019  2         SRC_ADDR = .DP_ADDR + 2;                           ! START OF THE ACTUAL DATA PATTERN
;  5020  2
;  5021  2         incr N from 1 to ((.MAD1 [BC_LO] + 1) / 2) do      ! FOR EACH WORD IN THIS WRITE REQUEST
;  5022  3             begin
;  5023  3             .IOB_ADDR = ..SRC_ADDR;                        ! MOVE 1 WORD
;  5024  3             IOB_ADDR = .IOB_ADDR + 2;                      ! ADVANCE DESTINATION ADDRESS
;  5025  3             SRC_ADDR = .SRC_ADDR + 2;                      ! ADVANCE SOURCE ADDRESS
;  5026  3             COUNT = .COUNT - 1;                            ! DECREMENT COUNT
;  5027  3
;  5028  3             if .COUNT eql 0                                ! IF END OF DATA PATTERN
;  5029  3             then
;  5030  4                 begin
;  5031  4                 COUNT = ..DP_ADDR;                         ! REPEAT DATA PATTERN
;  5032  4                 SRC_ADDR = .DP_ADDR + 2;
;  5033  3                 end;
;  5034  3
;  5035  2             end;                                           ! WORD TRANSFER LOOP
;  5036  2
;  5037  2     !       if (.CSR_MEM and .TST_PAR)                                          ! MMM
;  5038  2     !       then                                                                ! MMM
;  5039  2     !           if not .PAR_TSD                                                 ! MMM
;  5040  2     !           then                                                            ! MMM
;  5041  2     !               begin                                                       ! MMM
;  5042  2     !               GETPRI(CUR_PRIORITY);                                       ! MMM
;  5043  2     !               SETPRI(PRI07);                                              ! MMM
;  5044  2     !               IOB_ADDR = .MAD1 [BUF_0];                                   ! I/O BUFFER ADDRESS MMM
;  5045  2     !               incr COUNT from 1 to 2 do                                   ! MMM
;  5046  2     !                   begin                                                   ! MMM
;  5047  2     !                   %o'172100' = %o'5';                                     ! MMM
;  5048  2     !                   end;                                                    ! MMM
;  5049  2     !               .IOB_ADDR = %o'2';                                          ! MOVE 1 WORD MMM
;  5050  2     !               %o'172100' = %o'1';                                         ! MMM
;  5051  2     !               SETPRI(.CUR_PRIORITY);                                      ! MMM
;  5052  2     !               end;                                                        ! MMM
;  5053  2
;  5054  1         end;                                               ! ROUTINE FILL_BUFF
```

K14


```
                                        .SBTTL  FILL.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                FILL.BUFF::
                                        JSR     R1,$SAVE5                       ;                               4949
000004  005746                          TST     -(SP)
000006  032737  000100  000000G         BIT     #100,SWP.FLAGS                  ;                               4984
000014  001403                          BEQ     1$
000016  012701  000000G                 MOV     #SWP.UCNT,R1                    ; *,DP.ADDR                     4986
000022  000443                          BR      5$                              ;                               4984
000024  013700  000000G         1$:     MOV     SWP.DPAT,R0                     ;                               4990
000030  001402                          BEQ     2$
000032  010002                          MOV     R0,R2                           ; *,DP.NUM                      4992
000034  000414                          BR      3$                              ;                               4990
000036  013700  000000G         2$:     MOV     L$LUN,R0                        ;                               4995
000042  062700  000050'                 ADD     #DPST,R0
000046  005002                          CLR     R2                              ; DP.NUM
000050  151002                          BISB    (R0),R2                         ; *,DP.NUM
000052  105210                          INCB    (R0)                            ;                               4996
000054  121027  000025                  CMPB    (R0),#25                        ;                               4998
000060  101402                          BLOS    3$
000062  112710  000001                  MOVB    #1,(R0)                         ;                               5000
000066  010200                  3$:     MOV     R2,R0                           ; DP.NUM,*                      5004
000070  006300                          ASL     R0
000072  016001  001166'                 MOV     DPA.TBL-2(R0),R1                ; *,DP.ADDR
000076  020227  000021                  CMP     R2,#21                          ; DP.NUM,*                      5006
000102  103413                          BLO     5$
000104  013700  000114'                 MOV     MAD1,R0                         ;                               5011
000110  006002                          ROR     R2                              ; DP.NUM                        5009
000112  103004                          BCC     4$
000114  016061  000046  000002          MOV     46(R0),2(R1)                    ; *,*(DP.ADDR)                  5011
000122  000403                          BR      5$                              ;                               5009
000124  016061  000046  000004  4$:     MOV     46(R0),4(R1)                    ; *,*(DP.ADDR)                  5013
000132  013700  000114'         5$:     MOV     MAD1,R0                         ;                               5017
000136  016004  000032                  MOV     32(R0),R4                       ; *,IOB.ADDR
000142  011103                          MOV     (R1),R3                         ; DP.ADDR,COUNT                 5018
000144  012705  000002                  MOV     #2,R5                           ;                               5019
000150  060105                          ADD     R1,R5                           ; DP.ADDR,*
000152  010502                          MOV     R5,R2                           ; *,SRC.ADDR
000154  016046  000026                  MOV     26(R0),-(SP)                    ;                               5021
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$:     MOV     (R2)+,(R4)+                     ; SRC.ADDR,IOB.ADDR             5023
000204  005303                          DEC     R3                              ; COUNT                         5026
000206  001002                          BNE     7$                              ;                               5028
000210  011103                          MOV     (R1),R3                         ; DP.ADDR,COUNT                 5031
000212  010502                          MOV     R5,R2                           ; *,SRC.ADDR                    5032
000214  005200                  7$:     INC     R0                              ; N                             5021
000216  020066  000004                  CMP     R0,4(SP)                        ; N,*
```

L14


```
000222  003767                          BLE     6$
000224  062706  000006                  ADD     #6,SP
000230  000207                          RTS     PC                              ;                                       4949

; Routine Size:  77 words,      Routine Base:  $CODE$ + 20604
; Maximum stack depth per invocation:  10 words


;   5055  1
;   5056  1
```

M14


```
;    5057  1     GLOBAL ROUTINE PROC_RETPKT : NOVALUE =
;    5058  1
;    5059  1     !+
;    5060  1     !       THIS ROUTINE IS CALLED FROM THE MULTI DRIVE "EXECUTIVE" AND DUP COMMAND TO CHECK FOR
;    5061  1     !       AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;    5062  1     !       PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;    5063  1     !       BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;    5064  1     !       REQUIRE PROCESSING.
;    5065  1     !
;    5066  1     !       UNDER THE MULTI-DRIVE TEST, RETURN PACKETS ORIGINATE FROM THREE SOURCES:
;    5067  1     !               1. MSCP         - THE MORE COMMON, DESCRIBING A COMPLETED I/O
;    5068  1     !                                 OPERATION.
;    5069  1     !               2. DUP          - THE LESS COMMON, DESCRIBING A PORTION OF I/O
;    5070  1     !                                 COMMUNICATIONS WITH THE CONTROLLER PROGRAM.
;    5071  1     !               3. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;    5072  1     !                                 COMMAND TIMEOUT.
;    5073  1     !-
;    5074  1
;    5075  1       while .IODQ_IN neq .IODQ_OUT do                   ! DO UNTIL I/O DONE QUEUE IS EMPTY
;    5076  2           begin
;    5077  2           RP_INDX = OUT_IODQ ();                        ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;    5078  2           RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);   ! CALCULATE RETPKT ADDRESS
;    5079  3           if NOT (.RP_ADDR [CONID] eql CID_DUP)         ! if not DUP then
;    5080  2           then (SET_CPAR (.RP_ADDR [CTLR]););           ! SET UP CURRENT CONTROLLER PARAMETERS
;    5081  2
;    5082  2           selectoneu .RP_ADDR [CONID] of                ! CONNECTION ID INDICATES PACKET SOURCE
;    5083  2               set
;    5084  2
;    5085  2               [CID_MSCP] :        IO_RETPKT ();         ! DISK MSCP (I/O TRANSFER DONE)
;    5086  2               [CID_DUP] :         DIO_RETPKT ();        ! DUP (I/O TRANSFER DONE)
;    5087  2               [CID_DRIVER] :      DR_RETPKT ();         ! MESSAGE FROM "DRIVER"
;    5088  2
;    5089  2               [otherwise] :       PRINTF (DBM12, .RP_ADDR [CONID]);!"CONN ID = XXXXX RECEIVED"
;    5090  2               tes;
;    5091  2
;    5092  1           end;                                          ! UNITL I/O DONE QUEUE IS EMPTY


                                .SBTTL  PROC.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                PROC.RETPKT::
                                MOV     R1,-(SP)
000002  023737  000000G 000000G 1$:     CMP     IODQ.IN,IODQ.OUT                ;                       5057
000010  001467                        BEQ     7$                              ;                       5075
000012  004737  000000G                JSR     PC,OUT.IODQ                     ;                       5077
000016  010037  000000G                MOV     R0,RP.INDX
000022  010046                        MOV     R0,-(SP)                        ; RP.INDX,*             5078
000024  012746  000054                MOV     #54,-(SP)
000030  004737  000000G                JSR     PC,BL$MUL
000034  062700  000000G                ADD     #RETPKT,R0
000040  010037  000000G                MOV     R0,RP.ADDR
000044  126027  000003  000002        CMPB    3(R0),#2                        ;                       5079
000052  001406                        BEQ     2$
000054  116016  000002                MOVB    2(R0),(SP)                      ;                       5080
```

N14


```
000060  042716  177760                    BIC     #177760,(SP)
000064  004737  000000G                   JSR     PC,SET.CPAR
000070  013700  000000G           2$:     MOV     RP.ADDR,R0              ;                       5082
000074  005001                            CLR     R1
000076  156001  000003                    BISB    3(R0),R1
000102  005701                            TST     R1                      ;                       5085
000104  001003                            BNE     3$
000106  004737  000000V                   JSR     PC,IO.RETPKT
000112  000424                            BR      6$
000114  020127  000002            3$:     CMP     R1,#2                   ;                       5082
000120  001003                            BNE     4$                      ;                       5086
000122  004737  000000V                   JSR     PC,DIO.RETPKT
000126  000416                            BR      6$
000130  020127  000003            4$:     CMP     R1,#3                   ;                       5082
000134  001003                            BNE     5$                      ;                       5087
000136  004737  000000V                   JSR     PC,DR.RETPKT
000142  000410                            BR      6$
000144  010116                    5$:     MOV     R1,(SP)                 ;                       5082
000146  012746  000000G                   MOV     #DBM12,-(SP)            ;                       5089
000152  012746  000002                    MOV     #2,-(SP)
000156  010600                            MOV     SP,R0                   ; SP,*
000160  104417                            TRAP    17
000162  022626                            CMP     (SP)+,(SP)+
000164  022626                    6$:     CMP     (SP)+,(SP)+             ;                       5076
000166  000705                            BR      1$                      ;                       5075
000170  012601                    7$:     MOV     (SP)+,R1                ;                       5057
000172  000207                            RTS     PC

; Routine Size:  62 words,    Routine Base:  $CODE$ + 21036
; Maximum stack depth per invocation:  7 words
```

```
;    5093  1
;    5094  1     GLOBAL ROUTINE DIO_RETPKT : NOVALUE =
;    5095  1
;    5096  1     !+
;    5097  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
;    5098  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;    5099  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS.
;    5100  1     !
;    5101  1     !       IMPLICIT INPUTS:
;    5102  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5103  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5104  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    5105  1     !               DUOFF - CST OFFSET FOR THE CURRENT UNIT
;    5106  1     !               L$LUN - CURRENT UNIT NUMBER
;    5107  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5108  1     !
;    5109  1     !       IMPLICIT OUTPUTS
;    5110  1     !               CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] - IF THIS BIT SET NO DUP EXERCISER
;    5111  1     !
;    5112  1     !-
;    5113  2     BEGIN
;    5114  2
;    5115  2     LOCAL   FLAG : BYTE INITIAL(BYTE(TRUE)),
;    5116  2             SUM2 : WORD,
;    5117  2             SUM : WORD;                                    ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;    5118  2     !PRINTX (DER18);
;    5119  2
;    5120  2     IF .RP_ADDR [STATUS] NEQU ST_SUC                       ! IF STATUS CODE INDICATES ERROR
;    5121  2     THEN
;    5122  3             BEGIN
;    5123  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET DUP ERROR FLAG
;    5124  3             HARD_ERROR ();
;    5125  3             IF .RP_ADDR [ENDCOD] EQLU (OP_ELP + OP_END) OR  ! IF ENDCODE IS EXECUTE LOCAL PROGRAM
;    5126  4                .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)     !                 OR GET DUST STATUS
;    5127  4             THEN    BEGIN
;    5128  4                     CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;        ! TURN OFF DUP EXERCISR
;    5129  3                     END;
;    5130  3             END
;    5131  2     ELSE                                                   ! ELSE - I/O WAS SUCCESSFUL
;    5132  3         BEGIN
;    5133  3
;    5134  4         IF .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) ! IF ENDCODE IS GET DUST STATUS
;    5135  3         THEN
;    5136  4             BEGIN
;    5137  4             IF .RP_ADDR [9,11,1,0] EQL 1
;    5138  4                     THEN CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = ACTIVE         ! CONTROLLER IN AN ACTIVE STAE
;    5139  4                     ELSE CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = IDLE;          ! CONTROLLER IN AN IDLE STATE
;    5140  4             IF .RP_ADDR [9,9,1,0] NEQ 1 THEN       ! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
;    5141  5                     BEGIN
;    5142  5                     HARD_ERROR ();
;    5143  5                     CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;        ! TURN OFF DUP EXERCISR
;    5144  4                     END;
;    5145  3             END;
```

```
;   5146  3
;   5147  3
;   5148  3     IF (.RP_ADDR [ENDCOD] EQL (OP_RCD + OP_END)) AND
;   5149  3        (.DUPPKT [DUPTYPE] EQL 6) AND
;   5150  3        (.DUPPKT [DUPMESG] EQL 2) AND                    !IF IT IS A RECEIVE DBN COMMAND WITH TYPE 6 AND MESSAGE 2 THEN
;   5151  4        (.CST_ADDR [.DUOFF + OF_DBN, DUPWRITE] EQLU 1)       ! IF WRITE FLAG SET IN CST TABLE THEN COMPARE BLOCKS
;   5152  3             THEN DUP_COMPARE ();
;   5153  3
;   5154  2     END;                                                 ! COMPARE THE FOLLOWING 512 BYTES
;   5155  2
;   5156  2     PUT_RETPKT (.RP_INDX);
;   5157  1     END;                                                 ! ROUTINE DIO_RETPKT


                                        .SBTTL  DIO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                    DIO.RETPKT::
                                        MOV     R1,-(SP)                        ;
000002  112700  000001                  MOVB    #1,R0                           ; *,FLAG                        5094
000006  013701  000000G                 MOV     RP.ADDR,R1                      ;                               5113
000012  005761  000016                  TST     16(R1)                                                          5120
000016  001435                          BEQ     2$
000020  013700  001250'                 MOV     DUOFF,R0                        ;                               5123
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
000044  013700  000000G                 MOV     RP.ADDR,R0                      ;                               5124
000050  126027  000014  000203          CMPB    14(R0),#203                     ;                               5125
000056  001404                          BEQ     1$
000060  126027  000014  000201          CMPB    14(R0),#201                     ;                               5126
000066  001100                          BNE     6$
000070  013700  001250'          1$:     MOV     DUOFF,R0                        ;                               5128
000074  006300                          ASL     R0
000076  063700  000000G                 ADD     CST.ADDR,R0
000102  052760  100000  000020          BIS     #100000,20(R0)
000110  000467                          BR      6$
000112  126127  000014  000201  2$:     CMPB    14(R1),#201                     ;                               5120
000120  001036                          BNE     5$                              ;                               5134
000122  013700  001250'                 MOV     DUOFF,R0                        ;                               5138
000126  006300                          ASL     R0
000130  063700  000000G                 ADD     CST.ADDR,R0
000134  032761  004000  000022          BIT     #4000,22(R1)                    ;                               5137
000142  001404                          BEQ     3$
000144  052760  020000  000020          BIS     #20000,20(R0)                   ;                               5138
000152  000403                          BR      4$                              ;                               5137
000154  042760  020000  000020  3$:     BIC     #20000,20(R0)                   ;                               5139
000162  032761  001000  000022  4$:     BIT     #1000,22(R1)                    ;                               5140
000170  001012                          BNE     5$
000172  004737  000000V                 JSR     PC,HARD.ERROR                   ;                               5142
000176  013700  001250'                 MOV     DUOFF,R0                        ;                               5143
000202  006300                          ASL     R0
000204  063700  000000G                 ADD     CST.ADDR,R0
000210  052760  100000  000020          BIS     #100000,20(R0)
```

```
000216  013700  000000G                 5$:     MOV     RP.ADDR,R0                      ;                                       5148
000222  126027  000014  000205                  CMPB    14(R0),#205
000230  001017                                  BNE     6$
000232  023727  000000G 060002                  CMP     DUPPKT,#60002                   ;                                       5149
000240  001013                                  BNE     6$
000242  013700  001250'                         MOV     DUOFF,R0                        ;                                       5151
000246  006300                                  ASL     R0
000250  063700  000000G                         ADD     CST.ADDR,R0
000254  032760  010000  000020                  BIT     #10000,20(R0)
000262  001402                                  BEQ     6$
000264  004737  000000V                         JSR     PC,DUP.COMPARE                  ;                                       5152
000270  013746  000000G                 6$:     MOV     RP.INDX,-(SP)                   ;                                       5156
000274  004737  000000G                         JSR     PC,PUT.RETPKT
000300  005726                                  TST     (SP)+                           ;                                       5113
000302  012601                                  MOV     (SP)+,R1                        ;                                       5094
000304  000207                                  RTS     PC

; Routine Size:  99 words,      Routine Base:  $CODE$ + 21232
; Maximum stack depth per invocation:  3 words

;   5158  1
```

```
;    5159  1     GLOBAL ROUTINE DUP_COMPARE : NOVALUE =
;    5160  1
;    5161  1
;    5162  1     !+
;    5163  1     !       THIS ROUTINE IS CALLED BY DIO_RETPKT WHEN THE RECEIVE DATA COMMAND
;    5164  1     !       IS BEING PROCESSED. THIS COMMAND COMPARES THE WRITTEN BUFFER WITH
;    5165  1     !       THE PATERN WORD GIVEN IN SEND DATA COMMAND.  FOR EVERY WORD COMPARED
;    5166  1     !       THE ROUTINE INCREMENTS THE TALLY TABLE.  IF THE COMPARE SHOWS AN
;    5167  1     !       ERROR. THE DBN HARD ERROR COUNTER WILL BE INCREMENTED AND THE
;    5168  1     !       THE DBN NUMBER AND BYTE COUNT WILL BE PRINTED.
;    5169  1     !
;    5170  1     !       IMPLICIT INPUTS:
;    5171  1     !               S_PATTERN       ! THE SAVED PATTERN WRITTEN TO THE DBN'S
;    5172  1     !               S_DUPPKT        ! THE POINTER FOR DUP BUFFER
;    5173  1     !               T_ADDR          ! THE ADDRESS OF THE TALLY TABLE FOR THIS UNIT
;    5174  1     !               CST_ADDR        ! THE ADDRESS OF PRESENT CONTROLLER STATUS TABLE
;    5175  1     !-
;    5176  2     BEGIN
;    5177  2
;    5178  2     OWN
;    5179  2             COUNT : WORD;
;    5180  2
;    5181  2     !PRINTX (DER19);
;    5182  2     S_DUPPKT = 0;
;    5183  2     INCR COUNT FROM 1 TO 256 DO     !INDEX PIONTER FOR DATA STORED IN MSCP ENV PACKET
;    5184  3             BEGIN
;    5185  3             S_DUPPKT = .S_DUPPKT + 2;                       ! INITIALLY THIS SKIPS THE FIRST WORD OF DUPPKT
;    5186  3             IF .(DUPPKT + .S_DUPPKT) NEQ .S_PATTERN THEN    !IF THE CONTENTS OF DBN DOESN'T EQUAL PATTERN
;    5187  4                     BEGIN
;    5188  4                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;       ! SET DUP ERROR FLAG
;    5189  4                     ERRHRD (46, EH_10, EMS_22);             !LIST ERROR
;    5190  4                     EXITLOOP;
;    5191  3                     END;
;    5192  2             END;                                    !GO THROUGH ALL DBN WORDS
;    5193  1     END;                                            !END ROUTINE DUP-COMPARE


001306                                  .PSECT  $GGG$,  RO
001306                          COUNT:  .BLKW   1


                                        .SBTTL  DUP.COMPARE MULTI-DRIVE TEST ROUTINES
021540                                  .PSECT  $CODE$,  RO

000000  010146                  DUP.COMPARE::
                                        MOV     R1,-(SP)                ;                               5159
000002  005037  000000G                 CLR     S.DUPPKT                ;                               5182
000006  012701  000400                  MOV     #400,R1                 ; *,COUNT                       5183
000012  062737  000002  000000G 1$:     ADD     #2,S.DUPPKT             ;                               5185
000020  013700  000000G                 MOV     S.DUPPKT,R0             ;                               5186
000024  026037  000000G 000000G         CMP     DUPPKT(R0),S.PATTERN
000032  001415                          BEQ     2$
```

F15


```
000034  013700  001250'                        MOV     DUOFF,R0                        ;                               5188
000040  006300                                 ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  052760  040000  000020                 BIS     #40000,20(R0)
000054  104456                                 TRAP    56                              ;                               5189
000056  000056                                 .WORD   56
000060  000000G                                .WORD   EH.10
000062  000000G                                .WORD   EMS.22
000064  000402                                 BR      3$                              ;
000066  005301                         2$:     DEC     R1                              ; COUNT                         5187
000070  001350                                 BNE     1$                                                              5183
000072  012601                         3$:     MOV     (SP)+,R1                        ;                               5159
000074  000207                                 RTS     PC

; Routine Size:  31 words,      Routine Base:  $CODE$ + 21540
; Maximum stack depth per invocation:  3 words


;   5194  1
;   5195  1
;   5196  1
```

```
;  5197  1     GLOBAL routine IO_RETPKT : novalue =
;  5198  1
;  5199  1     !+
;  5200  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;  5201  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;  5202  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;  5203  1     !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;  5204  1     !
;  5205  1     !       IMPLICIT INPUTS:
;  5206  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  5207  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  5208  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  5209  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  5210  1     !               L$LUN - CURRENT UNIT NUMBER
;  5211  1     !-
;  5212  1
;  5213  2         begin
;  5214  2
;  5215  2         local
;  5216  2             FLAG : byte initial (byte (TRUE));
;  5217  2
;  5218  2         FSET_UPAR ();                               ! FIND UNIT'S ENTRY IS CST AND SET UP UNIT-RELATED DATA
;  5219  2         ST_CODE = .RP_ADDR [STSCOD];                ! GET STATUS CODE FROM RETPKT
;  5220  2         SB_CODE = .RP_ADDR [SUBCOD];                ! GET SUB-CODE, IF ANY
;  5221  2
;  5222  3         if (.ST_CODE neq ST_SUC)                    ! IF STATUS CODE INDICATES ERROR
;  5223  2         then
;  5224  3             begin
;  5225  3             HARD_ERROR ();                          ! UPDATE ERROR COUNT
;  5226  3             COMPARE_DATA = FALSE;                   ! NO POINT IN DOING HOST COMPARES ON ERRORS
;  5227  3
;  5228  3             if (.ST_CODE neq ST_OFL) and            ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
;  5229  3                (.ST_CODE neq ST_AVL) and
;  5230  4                (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
;  5231  3             then
;  5232  4                 begin
;  5233  4                 DUR [.L$LUN] = DU_HERR;             ! LOAD REASON FOR DROPPING UNIT
;  5234  4                 DODU (.L$LUN);                      ! DROP UNIT
;  5235  3                 end;
;  5236  3
;  5237  3             end
;  5238  2         else                                        ! IF I/O WAS SUCCESSFUL
;  5239  3             begin
;  5240  3             UPD_IO_TALLY ();                        ! UPDATE I/O TALLY (STATISTICS)
;  5241  3
;  5242  4             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
;  5243  3             then
;  5244  3                 COMPARE_DATA = TRUE;                ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS
;  5245  3
;  5246  3             if (BIT_TST (SWP_FLAGS, SWF_HWC)) and   ! IF HOST IS DOING WRITE-COMPARES
;  5247  4                (.COMPARE_DATA)
;  5248  3             then
;  5249  3                 FLAG = HOST_WRT_CHK ();             ! SAVE I/O PACKET OR DO WRITE-CHECK
```

H15


```
;  5250  3
;  5251  2              end;
;  5252  2
;  5253  2              if .FLAG                                ! IF FLAG IS STILL TRUE
;  5254  2              then
;  5255  2                  SWEEP ();                           ! DEALLOCATE BUFFER(S) AND RETPKT(S)
;  5256  2
;  5257  2              QIO [.CCTLR] = .QIO [.CCTLR] - 1;       ! DECREMENT NO. OF OUTSTANDING QIOs
;  5258  1              end;                                    ! ROUTINE IO_RETPKT


                                        .SBTTL  IO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          IO.RETPKT::
                                        JSR     R1,$SAVE2                       ;                       5197
000004  112701  000001                  MOVB    #1,R1                           ; *,FLAG                5213
000010  004737  000000V                 JSR     PC,FSET.UPAR                    ;                       5218
000014  013700  000000G                 MOV     RP.ADDR,R0                      ;                       5219
000020  116037  000016  000000G         MOVB    16(R0),ST.CODE
000026  042737  177740  000000G         BIC     #177740,ST.CODE
000034  016002  000016                  MOV     16(R0),R2                       ;                       5220
000040  006202                          ASR     R2
000042  006202                          ASR     R2
000044  006202                          ASR     R2
000046  006202                          ASR     R2
000050  006202                          ASR     R2
000052  042702  174000                  BIC     #174000,R2
000056  010237  000000G                 MOV     R2,SB.CODE
000062  005737  000000G                 TST     ST.CODE                         ;                       5222
000066  001431                          BEQ     1$
000070  004737  000000V                 JSR     PC,HARD.ERROR                   ;                       5225
000074  105037  001264'                 CLRB    COMPARE.DATA                    ;                       5226
000100  023727  000000G  000003         CMP     ST.CODE,#3                      ;                       5228
000106  001447                          BEQ     3$
000110  023727  000000G  000004         CMP     ST.CODE,#4                      ;                       5229
000116  001443                          BEQ     3$
000120  013700  000000G                 MOV     T.ADDR,R0                       ;                       5230
000124  026037  000014  000000G         CMP     14(R0),SWP.ERROR
000132  103435                          BLO     3$
000134  013700  000000G                 MOV     L$LUN,R0                        ;                       5233
000140  112760  000004  000000G         MOVB    #4,DUR(R0)
000146  104451                          TRAP    51                              ;                       5234
000150  000426                          BR      3$                              ;                       5222
000152  004737  000000V          1$:     JSR     PC,UPD.IO.TALLY                 ;                       5240
000156  013700  000000G                 MOV     RP.ADDR,R0                      ;                       5242
000162  126027  000014  000242         CMPB    14(R0),#242
000170  001003                          BNE     2$
000172  112737  000001  001264'         MOVB    #1,COMPARE.DATA                 ;                       5244
000200  032737  000040  000000G  2$:     BIT     #40,SWP.FLAGS                   ;                       5246
000206  001407                          BEQ     3$
000210  032737  000001  001264'         BIT     #1,COMPARE.DATA                 ;                       5247
000216  001403                          BEQ     3$
000220  004737  000000V                 JSR     PC,HOST.WRT.CHK                 ;                       5249
000224  110001                          MOVB    R0,R1                           ; *,FLAG
```

I15


```
000226  006001                          3$:     ROR     R1                      ; FLAG                          5253
000230  103002                                  BCC     4$
000232  004737  000000V                         JSR     PC,SWEEP                ;                               5255
000236  013700  000000G                 4$:     MOV     CCTLR,R0                ;                               5257
000242  105360  000000G                         DECB    QIO(R0)
000246  000207                                  RTS     PC                      ;                               5197

; Routine Size:  84 words,      Routine Base:  $CODE$ + 21636
; Maximum stack depth per invocation:  5 words
```

```
;    5259  1     GLOBAL routine FSET_UPAR : novalue =
;    5260  1
;    5261  1     !+
;    5262  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;    5263  1     !       CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;    5264  1     !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;    5265  1     !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;    5266  1     !       DATA PARAMETERS.
;    5267  1     !
;    5268  1     !       IMPLICIT INPUTS:
;    5269  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5270  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    5271  1     !-
;    5272  1
;    5273  2         begin
;    5274  2
;    5275  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    5276  2
;    5277  2             if .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]        ! IF RETPKT UNIT MATCHES CST ENTRY
;    5278  2             then
;    5279  3                 begin
;    5280  3                 SET_UPAR (.OFFSET);                                                ! SET UP UNIT-RELATED DATA
;    5281  3                 return;                                                            ! DONE
;    5282  2                 end;
;    5283  2
;    5284  2         PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                   ! "CAN'T FIND DISK XXX IN CST X"
;    5285  1         end;                                                                       ! ROUTINE FSET_UPAR

                                .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G   FSET.UPAR::
                            JSR     R1,$SAVE4                       ;                       5259
000004  012702  000003      MOV     #3,R2                           ; *,OFFSET              5275
000010  010201          1$: MOV     R2,R1                           ; OFFSET,*              5277
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,*              5280
000044  004737  000000G     JSR     PC,SET.UPAR
000050  005726              TST     (SP)+                           ;                       5281
000052  000207              RTS     PC                              ;                       5279
000054  062702  000012  2$: ADD     #12,R2                          ; *,OFFSET              5275
000060  020227  000041      CMP     R2,#41                          ; OFFSET,*
000064  003751              BLE     1$
000066  013746  000000G     MOV     CCTLR,-(SP)                     ;                       5284
000072  013700  000000G     MOV     RP.ADDR,R0
000076  016046  000010      MOV     10(R0),-(SP)
000102  012746  000000G     MOV     #DBM19,-(SP)
```

K15


```
000106  012746  000003                        MOV     #3,-(SP)
000112  010600                                MOV     SP,R0
000114  104417                                TRAP    17                      ; SP,*
000116  062706  000010                        ADD     #10,SP                  ;                       5273
000122  000207                                RTS     PC                      ;                       5259

; Routine Size:  42 words,      Routine Base:  $CODE$ + 22106
; Maximum stack depth per invocation:  11 words
```

L15


```
;  5286  1     GLOBAL routine HARD_ERROR : novalue =
;  5287  1
;  5288  1     !+
;  5289  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO INCREMENT THE HARD
;  5290  1     !       ERROR STATISTIC FIELD FOR THE CURRENT UNIT.  IF THE HARD ERROR COUNT
;  5291  1     !       HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS DROPPED
;  5292  1     !       FROM TESTING.
;  5293  1     !
;  5294  1     !       IMPLICIT INPUTS:
;  5295  1     !               L$LUN - CURRENT UNIT NUMBER
;  5296  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  5297  1     !               CUOFF - CST OFFSET FOR CURRENT UNIT
;  5298  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  5299  1     !-
;  5300  1
;  5301  2         begin
;  5302  2         T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;                    ! INCREMENT UNIT'S HARD ERROR COUNT
;  5303  2     if .RP_ADDR [CONID] EQL CID_MSCP                                   !FOR MSCP ERRORS   ZZZ
;  5304  2     THEN                                                               !                  ZZZ
;  5305  2
;  5306  2         selectoneu .ST_CODE of
;  5307  2             set
;  5308  2
;  5309  2             [ST_SUC]:       if .SB_CODE neq 0                          ! SUCCESS WITH NON-ZERO SUB-CODE
;  5310  2                             then
;  5311  3                                 begin
;  5312  3
;  5313  3                                 if .SB_CODE eql 4
;  5314  3                                 then
;  5315  3                                     T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1
;  5316  3                                 else
;  5317  3                                     T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5318  3
;  5319  3                                 if .APT_MODE
;  5320  3                                 then
;  5321  3                                     ERR_HRD_RTNE_APT (44)
;  5322  3                                 else
;  5323  3                                     ERR_HRD_RTNE (44);
;  5324  3
;  5325  3                                 end;
;  5326  2
;  5327  3             [ST_CMD]:       begin                                      ! INVALID COMMAND
;  5328  3                             T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5329  3
;  5330  3                             if .APT_MODE
;  5331  3                             then
;  5332  3                                 ERR_HRD_RTNE_APT (31)
;  5333  3                             else
;  5334  3                                 ERR_HRD_RTNE (31);
;  5335  3
;  5336  3                             end;
;  5337  2
;  5338  3             [ST_ABO]:       begin                                      ! COMMAND ABORTED
```

```
;  5339  3                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5340  3
;  5341  3                    if .APT_MODE
;  5342  3                    then
;  5343  3                        ERR_HRD_RTNE_APT (32)
;  5344  3                    else
;  5345  3                        ERR_HRD_RTNE (32);
;  5346  3
;  5347  2                    end;
;  5348  2
;  5349  3            [ST_OFL] :      begin                                           ! OFFLINE
;  5350  3                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5351  3
;  5352  3                    if .APT_MODE
;  5353  3                    then
;  5354  4                        begin
;  5355  4                        .MAIL_BOX_TESTNUM = 1;
;  5356  4                        .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  5357  3                        end;
;  5358  3
;  5359  3                    ERRDF (18, EGD_18, EMS_18);                             !
;  5360  3                    DUR [.L$LUN] = DU_DFATAL;                               ! DEVICE FATAL ERROR
;  5361  3                    DODU (.L$LUN);                                          ! DROP UNIT
;  5362  2                    end;
;  5363  2
;  5364  3            [ST_AVL]:       begin                                           ! WENT TO AVAILABLE STATE
;  5365  3                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5366  3
;  5367  3                    if .APT_MODE
;  5368  3                    then
;  5369  4                        begin
;  5370  4                        .MAIL_BOX_TESTNUM = 1;
;  5371  4                        .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  5372  3                        end;
;  5373  3
;  5374  3                    ERRDF (24, EGD_18, EMS_24);                             !
;  5375  3                    DUR [.L$LUN] = DU_DFATAL;                               ! DEVICE FATAL ERROR
;  5376  3                    DODU (.L$LUN);                                          ! DROP UNIT
;  5377  2                    end;
;  5378  2
;  5379  3            [ST_MFE]:       begin                                           ! MEDIA FORMAT ERROR
;  5380  3                    T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1;
;  5381  3
;  5382  3                    if .APT_MODE
;  5383  3                    then
;  5384  3                        ERR_HRD_RTNE_APT (35)
;  5385  3                    else
;  5386  3                        ERR_HRD_RTNE (35);
;  5387  3
;  5388  2                    end;
;  5389  2
;  5390  3            [ST_WPT]:       begin                                           ! DEVICE WRITE PROTECTED
;  5391  3
```

```
;  5392  3                        if .SB_CODE eql 128
;  5393  3                        then
;  5394  3                            T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1
;  5395  3                        else
;  5396  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5397  3
;  5398  3                        if .APT_MODE
;  5399  3                        then
;  5400  3                            ERR_HRD_RTNE_APT (36)
;  5401  3                        else
;  5402  3                            ERR_HRD_RTNE (36);
;  5403  3
;  5404  3                        end;
;  5405  2
;  5406  2                [ST_CMP]:       begin                                   ! COMPARE ERROR
;  5407  3                        T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  5408  3
;  5409  3                        if .APT_MODE
;  5410  3                        then
;  5411  3                            ERR_HRD_RTNE_APT (37)
;  5412  3                        else
;  5413  3                            ERR_HRD_RTNE (37);
;  5414  3
;  5415  3                        end;
;  5416  2
;  5417  2                [ST_DAT]:       begin                                   ! DATA ERROR
;  5418  3
;  5419  3                        if .SB_CODE eql 2
;  5420  3                        then
;  5421  3                            T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  5422  3                        else
;  5423  3                            T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  5424  3
;  5425  3                        if (.SB_CODE eql 0) and
;  5426  3                           (not .FORCED_ERROR) and
;  5427  4                           (BIT_TST (SWP_FLAGS, SWF_FER))
;  5428  3                        then
;  5429  4                            begin
;  5430  4                            FORCED_ERROR = TRUE;                        ! BLOCK WITH "FORCED ERROR" FOUND
;ZZZ5431  4                            FER0_LBN = .RP_ADDR [LBN_LO];               !
;ZZZ5432  4                            FER1_LBN = .RP_ADDR [LBN_HI];               !
;  5433  4                            FER_BC = .RP_ADDR [CBCNT_LO];               !
;  5434  3                            end;
;  5435  3
;  5436  3
;  5437  3                        if .APT_MODE
;  5438  3                        then
;  5439  3                            ERR_HRD_RTNE_APT (38)
;  5440  3                        else
;  5441  3                            ERR_HRD_RTNE (38);
;  5442  3
;  5443  2                        end;
;  5444  2
```

```
;  5445  3          [ST_HST]:       begin                                           ! HOST ACCESS ERROR
;  5446  3                          T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  5447  3
;  5448  3                          if .APT_MODE
;  5449  3                          then
;  5450  3                              ERR_HRD_RTNE_APT (39)
;  5451  3                          else
;  5452  3                              ERR_HRD_RTNE (39);
;  5453  3
;  5454  3                          end;
;  5455  3
;  5456  3          [ST_CNT]:       begin                                           ! CONTROLLER ERROR
;  5457  3                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5458  3
;  5459  3                          if .APT_MODE
;  5460  3                          then
;  5461  3                              ERR_HRD_RTNE_APT (40)
;  5462  3                          else
;  5463  3                              ERR_HRD_RTNE (40);
;  5464  3
;  5465  3                          end;
;  5466  3
;  5467  3          [ST_DRV]:       begin                                           ! DRIVE ERROR
;  5468  3
;  5469  3                          if .SB_CODE eql 3
;  5470  3                          then
;  5471  3                              T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  5472  3                          else
;  5473  3                              T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5474  3
;  5475  3                          if .APT_MODE
;  5476  3                          then
;  5477  3                              ERR_HRD_RTNE_APT (41)
;  5478  3                          else
;  5479  3                              ERR_HRD_RTNE (41);
;  5480  3
;  5481  3                          end;
;  5482  3
;  5483  3          [ST_DIA]:       begin                                           ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  5484  3                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  5485  3
;  5486  3                          if .APT_MODE
;  5487  3                          then
;  5488  3                              ERR_HRD_RTNE_APT (43)
;  5489  3                          else
;  5490  3                              ERR_HRD_RTNE (43);
;  5491  3
;  5492  3                          end;
;  5493  3
;  5494  3          [otherwise]:    begin                                           ! PRINT STATUS CODE IF NO MATCH
;  5495  3                          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  5496  3
;  5497  3                          if .APT_MODE
```

```
;    5498  3                        then
;    5499  3                            ERR_HRD_RTNE_APT (45)
;    5500  3                        else
;    5501  3                            ERR_HRD_RTNE (45);
;    5502  3
;    5503  2                        end;
;    5504  2
;    5505  2                tes;
;    5506  2
;    5507  2        if .RP_ADDR [CONID] EQL CID_DUP                                  !FOR DUP ERRORS                          ZZZ
;    5508  2        OR .D_FAIL EQL 1                                                 !EVEN IF UNRECOGNIZABLE AS SUCH          ZZZ
;    5509  2        THEN                                                             !                                        ZZZ
;    5510  2
;    5511  2            selectoneu .RP_ADDR [STSCOD] of
;    5512  2                SET
;    5513  3                [%o'0']         : begin                                  ! if status code succesful
;    5514  3                                  if .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) and ! IF ENDCODE IS GET DUST STATUS
;GRA  5515  3                                     .RP_ADDR [9,9,1,0] NEQ 1                 ! TEST TO SEE IF CONTROLLER LOCAL PRO
;    5516  3                                  then                                     ! (PG 18 OF DUP DOC)
;    5517  4                                      BEGIN
;MED  5518ZZ4                                      ERR_HRD_RTNE (60);               !UNABLE TO LOAD LOCAL CONTROLLER DUP
;    5519  4                                      T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5520  4                                      END
;    5521  3                                  else
;    5522  4                                      begin
;    5523  5                                      if (.DUPPKT [DUPTYPE] eql 5)         ! if fatal error
;    5524  4                                      then
;    5525ZZ4 !ZZZ                                      begin                            !DON'T DROP DEVICE ON DUP ERROR
;    5526ZZ4 !ZZZ                                      DUR [.L$LUN] = DU_DFATAL;        !GIVE F.E. A CHANCE TO SEE ERRORS
;    5527ZZ4 !ZZZ                                      DODU (.L$LUN);                   ! FATAL DEVICE ERROR DROP UNIT);
;    5528ZZ4 !ZZZ                                      end;                             ! SET REASON FOR DROPPING UNIT
;    5529  4                                      selectoneu .DUPPKT [DUPMESG] of
;    5530  4                                          SET
;    5531  5                                          [%o'1'] : begin
;    5532  5                                                    ERR_HRD_RTNE (62);          ! illegal unit number    !ZZZ
;    5533  5                                                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5534  4                                                    end;
;    5535  5                                          [%o'2'] : begin
;ock  5536ZZ5                                                 ERR_HRD_RTNE (63);          ! illegal relative or physical bl
;    5537  5                                                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5538  4                                                    end;
;    5539  5                                          [%o'3'] : begin
;    5540  5                                                    ERR_HRD_RTNE (64);          ! device error    !ZZZ
;    5541  5                                                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    5542  4                                                    end;
;    5543  5                                          [%o'4'] : begin
;    5544  5                                                    ERR_HRD_RTNE (65);          ! zero lenght message   !ZZZ
;    5545  5                                                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5546  4                                                    end;
;    5547  5                                          [%o'5'] : begin                                                  !MMM
;    5548  5                                                    ERR_HRD_RTNE (64);          ! device error           !MMM
;    5549  5                                                    T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;  !MMM
;    5550  4                                                    end;                                                 !MMM
```

D16


```
                                                        [OTHERWISE] : begin
                                                                ERR_HRD_RTNE (66);      ! DUP UNKNOWN STATUS CODE   !ZZZ
                                                                C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD
                                                                end;
                                                        tes;
                                                end;

                                        end;
                [%o'1']         : begin
                                ERR_HRD_RTNE (67);      ! INVALID COMMAND           !ZZZ
                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
                                end;
                [%o'2']         : begin
                                ERR_HRD_RTNE (68);      ! NO REGION AVAILABLE       !ZZZ
                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
                                end;
                [%o'3']         : begin
                                ERR_HRD_RTNE (69);              ! NO REGION SUITABLE        !ZZZ
                                T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
                                end;
                [%o'4']         : begin
                                ERR_HRD_RTNE (70);      ! PROGRAM NOT KNOWN         !ZZZ
                                T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
                                end;
                [%o'5']         : begin
                                ERR_HRD_RTNE (71);              ! LOAD FAILURE              !ZZZ
                                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
                                end;
                [%o'6']         : begin
                                ERR_HRD_RTNE (72);              ! STANDALONE                !ZZZ
                                T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
                                end;
                [OTHERWISE] : begin
                                ERR_HRD_RTNE (73);              ! DUP UNKNOWN STATUS CODE   !ZZZ
                                C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
                                end;
                TES;

        end;                                                                    ! ROUTINE HARD_ERROR


                                        .SBTTL  HARD.ERROR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G                 HARD.ERROR::
                                        JSR     R1,$SAVE4                       ;                       5286
000004  013701  000000G                 MOV     T.ADDR,R1                       ;                       5302
000010  005261  000014                  INC     14(R1)
000014  013703  000000G                 MOV     RP.ADDR,R3                      ;                       5303
000020  105763  000003                  TSTB    3(R3)
000024  001171                          BNE     12$
000026  013702  000000G                 MOV     ST.CODE,R2                      ;                       5306
000032  001027                          BNE     4$                              ;                       5309
000034  013704  000000G                 MOV     SB.CODE,R4
000040  001563                          BEQ     12$
```

E16


```
000042  012700  000050                          MOV     #50,R0                          ;                       5315
000046  060100                                  ADD     R1,R0
000050  020427  000004                          CMP     R4,#4                           ;                       5313
000054  001002                                  BNE     1$
000056  105210                                  INCB    (R0)                            ;                       5315
000060  000402                                  BR      2$                              ;                       5313
000062  105260  000001                  1$:     INCB    1(R0)                           ;                       5317
000066  032737  000001  001256'         2$:     BIT     #1,APT.MODE                     ;                       5319
000074  001403                                  BEQ     3$
000076  012746  000054                          MOV     #54,-(SP)                       ;                       5321
000102  000557                                  BR      14$
000104  012746  000054                  3$:     MOV     #54,-(SP)                       ;                       5323
000110  000557                                  BR      16$
000112  020227  000001                  4$:     CMP     R2,#1                           ;                       5327
000116  001014                                  BNE     6$
000120  105261  000051                          INCB    51(R1)                          ;                       5328
000124  032737  000001  001256'                 BIT     #1,APT.MODE                     ;                       5330
000132  001403                                  BEQ     5$
000134  012746  000037                          MOV     #37,-(SP)                       ;                       5332
000140  000570                                  BR      20$
000142  012746  000037                  5$:     MOV     #37,-(SP)                       ;                       5334
000146  000571                                  BR      22$
000150  020227  000002                  6$:     CMP     R2,#2                           ;                       5338
000154  001014                                  BNE     8$
000156  105261  000050                          INCB    50(R1)                          ;                       5339
000162  032737  000001  001256'                 BIT     #1,APT.MODE                     ;                       5341
000170  001403                                  BEQ     7$
000172  012746  000040                          MOV     #40,-(SP)                       ;                       5343
000176  000571                                  BR      24$
000200  012746  000040                  7$:     MOV     #40,-(SP)                       ;                       5345
000204  000571                                  BR      26$
000206  020227  000003                  8$:     CMP     R2,#3                           ;                       5349
000212  001036                                  BNE     10$
000214  105261  000050                          INCB    50(R1)                          ;                       5350
000220  032737  000001  001256'                 BIT     #1,APT.MODE                     ;                       5352
000226  001415                                  BEQ     9$
000230  012777  000001  001260'                 MOV     #1,@MAIL.BOX.TESTNUM            ;                       5355
000236  013700  000000G                         MOV     CUOFF,R0                        ;                       5356
000242  006300                                  ASL     R0
000244  063700  000000G                         ADD     CST.ADDR,R0
000250  111077  001262'                         MOVB    (R0),@MAIL.BOX.SUBTST
000254  042777  177760  001262'                 BIC     #177760,@MAIL.BOX.SUBTST
000262  104455                          9$:     TRAP    55                              ;                       5359
000264  000022                                  .WORD   22
000266  000000G                                 .WORD   EGD.18
000270  000000G                                 .WORD   EMS.18
000272  013700  000000G                         MOV     L$LUN,R0                        ;                       5360
000276  112760  000005  000000G                 MOVB    #5,DUR(R0)
000304  104451                                  TRAP    51                              ;                       5361
000306  000440                                  BR      12$                             ;                       5306
000310  020227  000004                  10$:    CMP     R2,#4                           ;                       5364
000314  001037                                  BNE     13$
000316  105261  000050                          INCB    50(R1)                          ;                       5365
```

F16


```
000322  032737  000001  001256'          BIT     #1,APT.MODE                     ;                       5367
000330  001415                           BEQ     11$
000332  012777  000001  001260'          MOV     #1,@MAIL.BOX.TESTNUM            ;                       5370
000340  013700  000000G                  MOV     CUOFF,R0                        ;                       5371
000344  006300                           ASL     R0
000346  063700  000000G                  ADD     CST.ADDR,R0
000352  111077  001262'                  MOVB    (R0),@MAIL.BOX.SUBTST
000356  042777  177760  001262'          BIC     #177760,@MAIL.BOX.SUBTST
000364  104455                  11$:     TRAP    55                              ;                       5374
000366  000030                           .WORD   30
000370  000000G                          .WORD   EGD.18
000372  000000G                          .WORD   EMS.24
000374  013700  000000G                  MOV     L$LUN,R0                        ;                       5375
000400  112760  000005  000000G          MOVB    #5,DUR(R0)
000406  104451                           TRAP    51
000410  000137  023436'          12$:    JMP     51$                             ;                       5376
000414  020227  000005          13$:    CMP     R2,#5                           ;                       5306
000420  001014                           BNE     17$                             ;                       5379
000422  105261  000046                   INCB    46(R1)
000426  032737  000001  001256'          BIT     #1,APT.MODE                     ;                       5380
000434  001403                           BEQ     15$                             ;                       5382
000436  012746  000043                   MOV     #43,-(SP)
000442  000564                  14$:     BR      35$                             ;                       5384
000444  012746  000043          15$:    MOV     #43,-(SP)
000450  000564                  16$:     BR      37$                             ;                       5386
000452  020227  000006          17$:    CMP     R2,#6
000456  001026                           BNE     23$                             ;                       5390
000460  012700  000050                   MOV     #50,R0
000464  060100                           ADD     R1,R0                           ;                       5394
000466  023727  000000G 000200           CMP     SB.CODE,#200                    ;                       5392
000474  001003                           BNE     18$
000476  105260  000001                   INCB    1(R0)                           ;                       5394
000502  000401                           BR      19$                             ;                       5392
000504  105210                  18$:     INCB    (R0)                            ;                       5396
000506  032737  000001  001256'  19$:    BIT     #1,APT.MODE                     ;                       5398
000514  001404                           BEQ     21$
000516  012746  000044                   MOV     #44,-(SP)                       ;                       5400
000522  000137  023360'          20$:    JMP     43$
000526  012746  000044          21$:    MOV     #44,-(SP)                       ;                       5402
000532  000416                  22$:     BR      26$
000534  020227  000007          23$:    CMP     R2,#7                           ;                       5406
000540  001014                           BNE     27$
000542  105261  000047                   INCB    47(R1)                          ;                       5407
000546  032737  000001  001256'          BIT     #1,APT.MODE                     ;                       5409
000554  001403                           BEQ     25$
000556  012746  000045                   MOV     #45,-(SP)                       ;                       5411
000562  000561                  24$:     BR      43$
000564  012746  000045          25$:    MOV     #45,-(SP)                       ;                       5413
000570  000561                  26$:     BR      45$
000572  020227  000010          27$:    CMP     R2,#10                          ;                       5417
000576  001054                           BNE     32$
000600  012700  000046                   MOV     #46,R0                          ;                       5421
000604  060100                           ADD     R1,R0
```

```
000606  023727  000000G 000002                  CMP     SB.CODE,#2                      ;                               5419
000614  001002                                  BNE     28$
000616  105210                                  INCB    (R0)
000620  000402                                  BR      29$                             ;                               5421
000622  105260  000001                  28$:    INCB    1(R0)                           ;                               5419
000626  005737  000000G                 29$:    TST     SB.CODE                         ;                               5423
000632  001024                                  BNE     30$                             ;                               5425
000634  132737  000001  000000G                BITB    #1,FORCED.ERROR                 ;                               5426
000642  001020                                  BNE     30$
000644  032737  004000  000000G                BIT     #4000,SWP.FLAGS                 ;                               5427
000652  001414                                  BEQ     30$
000654  112737  000001  000000G                MOVB    #1,FORCED.ERROR                 ;                               5430
000662  016337  000050  000000G                MOV     50(R3),FER0.LBN                 ;                               5431
000670  016337  000052  000000G                MOV     52(R3),FER1.LBN                 ;                               5432
000676  016337  000044  000000G                MOV     44(R3),FER.BC                   ;                               5433
000704  032737  000001  001256'        30$:    BIT     #1,APT.MODE                     ;                               5437
000712  001403                                  BEQ     31$
000714  012746  000046                          MOV     #46,-(SP)                       ;                               5439
000720  000521                                  BR      47$
000722  012746  000046                  31$:    MOV     #46,-(SP)                       ;                               5441
000726  000523                                  BR      49$
000730  020227  000011                  32$:    CMP     R2,#11                          ;                               5445
000734  001014                                  BNE     34$
000736  105261  000051                          INCB    51(R1)                          ;                               5446
000742  032737  000001  001256'                BIT     #1,APT.MODE                     ;                               5448
000750  001403                                  BEQ     33$
000752  012746  000047                          MOV     #47,-(SP)                       ;                               5450
000756  000502                                  BR      47$
000760  012746  000047                  33$:    MOV     #47,-(SP)                       ;                               5452
000764  000504                                  BR      49$
000766  020227  000012                  34$:    CMP     R2,#12                          ;                               5456
000772  001014                                  BNE     38$
000774  105261  000050                          INCB    50(R1)                          ;                               5457
001000  032737  000001  001256'                BIT     #1,APT.MODE                     ;                               5459
001006  001403                                  BEQ     36$
001010  012746  000050                          MOV     #50,-(SP)                       ;                               5461
001014  000463                                  BR      47$
001016  012746  000050                  36$:    MOV     #50,-(SP)                       ;                               5463
001022  000465                          37$:    BR      49$
001024  020227  000013                  38$:    CMP     R2,#13                          ;                               5467
001030  001023                                  BNE     42$
001032  023727  000000G 000003                  CMP     SB.CODE,#3                      ;                               5469
001040  001003                                  BNE     39$
001042  105261  000046                          INCB    46(R1)                          ;                               5471
001046  000402                                  BR      40$                             ;                               5469
001050  105261  000050                  39$:    INCB    50(R1)                          ;                               5473
001054  032737  000001  001256'        40$:    BIT     #1,APT.MODE                     ;                               5475
001062  001403                                  BEQ     41$
001064  012746  000051                          MOV     #51,-(SP)                       ;                               5477
001070  000435                                  BR      47$
001072  012746  000051                  41$:    MOV     #51,-(SP)                       ;                               5479
001076  000437                                  BR      49$
001100  020227  000037                  42$:    CMP     R2,#37                          ;                               5483
```

H16


```
001104  001014                          BNE     46$
001106  105261  000050                  INCB    50(R1)
001112  032737  000001  001256'         BIT     #1,APT.MODE             ;                       5484
001120  001403                          BEQ     44$                     ;                       5486
001122  012746  000053                  MOV     #53,-(SP)               ;                       5488
001126  000416                  43$:    BR      47$
001130  012746  000053          44$:    MOV     #53,-(SP)               ;                       5490
001134  000420                  45$:    BR      49$
001136  013700  000000G         46$:    MOV     CCTLR,R0                ;                       5495
001142  006300                          ASL     R0
001144  105260  000000G                 INCB    C.ERR.TBL(R0)
001150  032737  000001  001256'         BIT     #1,APT.MODE
001156  001405                          BEQ     48$                     ;                       5497
001160  012746  000055                  MOV     #55,-(SP)               ;                       5499
001164  004737  000000V          47$:    JSR     PC,ERR.HRD.RTNE.APT
001170  000404                          BR      50$
001172  012746  000055          48$:    MOV     #55,-(SP)               ;                       5497
001176  004737  000000V          49$:    JSR     PC,ERR.HRD.RTNE         ;                       5501
001202  005726                  50$:    TST     (SP)+
001204  013700  000000G          51$:    MOV     RP.ADDR,R0              ;                       5494
001210  126027  000003  000002          CMPB    3(R0),#2                ;                       5507
001216  001404                          BEQ     52$
001220  123727  000000G 000001          CMPB    D.FAIL,#1
001226  001163                          BNE     70$                     ;                       5508
001230  116001  000016          52$:    MOVB    16(R0),R1               ;                       5511
001234  042701  177740                  BIC     #177740,R1
001240  001072                          BNE     60$                     ;                       5513
001242  126027  000014  000201          CMPB    14(R0),#201             ;                       5514
001250  001015                          BNE     54$
001252  032760  001000  000022          BIT     #1000,22(R0)            ;                       5515
001260  001011                          BNE     54$
001262  012746  000074                  MOV     #74,-(SP)               ;                       5518
001266  004737  000000V          53$:    JSR     PC,ERR.HRD.RTNE
001272  013700  000000G                 MOV     T.ADDR,R0               ;                       5519
001276  105260  000050                  INCB    50(R0)
001302  000534                          BR      69$
001304  013700  000000G          54$:    MOV     DUPPKT,R0               ;                       5517
001310  042700  007777                  BIC     #7777,R0                ;                       5523
001314  020027  050000                  CMP     R0,#50000
001320  001126                          BNE     70$
001322  013701  000000G                 MOV     DUPPKT,R1
001326  042701  170000                  BIC     #170000,R1              ;                       5529
001332  020127  000001                  CMP     R1,#1
001336  001003                          BNE     55$                     ;                       5531
001340  012746  000076                  MOV     #76,-(SP)               ;                       5532
001344  000473                          BR      66$
001346  020127  000002          55$:    CMP     R1,#2                   ;                       5535
001352  001003                          BNE     56$
001354  012746  000077                  MOV     #77,-(SP)               ;                       5536
001360  000465                          BR      66$
001362  020127  000003          56$:    CMP     R1,#3                   ;                       5539
001366  001003                          BNE     58$
001370  012746  000100          57$:    MOV     #100,-(SP)              ;                       5540
```

I16


```
001374  000734                          BR      53$
001376  020127  000004         58$:     CMP     R1,#4                           ;                       5543
001402  001003                          BNE     59$
001404  012746  000101                  MOV     #101,-(SP)                      ;                       5544
001410  000451                          BR      66$
001412  020127  000005         59$:     CMP     R1,#5                           ;                       5547
001416  001764                          BEQ     57$                             ;                       5548
001420  012746  000102                  MOV     #102,-(SP)                      ;                       5552
001424  000454                          BR      68$
001426  020127  000001         60$:     CMP     R1,#1                           ;                       5559
001432  001003                          BNE     61$
001434  012746  000103                  MOV     #103,-(SP)                      ;                       5560
001440  000712                          BR      53$
001442  020127  000002         61$:     CMP     R1,#2                           ;                       5563
001446  001003                          BNE     62$
001450  012746  000104                  MOV     #104,-(SP)                      ;                       5564
001454  000704                          BR      53$
001456  020127  000003         62$:     CMP     R1,#3                           ;                       5567
001462  001003                          BNE     63$
001464  012746  000105                  MOV     #105,-(SP)                      ;                       5568
001470  000421                          BR      66$
001472  020127  000004         63$:     CMP     R1,#4                           ;                       5571
001476  001003                          BNE     64$
001500  012746  000106                  MOV     #106,-(SP)                      ;                       5572
001504  000413                          BR      66$
001506  020127  000005         64$:     CMP     R1,#5                           ;                       5575
001512  001003                          BNE     65$
001514  012746  000107                  MOV     #107,-(SP)                      ;                       5576
001520  000662                          BR      53$
001522  020127  000006         65$:     CMP     R1,#6                           ;                       5579
001526  001011                          BNE     67$
001530  012746  000110                  MOV     #110,-(SP)                      ;                       5580
001534  004737  000000V       66$:     JSR     PC,ERR.HRD.RTNE
001540  013700  000000G                 MOV     T.ADDR,R0                       ;                       5581
001544  105260  000051                  INCB    51(R0)
001550  000411                          BR      69$                             ;                       5579
001552  012746  000111         67$:     MOV     #111,-(SP)                      ;                       5584
001556  004737  000000V       68$:     JSR     PC,ERR.HRD.RTNE
001562  013700  000000G                 MOV     CCTLR,R0                        ;                       5585
001566  006300                          ASL     R0
001570  105260  000000G                 INCB    C.ERR.TBL(R0)
001574  005726                 69$:     TST     (SP)+                           ;                       5583
001576  000207                 70$:     RTS     PC                              ;                       5286

; Routine Size:  448 words,    Routine Base:  $CODE$ + 22232
; Maximum stack depth per invocation:  7 words

;   5590  1
```

J16


```
;    5591  1     GLOBAL routine UPD_IO_TALLY : novalue =
;    5592  1
;    5593  1     !+
;    5594  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    5595  1     !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;    5596  1     !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;    5597  1     !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;    5598  1     !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS DROPPED.
;    5599  1     !
;    5600  1     !       IMPLICIT INPUTS:
;    5601  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5602  1     !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5603  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    5604  1     !               CUOFF - CST OFFSET FOR THE CURRENT UNIT
;    5605  1     !               L$LUN - CURRENT UNIT NUMBER
;    5606  1     !-
;    5607  1
;    5608  2         begin
;    5609  2
;    5610  2         local
;    5611  2             THOUSANDS : word,
;    5612  2             MILLIONS : word,
;    5613  2             FILL_CNT : word,                                    ! TOTAL NO. OF BYTES XFERRED TO/FROM A UNIT
;    5614  2             PARTIAL_CNT : word,                                 ! PARTIAL SECTOR FILL COUNT                  MMM
;    5615  2             BYTES_PER_SECTOR : word initial (512),              ! PARTIAL SECTOR TRANSFER COUNT              MMM
;    5616  2             INDEX : word;                                       ! RD AND RX BYTES PER SECTOR                 MMM
;    5617  2                                                                 ! UNIT NO.                                   MMM
;    5618  2         INDEX = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];       !POINT TO TYPER + TYPEW FOR THIS UNIT    MMM
;    5619  2
;    5620  2         PARTIAL_CNT = .RP_ADDR [BCNT_LO] MOD .BYTES_PER_SECTOR; ! PARTIAL SECTOR COUNT TRANSFERED        MMM
;    5621  2
;    5622  2         if .PARTIAL_CNT eql 0                                   ! CALCULATE ZERO FILL LENGHT ON          MMM
;    5623  2         then FILL_CNT = 0                                       ! PARTIAL SECTOR TRANSFER AND
;    5624  2         else FILL_CNT = .BYTES_PER_SECTOR - .PARTIAL_CNT;       ! ADD TO TALLY COUNT
;    5625  2
;    5626  3         if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)              ! IF ENDCODE IS READ
;    5627  3         then
;    5628  3             begin
;M   5629  3             TYPER [.INDEX] = .TYPER [.INDEX] + 1;               !INCREMENT READ COUNT                    ZZZ M
;    5630  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1; ! INCREMENT NO. OF READS AND ADD BYTE COUNT
;    5631  3             T_ADDR [BYTES_READ_LO] = .T_ADDR [BYTES_READ_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;                ! MMM
;    5632  3             T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;              ! MMM
;    5633  3             OVF_CHK (T_ADDR [TOT_READS_LO]);                    ! CHECK FOR FIELD OVERFLOW
;    5634  3             OVF_CHK (T_ADDR [BYTES_READ_LO]);                   !
;    5635  3             OVF_CHK (T_ADDR [TOT_BYT_RED_LO]);                  !
;    5636  3             end
;    5637  2         else
;    5638  2
;    5639  3             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)         ! IF ENDCODE IS WRITE
;    5640  3             then
;    5641  3                 begin
;M   5642  3                 TYPEW [.INDEX] = .TYPEW [.INDEX] + 1;           !INCREMENT WRITE COUNT                   ZZZ M
;    5643  3                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1;   ! INCREMENT NO. OF WRITES, ADD BYTE COUNT
```

K16


```
;  5644  3                T_ADDR [BYTES_WRIT_LO] = .T_ADDR [BYTES_WRIT_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;                ! MMM
;  5645  3                T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO] + .FILL_CNT ;             ! MMM
;  5646  3                OVF_CHK (T_ADDR [TOT_WRITES_LO]);                                  ! CHECK FOR FIELD OVERFLOW
;  5647  3                OVF_CHK (T_ADDR [BYTES_WRIT_LO]);                                  !
;  5648  3                OVF_CHK (T_ADDR [TOT_BYT_WRT_LO]);                                 !
;  5649  2                end;
;  5650  2
;  5651  2            if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) or
;  5652  3               (.RP_ADDR [ENDCOD] eql (OP_WRT or OP_END))
;  5653  2            then
;  5654  3                begin
;  5655  3                MILLIONS = .T_ADDR [MBYTES_READ] + .T_ADDR [MBYTES_WRIT];          ! TOTAL BYTES TRANSFERRED
;  5656  3                THOUSANDS = .T_ADDR [BYTES_READ_HI] + .T_ADDR [BYTES_WRIT_HI];
;  5657  3
;  5658  3                if .THOUSANDS gequ 1000
;  5659  3                then
;  5660  4                    begin
;  5661  4                    MILLIONS = .MILLIONS + 1;                                      ! COUNT THE LOWER OVERFLOW TOO!
;  5662  4                    THOUSANDS = .THOUSANDS - 1000;
;  5663  3                    end;
;  5664  3
;  5665  3
;  5666  4            IF  (.TYPER [.INDEX] GEQU %o'100000' ) OR (.TYPEW [.INDEX] GEQU %O'100000')   ! RESET COUNTERS AT SOME          MMM
;  5667  3            THEN                                                                          ! CONVENIENT POINT TO AVOID   MMM
;  5668  4               BEGIN                                                                      ! OVERFLOWING                 MMM
;  5669  4               TYPER [.INDEX] = 0;                                                        !                             MMM
;  5670  4               TYPEW [.INDEX] = 0;                                                        !                             MMM
;  5671  4               BAL_IN_PROGRESS [.INDEX] = 0;                                              !                             MMM
;  5672  4               FORCE_WR [.INDEX] = 0;                                                     !                             MMM
;  5673  3               END;                                                                       !                             MMM
;  5674  3                                                                                          !                             MMM
;  5675  4            IF .TYPER [.INDEX] GTRU ((2 * .TYPEW [.INDEX]) + 20)                          ! TOO MANY READS ?            MMM
;  5676  3            THEN                                                                          !                             MMM
;  5677  4               BEGIN                                                                      ! FORCE WRITE OPERATIONS      MMM
;  5678  4               BAL_IN_PROGRESS [.INDEX] = TRUE;                                           !                             MMM
;  5679  4               FORCE_WR [.INDEX] = TRUE;                                                  !                             MMM
;  5680  3               END;                                                                       !                             MMM
;  5681  3                                                                                          !                             MMM
;  5682  3            IF ((2 * .TYPEW [.INDEX]) + 20)  GTRU .TYPER [.INDEX]                         ! TOO MANY WRITES ?           MMM
;  5683  3            THEN                                                                          !                             MMM
;  5684  4               BEGIN                                                                      ! FORCE READ OPERATIONS       MMM
;  5685  4               BAL_IN_PROGRESS [.INDEX] = TRUE;                                           !                             MMM
;  5686  4               FORCE_WR [.INDEX] = FALSE;                                                 !                             MMM
;  5687  3               END;                                                                       !                             MMM
;  5688  3                                                                                          !                             MMM
;  5689  3            IF .TYPER [.INDEX] GTRU ((2 * .TYPEW [.INDEX]) + 5)  AND                      ! ENOUGH WRITES TO STOP       MMM
;  5690  3               BAL_IN_PROGRESS [.INDEX]                                                   ! BALANCE PROCESS ?           MMM
;  5691  3            THEN                                                                          !                             MMM
;  5692  4               BEGIN                                                                      !                             MMM
;  5693  4               BAL_IN_PROGRESS [.INDEX] = FALSE;                                          !                             MMM
;  5694  4               FORCE_WR [.INDEX] = FALSE;                                                 !                             MMM
;  5695  3               END;                                                                       !                             MMM
;  5696  3
```

L16


```
;  5697  3     IF (  ((2 * .TYPEW [.INDEX]) + 5) GTRU .TYPER [.INDEX]  )  AND                  ! ENOUGH READS TO STOP          MMM
;  5698  3        BAL_IN_PROGRESS [.INDEX]                                                    ! BALANCE PROCESS ?             MMM
;  5699  3     THEN                                                                           !                               MMM
;  5700  4        BEGIN                                                                       !                               MMM
;  5701  4        BAL_IN_PROGRESS [.INDEX] = FALSE;                                           !                               MMM
;  5702  4        FORCE_WR [.INDEX] = FALSE;                                                  !                               MMM
;  5703  3        END;                                                                        !                               MMM
;  5704  3                                                                                    !                               MMM
;  5705  3
;  5706  3
;  5707  3
;  5708  3
;  5709  3     !
;  5710  3     ! THIS ADDED BECAUSE IT WILL TAKE FOREVER TO TRANSFER ON THE ORDER OF A MEGABYTE TO A FLOPPY
;  5711  3     ! BUT IT IS A MUCH MORE REASONABLE MEASURE FOR THE RD51/52 WINCHESTER.  THE QUESTION NOW REFERS TO
;  5712  3     ! THE TOTAL DATA TRANSFER TO THE CONTROLLER AND THIS IS PRETTY CLOSE SINCE THE FLOPPIES GET
;  5713  3     ! ABOUT 1/1000 THE DATA THE HARD DISK(S) GET.
;  5714  3     !
;  5715  3
;  5716  3           if .SWP_XFER eql 0                                                       ! IF THERE IS A TRANSFER LIMIT
;  5717  3           then
;  5718  4               begin
;  5719  4
;  5720  4               if .THOUSANDS gtru 50                                                !ZZZ
;  5721  4               then
;  5722  4                   EOP_FLAG = TRUE;                                                 ! SET END-OF-PASS FLAG
;  5723  4
;  5724  4               end
;  5725  3           else
;  5726  3
;  5727  3               if .MILLIONS gequ .SWP_XFER                                          ! IF TRANSFER LIMIT IS REACHED
;  5728  3               then
;  5729  3                   EOP_FLAG = TRUE;                                                 ! SET END-OF-PASS FLAG
;  5730  3
;  5731  2           end;                                                                     ! IF UNIT IS STILL ALIVE
;  5732  2
;  5733  2        !.....................................................................
;  5734  2        !                                                                     .
;  5735  2        ! THE FOLLOWING IS ADDED TO MAKE THE RUN TIME ABOUT 1.5 MINUTES FOR A  .
;  5736  2        ! QUICK PASS IF ALL UNITS UNDER TEST ARE FLOPPIES.                     .
;  5737  2        !                                                                     .
;  5738  2        !.....................................................................
;  5739  2     !!ZZZ   IF .RD_COUNT EQL 0                                                    !IF THERE ARE NO WINCHESTERS   ZZZ
;  5740  2     !!ZZZ      THEN                                                               !                              ZZZ
;  5741  2     !!ZZZ        BEGIN                                                            !                              ZZZ
;  5742  2     !!ZZZ        IF .THOUSANDS GTRU 44                                            !IF ABOUT 1.5 MINUTES GONE BY  ZZZ
;  5743  2     !!ZZZ        THEN                                                             !                              ZZZ
;  5744  2     !!ZZZ           EOP_FLAG = TRUE;                                              !SET THE END OF PASS FLAG      ZZZ
;  5745  2     !!ZZZ        END;                                                             !                              ZZZ
;  5746  2
;  5747  2
;  5748  2        ROUND_OUTPUT ();                                                          ! ROUND TOTALS TO FIT PRINT POSITIONS
;  5749  1        end;                                                                      ! ROUTINE UPD_IO_TALLY
```

M16


```
                                        .SBTTL  UPD.IO,TALLY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       UPD.IO.TALLY::
                                        JSR     R1,$SAVE5                       ;
000004  012701  001000                  MOV     #1000,R1                        ; *,BYTES.PER.SECTO                     5591
000010  013700  000000G                 MOV     CUOFF,R0                        ;                                       5608
000014  006300                          ASL     R0                                                                      5618
000016  063700  000000G                 ADD     CST.ADDR,R0
000022  111003                          MOVB    (R0),R3                         ; *,INDEX
000024  042703  177760                  BIC     #177760,R3                      ; *,INDEX
000030  013700  000000G                 MOV     RP.ADDR,R0                      ;                                       5620
000034  016004  000020                  MOV     20(R0),R4
000040  010446                          MOV     R4,-(SP)
000042  010146                          MOV     R1,-(SP)                        ; BYTES.PER.SECTO,*
000044  004737  000000G                 JSR     PC,BL$MOD
000050  005700                          TST     R0                              ; PARTIAL.CNT                           5622
000052  001002                          BNE     1$
000054  005002                          CLR     R2                              ; FILL.CNT                              5623
000056  000402                          BR      2$                              ;                                       5622
000060  010102              1$:         MOV     R1,R2                           ; BYTES.PER.SECTO,FILL.CNT              5624
000062  160002                          SUB     R0,R2                           ; PARTIAL.CNT,FILL.CNT
000064  013700  000000G       2$:       MOV     RP.ADDR,R0                      ;                                       5626
000070  126027  000014  000241          CMPB    14(R0),#241
000076  001040                          BNE     3$
000100  010300                          MOV     R3,R0                           ; INDEX,*                               5629
000102  006300                          ASL     R0
000104  005260  000000G                 INC     TYPER(R0)
000110  013700  000000G                 MOV     T.ADDR,R0                       ;                                       5630
000114  005260  000016                  INC     16(R0)
000120  010401                          MOV     R4,R1                           ;                                       5631
000122  061001                          ADD     (R0),R1
000124  060201                          ADD     R2,R1                           ; FILL.CNT,*
000126  010110                          MOV     R1,(R0)
000130  010401                          MOV     R4,R1                           ;                                       5632
000132  066001  000032                  ADD     32(R0),R1
000136  060201                          ADD     R2,R1                           ; FILL.CNT,*
000140  010160  000032                  MOV     R1,32(R0)
000144  012716  000016                  MOV     #16,(SP)                        ;                                       5633
000150  060016                          ADD     R0,(SP)
000152  004737  000000V                 JSR     PC,OVF.CHK
000156  013716  000000G                 MOV     T.ADDR,(SP)                     ;                                       5634
000162  004737  000000V                 JSR     PC,OVF.CHK
000166  013716  000000G                 MOV     T.ADDR,(SP)                     ;                                       5635
000172  062716  000032                  ADD     #32,(SP)
000176  000447                          BR      4$
000200  126027  000014  000242  3$:     CMPB    14(R0),#242                     ;                                       5639
000206  001045                          BNE     5$
000210  010300                          MOV     R3,R0                           ; INDEX,*                               5642
000212  006300                          ASL     R0
000214  005260  000000G                 INC     TYPEW(R0)
000220  013700  000000G                 MOV     T.ADDR,R0                       ;                                       5643
000224  005260  000024                  INC     24(R0)
```

```
000230  010401                          MOV     R4,R1                           ;                                       5644
000232  066001  000006                  ADD     6(R0),R1
000236  060201                          ADD     R2,R1                           ; FILL.CNT,*
000240  010160  000006                  MOV     R1,6(R0)
000244  010401                          MOV     R4,R1                           ;                                       5645
000246  066001  000040                  ADD     40(R0),R1
000252  060201                          ADD     R2,R1                           ; FILL.CNT,*
000254  010160  000040                  MOV     R1,40(R0)
000260  012716  000024                  MOV     #24,(SP)                        ;                                       5646
000264  060016                          ADD     R0,(SP)
000266  004737  000000V                 JSR     PC,OVF.CHK
000272  013716  000000G                 MOV     T.ADDR,(SP)                     ;                                       5647
000276  062716  000006                  ADD     #6,(SP)
000302  004737  000000V                 JSR     PC,OVF.CHK
000306  013716  000000G                 MOV     T.ADDR,(SP)                     ;                                       5648
000312  062716  000040                  ADD     #40,(SP)
000316  004737  000000V          4$:    JSR     PC,OVF.CHK
000322  013700  000000G          5$:    MOV     RP.ADDR,R0                      ;                                       5651
000326  126027  000014  000241         CMPB    14(R0),#241
000334  001404                          BEQ     6$
000336  126027  000014  000242         CMPB    14(R0),#242                     ;                                       5652
000344  001134                          BNE     16$
000346  013700  000000G          6$:    MOV     T.ADDR,R0                       ;                                       5655
000352  016005  000004                  MOV     4(R0),R5                        ; *,MILLIONS
000356  066005  000012                  ADD     12(R0),R5                       ; *,MILLIONS
000362  016004  000002                  MOV     2(R0),R4                        ; *,THOUSANDS                           5656
000366  066004  000010                  ADD     10(R0),R4                       ; *,THOUSANDS
000372  020427  001750                  CMP     R4,#1750                        ; THOUSANDS,*                           5658
000376  103403                          BLO     7$
000400  005205                          INC     R5                              ; MILLIONS                              5661
000402  162704  001750                  SUB     #1750,R4                        ; *,THOUSANDS                           5662
000406  010300                  7$:     MOV     R3,R0                           ; INDEX,*                               5666
000410  006300                          ASL     R0
000412  012702  000000G                 MOV     #TYPER,R2
000416  060002                          ADD     R0,R2
000420  021227  100000                  CMP     (R2),#100000
000424  103004                          BHIS    8$
000426  026027  000000G 100000         CMP     TYPEW(R0),#100000
000434  103407                          BLO     9$
000436  005012                  8$:     CLR     (R2)                            ;                                       5669
000440  005060  000000G                 CLR     TYPEW(R0)                       ;                                       5670
000444  005060  000000G                 CLR     BAL.IN.PROGRESS(R0)             ;                                       5671
000450  005060  000000G                 CLR     FORCE.WR(R0)                    ;                                       5672
000454  016003  000000G          9$:    MOV     TYPEW(R0),R3                    ;                                       5675
000460  006303                          ASL     R3
000462  010301                          MOV     R3,R1
000464  062701  000024                  ADD     #24,R1
000470  021201                          CMP     (R2),R1
000472  101406                          BLOS    10$
000474  012760  000001  000000G         MOV     #1,BAL.IN.PROGRESS(R0)          ;                                       5678
000502  012760  000001  000000G         MOV     #1,FORCE.WR(R0)                 ;                                       5679
000510  020112                  10$:    CMP     R1,(R2)                         ;                                       5682
000512  101405                          BLOS    11$
```

```
000514  012760  000001  000000G                 MOV     #1,BAL.IN.PROGRESS(R0)          ;                       5685
000522  005060  000000G                         CLR     FORCE.WR(R0)                    ;                       5686
000526  062703  000005                 11$:    ADD     #5,R3                           ;                       5689
000532  021203                                 CMP     (R2),R3
000534  101411                                 BLOS    12$
000536  010001                                 MOV     R0,R1
000540  062701  000000G                         ADD     #BAL.IN.PROGRESS,R1             ;                       5690
000544  006001                                 ROR     R1
000546  103004                                 BCC     12$
000550  005060  000000G                         CLR     BAL.IN.PROGRESS(R0)             ;                       5693
000554  005060  000000G                         CLR     FORCE.WR(R0)                    ;                       5694
000560  020312                 12$:    CMP     R3,(R2)                         ;                       5697
000562  101411                                 BLOS    13$
000564  010001                                 MOV     R0,R1
000566  062701  000000G                         ADD     #BAL.IN.PROGRESS,R1             ;                       5698
000572  006001                                 ROR     R1
000574  103004                                 BCC     13$
000576  005060  000000G                         CLR     BAL.IN.PROGRESS(R0)             ;                       5701
000602  005060  000000G                         CLR     FORCE.WR(R0)                    ;                       5702
000606  013700  000000G                 13$:    MOV     SWP.XFER,R0                     ;                       5716
000612  001004                                 BNE     14$
000614  020427  000062                         CMP     R4,#62                          ; THOUSANDS,*           5720
000620  101406                                 BLOS    16$
000622  000402                                 BR      15$
000624  020500                 14$:    CMP     R5,R0                           ;                       5722
000626  103403                                 BLO     16$                             ; MILLIONS,*            5727
000630  112737  000001  000000G        15$:    MOVB    #1,EOP.FLAG                     ;                       5729
000636  004737  000000V                16$:    JSR     PC,ROUND.OUTPUT                 ;                       5748
000642  022626                                 CMP     (SP)+,(SP)+                     ;                       5608
000644  000207                                 RTS     PC                              ;                       5591

; Routine Size:  211 words,     Routine Base:  $CODE$ + 24032
; Maximum stack depth per invocation:  9 words
```

```
;   5750  1     GLOBAL routine OVF_CHK (ADDR) : novalue =
;   5751  1     !+
;   5752  1     !       THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;   5753  1     !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;   5754  1     !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;   5755  1     !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;   5756  1     !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;   5757  1     !
;   5758  1     !       INPUTS:
;   5759  1     !               ADDR - ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;   5760  1     !                      THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;   5761  1     !-
;   5762  2         begin
;   5763  2
;   5764  2         while ..ADDR gequ 1000 do                   ! IF LO-ORDER OVERFLOW
;   5765  3             begin
;   5766  3             .ADDR = ..ADDR - 1000;                  ! SUBTRACT 1000
;   5767  3             (.ADDR + 2) = .(.ADDR + 2) + 1;         ! INCR HI-ORDER
;   5768  2             end;
;   5769  2
;   5770  2         if .(.ADDR + 2) gequ 1000                   ! IF HI-ORDER OVERFLOW
;   5771  2         then
;   5772  3             begin
;   5773  3             (.ADDR + 2) = .(.ADDR + 2) - 1000;      ! SUBTRACT 1000
;   5774  3             (.ADDR + 4) = .(.ADDR + 4) + 1;         ! INCREMENT MBYTES
;   5775  2             end;
;   5776  2
;   5777  1         end;                                        ! ROUTINE OVF_CHK


                                .SBTTL  OVF.CHK MULTI-DRIVE TEST ROUTINES
000000  010146         OVF.CHK::
                                MOV     R1,-(SP)                        ;                       5750
000002  016600  000004          MOV     4(SP),R0                        ; ADDR,*                5764
000006  012701  000002          MOV     #2,R1                           ;                       5767
000012  060001                  ADD     R0,R1
000014  021027  001750  1$:     CMP     (R0),#1750                      ;                       5764
000020  103404                  BLO     2$
000022  162710  001750          SUB     #1750,(R0)                      ;                       5766
000026  005211                  INC     (R1)                            ;                       5767
000030  000771                  BR      1$                              ;                       5764
000032  021127  001750  2$:     CMP     (R1),#1750                      ;                       5770
000036  103404                  BLO     3$
000040  162711  001750          SUB     #1750,(R1)                      ;                       5773
000044  005260  000004          INC     4(R0)                           ;                       5774
000050  012601          3$:     MOV     (SP)+,R1                        ;                       5750
000052  000207                  RTS     PC

; Routine Size:  22 words,      Routine Base:  $CODE$ + 24700
; Maximum stack depth per invocation:  2 words
```

```
;    5778  1     GLOBAL routine ROUND_OUTPUT : novalue =
;    5779  1
;    5780  1     !+
;    5781  1     !       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
;    5782  1     !-
;    5783  1
;    5784  2         begin
;    5785  2
;    5786  2         if .T_ADDR [TOT_READS_HI] gtru 9999
;    5787  2         then
;    5788  3             begin
;    5789  3
;    5790  3             if .T_ADDR [TOT_READS_LO] lssu 999
;    5791  3             then
;    5792  4                 begin
;    5793  4                 T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 1;
;    5794  4                 T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
;    5795  3                 end;
;    5796  3
;    5797  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] - 999;
;    5798  3             T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 9999;
;    5799  2             end;
;    5800  2
;    5801  2         if .T_ADDR [TOT_WRITES_HI] gtru 9999
;    5802  2         then
;    5803  3             begin
;    5804  3
;    5805  3             if .T_ADDR [TOT_WRITES_LO] lssu 999
;    5806  3             then
;    5807  4                 begin
;    5808  4                 T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 1;
;    5809  4                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
;    5810  3                 end;
;    5811  3
;    5812  3             T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] - 999;
;    5813  3             T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 9999;
;    5814  2             end;
;    5815  2
;    5816  2         if .T_ADDR [MTOT_BYT_RED] gtru 999
;    5817  2         then
;    5818  3             begin
;    5819  3
;    5820  3             if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    5821  3             then
;    5822  4                 begin
;    5823  4                 T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    5824  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    5825  3                 end;
;    5826  3
;    5827  3             if .T_ADDR [TOT_BYT_RED_LO] lssu 999
;    5828  3             then
;    5829  4                 begin
;    5830  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 1;
```

```
;    5831  4                  T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + 1000;
;    5832  4
;    5833  4                  if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    5834  4                  then
;    5835  5                      begin
;    5836  5                      T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    5837  5                      T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    5838  4                      end;
;    5839  3                  end;
;    5840  3
;    5841  3              T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] - 999;
;    5842  3              T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 999;
;    5843  3              T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 999;
;    5844  2              end;
;    5845  2
;    5846  2          if .T_ADDR [MTOT_BYT_WRT] gtru 999
;    5847  2          then
;    5848  3              begin
;    5849  3
;    5850  3              if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    5851  3              then
;    5852  4                  begin
;    5853  4                  T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5854  4                  T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    5855  3                  end;
;    5856  3
;    5857  3              if .T_ADDR [TOT_BYT_WRT_LO] lssu 999
;    5858  3              then
;    5859  4                  begin
;    5860  4                  T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 1;
;    5861  4                  T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + 1000;
;    5862  4
;    5863  4                  if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    5864  4                  then
;    5865  5                      begin
;    5866  5                      T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    5867  5                      T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    5868  4                      end;
;    5869  3                  end;
;    5870  3
;    5871  3              T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] - 999;
;    5872  3              T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 999;
;    5873  3              T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 999;
;    5874  2              end;
;    5875  2
;    5876  1          end;


                                        .SBTTL  ROUND.OUTPUT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          ROUND.OUTPUT::
                                        JSR     R1,$SAVE3                       ;                                       5778
000004  013700  000000G                 MOV     T.ADDR,R0                       ;                                       5786
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                  ;                       5790
000030  060001                          ADD     R0,R1
000032  021127  001747                  CMP     (R1),#1747
000036  103003                          BHIS    1$
000040  005312                          DEC     (R2)
000042  062711  001750                  ADD     #1750,(R1)              ;                       5793
000046  162711  001747          1$:     SUB     #1747,(R1)              ;                       5794
000052  162712  023417                  SUB     #23417,(R2)             ;                       5797
000056  012702  000026          2$:     MOV     #26,R2                  ;                       5798
000062  060002                          ADD     R0,R2                   ;                       5801
000064  021227  023417                  CMP     (R2),#23417
000070  101415                          BLOS    4$
000072  012701  000024                  MOV     #24,R1                  ;                       5805
000076  060001                          ADD     R0,R1
000100  021127  001747                  CMP     (R1),#1747
000104  103003                          BHIS    3$
000106  005312                          DEC     (R2)
000110  062711  001750                  ADD     #1750,(R1)              ;                       5808
000114  162711  001747          3$:     SUB     #1747,(R1)              ;                       5809
000120  162712  023417                  SUB     #23417,(R2)             ;                       5812
000124  012703  000036          4$:     MOV     #36,R3                  ;                       5813
000130  060003                          ADD     R0,R3                   ;                       5816
000132  021327  001747                  CMP     (R3),#1747
000136  101436                          BLOS    7$
000140  012701  000034                  MOV     #34,R1                  ;                       5820
000144  060001                          ADD     R0,R1
000146  021127  001747                  CMP     (R1),#1747
000152  103003                          BHIS    5$
000154  005313                          DEC     (R3)
000156  062711  001750                  ADD     #1750,(R1)              ;                       5823
000162  012702  000032          5$:     MOV     #32,R2                  ;                       5824
000166  060002                          ADD     R0,R2                   ;                       5827
000170  021227  001747                  CMP     (R2),#1747
000174  103011                          BHIS    6$
000176  005311                          DEC     (R1)
000200  062712  001750                  ADD     #1750,(R2)              ;                       5830
000204  021127  001747                  CMP     (R1),#1747              ;                       5831
000210  103003                          BHIS    6$                      ;                       5833
000212  005313                          DEC     (R3)
000214  062711  001750                  ADD     #1750,(R1)              ;                       5836
000220  162712  001747          6$:     SUB     #1747,(R2)              ;                       5837
000224  162711  001747                  SUB     #1747,(R1)              ;                       5841
000230  162713  001747                  SUB     #1747,(R3)              ;                       5842
000234  012702  000044          7$:     MOV     #44,R2                  ;                       5843
000240  060002                          ADD     R0,R2                   ;                       5846
000242  021227  001747                  CMP     (R2),#1747
000246  101435                          BLOS    10$
000250  012701  000042                  MOV     #42,R1                  ;                       5850
000254  060001                          ADD     R0,R1
000256  021127  001747                  CMP     (R1),#1747
```

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

; Routine Size:  114 words,    Routine Base:  $CODE$ + 24754
; Maximum stack depth per invocation:  5 words
```

```
;    5877  1     GLOBAL routine HOST_WRT_CHK =
;    5878  1
;    5879  1     !+
;    5880  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    5881  1     !       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
;    5882  1     !       OPTION WAS SELECTED BY THE OPERATOR.
;    5883  1     !
;    5884  1     !       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
;    5885  1     !       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
;    5886  1     !       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
;    5887  1     !       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A BYTE-BY-BYTE
;    5888  1     !       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
;    5889  1     !       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
;    5890  1     !
;    5891  1     !       IMPLICIT INPUTS:
;    5892  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    5893  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;    5894  1     !-
;    5895  1
;    5896  2         begin
;    5897  2
;    5898  2         local
;    5899  2             BUFF1 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;    5900  2             BUFF2 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;    5901  2             BUFFW,                                            ! I/O BUFFER ADDRESS
;    5902  2             COUNT : word,                                     ! BYTE COUNT
;    5903  2             FLAG : byte initial (byte (TRUE)),
;    5904  2             index : signed word;
;    5905  2
;    5906  3         if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)           ! IF WRITE OPERATION
;    5907  2         then
;    5908  2             FLAG = FALSE                                      ! DON'T CALL SWEEP FROM IO_RETPKT
;    5909  2         else
;    5910  2
;    5911  2             if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) and
;    5912  3                ((index = RPS_REM ()) geq 0)                   ! IF ASSOCIATED WRITE PACKET IS FOUND
;    5913  2             then
;    5914  3                 begin
;    5915  3                 BUFFW = RETPKT [.index, BUFF_0];              ! ADDR OF ADDR OF WRITE I/O BUFFER
;    5916  3                 BUFF1 = ..BUFFW;                              ! ADDR OF WRITE I/O BUFFER
;    5917  3                 BUFF2 = .RP_ADDR [BUFF_0];                    ! ADDR OF READ I/O BUFFER
;    5918  3                 COUNT = .RP_ADDR [BCNT_LO];                   ! BYTE COUNT
;    5919  3
;    5920  3                 incr I from 1 to .COUNT do                    ! FOR EACH BYTE IN BUFFERS
;    5921  3
;    5922  3                     if .(.BUFF1)<0, 8, 0> eql .(.BUFF2)<0, 8, 0>    ! IF BYTES COMPARE O.K.
;    5923  3                     then
;    5924  4                         begin
;    5925  4                         BUFF1 = .BUFF1 + 1;                   ! ADVANCE WRITE BUFFER ADDR
;    5926  4                         BUFF2 = .BUFF2 + 1;                   ! ADVANCE READ BUFFER ADDR
;    5927  4                         end
;    5928  3                     else
;    5929  4                         begin                                 ! ELSE - COMPARE ERROR
```

```
;    5930  4                    T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;    5931  4                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5932  4
;    5933  4                    EMS_CMP (RETPKT + (.index * RP_LEN * 2));    !MMM PRINT ERROR INFORMATION BEFORE
;    5934  4                                                                 ! HARD ERROR CALL
;    5935  4                                                                 ! IF FLAG:HOE IS SET, THEN ERROR INFORMATION
;    5936  4                                                                 ! IS PRINTED BEFORE HALT !
;    5937  4                    if .APT_MODE
;    5938  4                    then
;    5939  4                        ERR_HRD_RTNE_APT (42)                    ! I/O REQUEST FAILED
;    5940  4                    else
;    5941  4                        ERR_HRD_RTNE (42);
;    5942  4
;    5943  4    !MMM            EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;    5944  4
;    5945  4                    if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;    5946  4                    then
;    5947  5                        begin
;    5948  5                        DUR [.L$LUN] = DU_HERR;                  ! IF ERROR COUNT EXCEEDED
;    5949  5                        DODU (.L$LUN);                           ! DROP UNIT
;    5950  4                        end;
;    5951  4
;    5952  4                    exitloop;                                    ! NO NEED TO CONTINUE
;    5953  3                    end;                                         ! IF COMPARE ERROR
;    5954  3
;    5955  2                end;                                             ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;    5956  2
;    5957  2            return (.FLAG);
;    5958  1            end;                                                 ! ROUTINE HOST_WRT_CHK


                                    .SBTTL  HOST.WRT.CHK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       HOST.WRT.CHK::
                            JSR     R1,$SAVE5                       ;                       5877
000004  005746              TST     -(SP)
000006  112705  000001      MOVB    #1,R5                           ; *,FLAG                5896
000012  013700  000000G     MOV     RP.ADDR,R0                      ;                       5906
000016  126027  000014  000242  CMPB    14(R0),#242
000024  001002              BNE     1$
000026  105005              CLRB    R5                              ; FLAG                  5908
000030  000511              BR      8$                              ;                       5906
000032  126027  000014  000241  1$:     CMPB    14(R0),#241         ;                       5911
000040  001105              BNE     8$
000042  004737  000000V     JSR     PC,RPS.REM                      ;                       5912
000046  005700              TST     R0                              ; INDEX
000050  002501              BLT     8$
000052  010046              MOV     R0,-(SP)                        ; INDEX,*               5915
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           5916
000076  013700  000000G     MOV     RP.ADDR,R0                      ;                       5917
```

K1


```
000102  016002  000024                        MOV     24(R0),R2               ; *,BUFF2
000106  016004  000020                        MOV     20(R0),R4               ; *,COUNT                       5918
000112  005003                                CLR     R3                      ; I                             5920
000114  000453                                BR      6$
000116  121112                        2$:     CMPB    (R1),(R2)               ; BUFF1,BUFF2                   5922
000120  001003                                BNE     3$
000122  005201                                INC     R1                      ; BUFF1                         5925
000124  005202                                INC     R2                      ; BUFF2                         5926
000126  000446                                BR      6$                      ;                               5922
000130  013700  000000G                 3$:    MOV     T.ADDR,R0               ;                               5930
000134  005260  000014                        INC     14(R0)
000140  105260  000051                        INCB    51(R0)                  ;                               5931
000144  016616  000004                        MOV     4(SP),(SP)              ;                               5933
000150  062716  000000G                       ADD     #RETPKT,(SP)
000154  004737  000000G                       JSR     PC,EMS.CMP
000160  032737  000001  001256'               BIT     #1,APT.MODE             ;                               5937
000166  001405                                BEQ     4$
000170  012716  000052                        MOV     #52,(SP)                ;                               5939
000174  004737  000000V                       JSR     PC,ERR.HRD.RTNE.APT
000200  000404                                BR      5$                      ;                               5937
000202  012716  000052                 4$:    MOV     #52,(SP)                ;                               5941
000206  004737  000000V                       JSR     PC,ERR.HRD.RTNE
000212  013700  000000G                 5$:    MOV     T.ADDR,R0               ;                               5945
000216  026037  000014  000000G               CMP     14(R0),SWP.ERROR
000224  103412                                BLO     7$
000226  013700  000000G                       MOV     L$LUN,R0                ;                               5948
000232  112760  000004  000000G               MOVB    #4,DUR(R0)
000240  104451                                TRAP    51                      ;                               5949
000242  000403                                BR      7$                      ;                               5929
000244  005203                        6$:     INC     R3                      ; I                             5920
000246  020304                                CMP     R3,R4                   ; I,COUNT
000250  003722                                BLE     2$
000252  022626                        7$:     CMP     (SP)+,(SP)+             ;                               5914
000254  005000                        8$:     CLR     R0                      ;                               5957
000256  150500                                BISB    R5,R0                   ; FLAG,*
000260  005726                                TST     (SP)+                   ;                               5877
000262  000207                                RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 25320
; Maximum stack depth per invocation:  11 words
```

```
;    5959  1     GLOBAL routine SWEEP : novalue =
;    5960  1
;    5961  1     !+
;    5962  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;    5963  1     !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;    5964  1     !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;    5965  1     !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;    5966  1     !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;    5967  1     !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;    5968  1     !       DEALLOCATED.
;    5969  1     !
;    5970  1     !       IMPLICIT INPUTS:
;    5971  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5972  1     !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;    5973  1     !-
;    5974  1
;    5975  2         begin
;    5976  2
;    5977  2         local
;    5978  2             index : signed word;
;    5979  2
;    5980  2         if (.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD         ! IF READ OPCODE OR ENDCODE
;    5981  2         then
;    5982  2
;    5983  3             if BIT_TST (SWP_FLAGS, SWF_HWC)                 ! IF HOST IS DOING WRITE-COMPARES
;    5984  2             then
;    5985  2
;    5986  2                 if (index = RPS_REM ()) geq 0               ! IF ASSOCIATED WRITE RETPKT IS FOUND
;    5987  2                 then
;    5988  3                     begin
;    5989  3                     PUT_IO_BUFF (RETPKT [.index, BUFF_0]);  ! RETURN WRITE I/O BUFFER TO POOL
;    5990  3                     PUT_RETPKT (.index);                    ! RETURN WRITE PACKET TO POOL
;    5991  2                     end;
;    5992  2
;    5993  2             PUT_IO_BUFF (RP_ADDR [BUFF_0]);                 ! RETURN CURRENT I/O BUFFER TO POOL
;    5994  2             PUT_RETPKT (.RP_INDX);                          ! RETURN CURRENT RETPKT TO POOL
;    5995  1             end;                                            ! ROUTINE SWEEP


                                        .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                 SWEEP:: MOV     R1,-(SP)                        ;                               5959
000002  013700  000000G                 MOV     RP.ADDR,R0                      ;                               5980
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                   ;                               5983
000032  001422                         BEQ     1$
000034  004737  000000V                 JSR     PC,RPS.REM                      ;                               5986
000040  010001                         MOV     R0,R1                           ; *,INDEX
000042  002416                         BLT     1$
000044  C10146                         MOV     R1,-(SP)                        ; INDEX,*                       5989
000046  012746  000054                 MOV     #54,-(SP)
```

M1


```
000052  004737  000000G                        JSR     PC,BL$MUL
000056  062700  000024G                        ADD     #RETPKT+24,R0
000062  010016                                 MOV     R0,(SP)
000064  004737  000000G                        JSR     PC,PUT.IO.BUFF
000070  010116                                 MOV     R1,(SP)                 ; INDEX,*                       5990
000072  004737  000000G                        JSR     PC,PUT.RETPKT
000076  022626                                 CMP     (SP)+,(SP)+             ;                               5988
000100  013746  000000G             1$:        MOV     RP.ADDR,-(SP)           ;                               5993
000104  062716  000024                         ADD     #24,(SP)
000110  004737  000000G                        JSR     PC,PUT.IO.BUFF
000114  013716  000000G                        MOV     RP.INDX,(SP)            ;                               5994
000120  004737  000000G                        JSR     PC,PUT.RETPKT
000124  005726                                 TST     (SP)+                   ;                               5975
000126  012601                                 MOV     (SP)+,R1                ;                               5959
000130  000207                                 RTS     PC

; Routine Size:  45 words,      Routine Base:  $CODE$ + 25604
; Maximum stack depth per invocation:  4 words
```

```
;    5996  1     GLOBAL routine RPS_REM =
;    5997  1
;    5998  1     !+
;    5999  1     !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;    6000  1     !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;    6001  1     !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;    6002  1     !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;    6003  1     !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;    6004  1     !       OPERATION IS RETURNED TO THE CALLER.
;    6005  1     !
;    6006  1     !       IMPLICIT INPUTS:
;    6007  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6008  1     !
;    6009  1     !       OUTPUTS:
;    6010  1     !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;    6011  1     !                       A CRN WHICH IS ONE LESS THAN THE CURRENT
;    6012  1     !-
;    6013  1
;    6014  2         begin
;    6015  2
;    6016  2         local
;    6017  2             index : signed word initial (-1);                          ! ASSUME NOT FOUND
;    6018  2
;    6019  2         incr COUNT from 0 to RP_CNT - 1 do                             ! FOR EACH ENTRY IN RP_SAVE
;    6020  2
;    6021  2             if (.RP_USE [.COUNT] eql .CCTLR) and                       ! IF THIS IS A VALID RETPKT INDEX
;    6022  3                (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;    6023  2             then
;    6024  2
;    6025  3                 if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO] - 1)) and     ! IF CORRECT CRN
;    6026  2                     (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI])) or
;    6027  3                    ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI] - 1)) and
;    6028  3                     (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;    6029  3                     (.RP_ADDR [CRF_LO] eql 0))
;    6030  2                 then
;    6031  3                     begin
;    6032  3                     index = .COUNT;                                    ! INDEX TO BE RETURNED
;    6033  3                     exitloop;                                          ! DONE
;    6034  2                     end;
;    6035  2
;    6036  2         return .index;
;    6037  1         end;                                                           ! ROUTINE RPS_REM


                                .SBTTL  RPS.REM MULTI-DRIVE TEST ROUTINES
000000  004137  000000G    RPS.REM::
                            JSR     R1,$SAVE4                       ;                       5996
000004  012704  177777      MOV     #-1,R4                          ; *,INDEX               6014
000010  005003              CLR     R3                              ; COUNT                 6019
000012  116300  000000G 1$: MOVB    RP.USE(R3),R0                   ; *(COUNT),*            6021
000016  020037  000000G     CMP     R0,CCTLR
000022  001053              BNE     4$
000024  010346              MOV     R3,-(SP)                        ; COUNT,*               6022
```

B2


```
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,*                       6025
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)      ;                               6026
000112  001415                                  BEQ     3$
000114  016102  000006                  2$:     MOV     6(R1),R2                ;                               6027
000120  005302                                  DEC     R2
000122  026002  000006G                         CMP     RETPKT+6(R0),R2
000126  001011                                  BNE     4$
000130  026027  000004G 177777                  CMP     RETPKT+4(R0),#-1        ;                               6028
000136  001005                                  BNE     4$
000140  005761  000004                          TST     4(R1)                   ;                               6029
000144  001002                                  BNE     4$
000146  010304                          3$:     MOV     R3,R4                   ; COUNT,INDEX                   6032
000150  000404                                  BR      5$                      ;                               6031
000152  005203                          4$:     INC     R3                      ; COUNT                         6019
000154  020327  000007                          CMP     R3,#7                   ; COUNT,*
000160  003714                                  BLE     1$
000162  010400                          5$:     MOV     R4,R0                   ; INDEX,*                       6014
000164  000207                                  RTS     PC                      ;                               5996

; Routine Size:  59 words,      Routine Base:  $CODE$ + 25736
; Maximum stack depth per invocation:  8 words
```

C2


```
;  6038  1     GLOBAL routine DR_RETPKT : novalue =
;  6039  1
;  6040  1     !+
;  6041  1     !       THIS ROUTINE IS CALLED BY PROC RETPKT FOR ALL PACKETS ORIGINATING AT
;  6042  1     !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;  6043  1     !       FATAL DEVICE ERRORS.
;  6044  1     !
;  6045  1     !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;  6046  1     !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;  6047  1     !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED.
;  6048  1     !
;  6049  1     !       IMPLICIT INPUTS:
;  6050  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;  6051  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;  6052  1     !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;  6053  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  6054  1     !-
;  6055  1
;  6056  2         begin
;  6057  2
;  6058  2
;  6059  2         PUTA_BUFF ();                           ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;  6060  2
;  6061  2         incr index from 0 to RP_CNT - 1 do      ! FOR EACH ENTRY IN CONTROLLER'S RP_SAVE
;  6062  2
;  6063  2             if .RP_USE [.index] eql .CCTLR      ! IF VALID RETPKT INDEX
;  6064  2             then
;  6065  2                 PUT_RETPKT (.index);            ! RETURN RETPKT TO POOL
;  6066  2
;  6067  2         QIO [.CCTLR] = 0;                       ! CLEAR NO. OF OUTSTANDING QIOs
;  6068  2         CST_ADDR [STATE] = OFFLINE;             ! MARK CST OFFLINE
;  6069  2         DROP_CTLR (.CCTLR, DU_CFATAL);          ! DROP CONTROLLER'S UNITS
;  6070  2         PUT_RETPKT (.RP_INDX);                  ! PUT BACK RETPKT
;  6071  1         end;                                    ! ROUTINE DR_RETPKT


                                .SBTTL  DR.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146          DR.RETPKT::
                        MOV     R1,-(SP)                ;                               6038
000002  004737  000000G JSR     PC,PUTA.BUFF            ;                               6059
000006  005001          CLR     R1                      ; INDEX                         6061
000010  116100  000000G 1$:     MOVB    RP.USE(R1),R0   ; *(INDEX),*                    6063
000014  020037  000000G         CMP     R0,CCTLR
000020  001004                  BNE     2$
000022  010146                  MOV     R1,-(SP)        ; INDEX,*                       6065
000024  004737  000000G         JSR     PC,PUT.RETPKT
000030  005726                  TST     (SP)+
000032  005201          2$:     INC     R1              ; INDEX                         6061
000034  020127  000007          CMP     R1,#7           ; INDEX,*
000040  003763                  BLE     1$
000042  013701  000000G         MOV     CCTLR,R1        ;                               6067
000046  105061  000000G         CLRB    QIO(R1)
000052  013700  000000G         MOV     CST.ADDR,R0     ;                               6068
```

D2


```
000056  042760  100000  000002                  BIC     #100000,2(R0)
000064  010146                                  MOV     R1,-(SP)                ;                       6069
000066  012746  000006                          MOV     #6,-(SP)
000072  004737  000000G                         JSR     PC,DROP.CTLR
000076  013716  000000G                         MOV     RP.INDX,(SP)            ;                       6070
000102  004737  000000G                         JSR     PC,PUT.RETPKT
000106  022626                                  CMP     (SP)+,(SP)+             ;                       6056
000110  012601                                  MOV     (SP)+,R1                ;                       6038
000112  000207                                  RTS     PC

; Routine Size:  38 words,      Routine Base:  $CODE$ + 26124
; Maximum stack depth per invocation:  4 words
```

E2


```
;   6072  1     %sbttl 'RDRX INTERRUPT SERVICE ROUTINES'
;   6073  1
;   6074  1     !+
;   6075  1     !       THERE EXISTS AN RDRX INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;   6076  1     !       CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;   6077  1     !       APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
;   6078  1     !       BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;   6079  1     !-
;   6080  1
;   6081  2     BGNSRV (AZINTO);
;   6082  2     ICTLR = 0;
;   6083  2     AZINT ();
;   6084  1     ENDSRV;


                                        .SBTTL  AZINTO RDRX INTERRUPT SERVICE ROUTINES
000000  010046                 AZINTO::MOV     R0,-(SP)                        ;                       6081
000002  005037  000104'                 CLR     ICTLR                           ;                       6082
000006  004737  000000V                 JSR     PC,AZINT                        ;                       6083
000012  012600                          MOV     (SP)+,R0                        ;                       6081
000014  000002                          RTI

; Routine Size:  7 words,      Routine Base:  $CODE$ + 26240
; Maximum stack depth per invocation:  2 words
```

F2


```
;   6085  1     GLOBAL routine AZINT : novalue =
;   6086  1
;   6087  1     !+
;   6088  1     !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RDRX CONTROLLERS.
;   6089  1     !       AFTER CALCULATING THE DCT ADDRESS FOR THE INTERRUPTING DEVICE, THIS
;   6090  1     !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;   6091  1     !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;   6092  1     !       OTHERWISE, THE SA VALUE IS CHECKED FOR A FATAL ERROR, AND THE COMMAND
;   6093  1     !       AND RESPONSE RINGS ARE POLLED.
;   6094  1     !-
;   6095  1
;   6096  2         begin
;   6097  2         IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);           ! GET DCT ADDRESS
;   6098  2         ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);           ! GET CST ADDRESS
;   6099  2         IRDRX_ADDR = .ICST_ADDR [IP_ADDR];                  ! GET RDRX ADDRESS
;   6100  2         ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2);    ! GET COMM_AREA ADDR
;   6101  2         IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];   ! SAVE SA REGISTER
;   6102  2
;   6103  2         if .IDCT_ADDR [IG_INT]                              ! IGNORE INTERRUPT?
;   6104  2         then
;   6105  2             return;                                         ! RETURN IF INTERRUPTS IGNORED
;   6106  2
;   6107  3         if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)            ! IF FATAL ERROR
;   6108  2         then
;   6109  2             FATAL_ERROR ()
;   6110  2         else
;   6111  3             begin
;   6112  3             POLL_CRING ();                                  ! POLL COMMAND RING
;   6113  3             POLL_RRING ();                                  ! POLL RESPONSE RING
;   6114  2             end;
;   6115  2
;   6116  1         end;


                                .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146          AZINT:: MOV     R1,-(SP)                        ;                       6085
000002  005746                  TST     -(SP)
000004  013701  000104'         MOV     ICTLR,R1                        ;                       6097
000010  010146                  MOV     R1,-(SP)
000012  012746  000022          MOV     #22,-(SP)
000016  004737  000000G         JSR     PC,BL$MUL
000022  062700  000000G         ADD     #DCT,R0
000026  010037  000100'         MOV     R0,IDCT.ADDR
000032  010116                  MOV     R1,(SP)                         ;                       6098
000034  012746  000126          MOV     #126,-(SP)
000040  004737  000000G         JSR     PC,BL$MUL
000044  062700  000000G         ADD     #CST,R0
000050  010037  000076'         MOV     R0,ICST.ADDR
000054  011037  000000G         MOV     (R0),IRDRX.ADDR                 ; ICST.ADDR,*           6099
000060  010116                  MOV     R1,(SP)                         ;                       6100
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                    ;                               6101
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                   6103
000132  001016                                 BNE     2$                              ;                               6085
000134  016601  000010                         MOV     10(SP),R1                       ;                               6107
000140  042701  077777                         BIC     #77777,R1
000144  020127  100000                         CMP     R1,#-100000
000150  001003                                 BNE     1$
000152  004737  000000V                        JSR     PC,FATAL.ERROR                  ;                               6109
000156  000404                                 BR      2$                              ;                               6107
000160  004737  000000V             1$:        JSR     PC,POLL.CRING                   ;                               6112
000164  004737  000000V                        JSR     PC,POLL.RRING                   ;                               6113
000170  062706  000012             2$:        ADD     #12,SP                          ;                               6085
000174  012601                                 MOV     (SP)+,R1
000176  000207                                 RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 26256
; Maximum stack depth per invocation:  7 words


;   6117  1
```

H2


```
;  6118  1
;  6119  1     GLOBAL ROUTINE DUP_RSP : NOVALUE =                                          !ZZZ
;  6120  1
;  6121  1     !+
;  6122  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH DUP RESPONSE
;  6123  1     !       ITS GENERAL PURPOSE IS TO ACT ON A DATAGRAM OR SEQUENTIAL MESSAGE.
;  6124  1     !       IF THE MESSAGE TYPE IS SEQUENTIAL, THE ROUTINE COPIES THE
;  6125  1     !       CONTENTS OF THE MESSAGE ENVELOPE INTO A RETURN PACKET SO THAT THE
;  6126  1     !       ENVELOPE CAN BE RETURNED TO THE CONTROLLER.
;  6127  1     !
;  6128  1     !       IMPLICIT INPUTS:
;  6129  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;  6130  1     !               IPKT_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;  6131  1     !-
;  6132  2         begin
;  6133  2
;  6134  2         local
;  6135  2             R_INDEX : signed word,
;  6136  2             DEBUG,                                                  !ZZZ
;  6137  2             SRC_ADDR,
;  6138  2             DST_ADDR,
;  6139  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  6140  2         !PRINTX (DER34);
;  6141  2
;  6142  2         incr COUNT from 0 to PKT_CNT - 1 do
;  6143  2
;  6144  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;  6145  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  6146  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;  6147  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;  6148  2                (.MSCP_PKT [.COUNT, CONNID] eql CID_DUP) and
;  6149  3                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;  6150  2             then
;  6151  3                 begin
;  6152  3                 P_INDEX = .COUNT;                  ! SET PKT NUMBER
;  6153  3                 exitloop;
;  6154  2                 end;
;  6155  2
;  6156  2             if .P_INDEX lss 0                      ! IF COMMAND NOT FOUND
;  6157  2             then
;  6158  3                 begin
;  6159  3                 PRINTF (DBM108, .IPKT_ADDR [CRN_LO]);   ! UNKNOWN COMMAND REF. NUMBER
;  6160  3                 return;
;  6161  2                 end;
;  6162  2
;  6163  2             if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0    ! IF RETPKT IS NOT AVAILABLE
;  6164  2             then
;  6165  2                 DEBUG = TRUE                       !TO SEE IF THIS PATH TAKEN          ZZZ
;  6166  2     !           PRINTF (DBM112)                    ! "DUP-RSP: RETPKT NOT AVAILABLE"   ZZZ
;  6167  2             else
;  6168  3                 begin
;  6169  3                 SRC_ADDR = .IPKT_ADDR + 6;         ! SET UP COPY (SKIP OVER PKT DESC)
;  6170  3                 R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);   ! START OF ALLOCATED RETPKT
```

I2


```
;   6171  3
;   6172  3              incr COUNT from 1 to RP_LEN do
;   6173  4                  begin
;   6174  4                  .DST_ADDR = ..SRC_ADDR;          ! COPY 1 WORD
;   6175  4                  DST_ADDR = .DST_ADDR + 2;        ! ADVANCE DESTINATION ADDR
;   6176  4                  SRC_ADDR = .SRC_ADDR + 2;        ! ADVANCE SOURCE ADDR
;   6177  3                  end;                             ! COPY LOOP
;   6178  3
;   6179  3              IN_IODQ (.R_INDEX);                  ! PUT RETPKT INDEX INTO IODQ
;   6180  2              end;                                 ! IF RETPKT WAS ALLOCATED
;   6181  2
;   6182  2
;   6183  2          if .P_INDEX geq 0                        ! IF ASSOC CMD PKT WAS FOUND
;   6184  2           then
;   6185  2              PUT_PKT (.P_INDEX);                  ! RETURN COMMAND PACKET TO POOL
;   6186  2
;   6187  1           end;                                    ! ROUTINE DUP-RSP

                                        .SBTTL  DUP.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DUP.RSP::
                                        JSR     R1,$SAVE3                       ;                       6119
000004  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                       6144
000010  005002                          CLR     R2                              ; COUNT                 6142
000012  010246                  1$:     MOV     R2,-(SP)                        ; COUNT,*               6144
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)          ;                       6145
000044  001020                          BNE     2$
000046  026011  000000G                 CMP     MSCP.PKT(R0),(R1)               ;                       6146
000052  001415                          BEQ     2$
000054  105760  000022G                 TSTB    MSCP.PKT+22(R0)                 ;                       6147
000060  100412                          BMI     2$
000062  126027  000011G 000002          CMPB    MSCP.PKT+11(R0),#2              ;                       6148
000070  001006                          BNE     2$
000072  105761  000022                  TSTB    22(R1)                          ;                       6149
000076  100003                          BPL     2$
000100  010237  000000G                 MOV     R2,P.INDEX                      ; COUNT,*               6152
000104  000406                          BR      3$                              ;                       6151
000106  005202                  2$:     INC     R2                              ; COUNT                 6142
000110  020227  000013                  CMP     R2,#13                          ; COUNT,*
000114  003736                          BLE     1$
000116  005737  000000G                 TST     P.INDEX                         ;                       6156
000122  002013                  3$:     BGE     4$
000124  016146  000012                  MOV     12(R1),-(SP)                    ;                       6159
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                           ;                       6160
```

J2


```
000150  000207                          RTS     PC                              ;                                       6158
000152  013746  000104'          4$:     MOV     ICTLR,-(SP)                     ;                                       6163
000156  004737  000000G                  JSR     PC,GET.RETPKT
000162  010001                          MOV     R0,R1                           ; *,R.INDEX
000164  005726                          TST     (SP)+
000166  005701                          TST     R1                              ; R.INDEX
000170  002003                          BGE     5$
000172  012700  000001                  MOV     #1,R0                           ; *,DEBUG                               6165
000176  000425                          BR      7$                              ;                                       6163
000200  013702  000000G          5$:     MOV     IPKT.ADDR,R2                    ; *,SRC.ADDR                            6169
000204  062702  000006                  ADD     #6,R2                           ; *,SRC.ADDR
000210  010146                          MOV     R1,-(SP)                        ; R.INDEX,*                             6170
000212  012746  000054                  MOV     #54,-(SP)
000216  004737  000000G                  JSR     PC,BL$MUL
000222  062700  000000G                  ADD     #RETPKT,R0
000226  010003                          MOV     R0,R3                           ; *,DST.ADDR
000230  012700  000026                  MOV     #26,R0                          ; *,COUNT                               6172
000234  012223                  6$:     MOV     (R2)+,(R3)+                     ; SRC.ADDR,DST.ADDR                     6174
000236  005300                          DEC     R0                              ; COUNT                                 6172
000240  001375                          BNE     6$
000242  010116                          MOV     R1,(SP)                         ; R.INDEX,*                             6179
000244  004737  000000G                  JSR     PC,IN.IODG
000250  022626                          CMP     (SP)+,(SP)+
000252  013700  000000G          7$:     MOV     P.INDEX,R0                      ;                                       6168
000256  002404                          BLT     8$                              ;                                       6183
000260  010046                          MOV     R0,-(SP)
000262  004737  000000G                  JSR     PC,PUT.PKT                      ;                                       6185
000266  005726                          TST     (SP)+
000270  000207                  8$:     RTS     PC                              ;                                       6119

; Routine Size:  93 words,     Routine Base:  $CODE$ + 26456
; Maximum stack depth per invocation:  9 words

;   6188  1
```

K2


```
;  6189  1     GLOBAL routine FATAL_ERROR : novalue =
;  6190  1
;  6191  1     !+
;  6192  1     !       THIS ROUTINE IS CALLED BY THE INTERRUPT SERVICE ROUTINE (AZINT) UPON
;  6193  1     !       DETECTING AN UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER.
;  6194  1     !       ITS PURPOSE IS TO CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF
;  6195  1     !       THE EXERCISER, AND TO INFORM THE "PROGRAM" PORTION OF THE EVENT VIA
;  6196  1     !       RETURN PACKET.
;  6197  1     !
;  6198  1     !       IMPLICIT INPUTS:
;  6199  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;  6200  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;  6201  1     !               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S CST
;  6202  1     !               IRDRX_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S IP REGISTER
;  6203  1     !-
;  6204  1
;  6205  2         begin
;  6206  2
;  6207  2         local
;  6208  2             index : signed word,
;  6209  2             U_SAVE : word;
;  6210  2
;  6211  2         SA_REG = .IDCT_ADDR [SA_SAVE];
;  6212  2         U_SAVE = .L$LUN;                                    ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
;  6213  2         C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;  6214  2
;  6215  2         if .APT_MODE
;  6216  2         then
;  6217  3             begin
;  6218  3             .MAIL_BOX_TESTNUM = 1;
;  6219  3             .MAIL_BOX_SUBTST = 0;
;  6220  2             end;
;  6221  2
;  6222  2         L$LUN = .ICST_ADDR [OF_UN + OF_DATA, D_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
;  6223  2         ERRDF (14, EGD_14, EMS_14);                         ! FATAL CONTROLLER ERROR
;  6224  2         L$LUN = .U_SAVE;                                    ! RESTORE PRE-INTERRUPT CURRENT UNIT
;  6225  2         DRV_CTLERR (.ICTLR);                                ! CLEAN UP DRIVER DATA FOR CONTROLLER
;  6226  2
;  6227  2         if (index = GET_RETPKT (.ICTLR)) lss 0              ! TRY TO GET A RETPKT; IF FAILURE
;  6228  2         then
;  6229  3             PRINTF (DBM18)                                  ! "FATAL_ERROR: RETPKT NOT AVAILABLE"
;  6230  2         else
;  6231  3             begin                                           ! IF RETPKT WAS ALLOCATED
;  6232  3             RETPKT [.index, CONID] = CID_DRIVER;            ! SET CONNECTION ID TO "DRIVER"
;  6233  3             RETPKT [.index, MESTYP] = MT_FATAL;             ! FATAL ERROR
;  6234  3             RETPKT [.index, CTLR] = .ICTLR;                 ! CONTROLLER NUMBER
;  6235  3             IN_IODQ (.index);                               ! LOAD RETPKT INDEX INTO IODQ
;  6236  2             end;                                            ! IF RETPKT WAS ALLOCATED
;  6237  2
;  6238  1         end;                                                ! ROUTINE FATAL_ERR

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

L2


```
000000  004137  000000G                FATAL.ERROR::
                                                JSR     R1,$SAVE2                       ;                               6189
000004  013700  000100'                        MOV     IDCT.ADDR,R0                    ;                               6211
000010  016037  000002  000000G                MOV     2(R0),SA.REG
000016  013701  000000G                        MOV     L$LUN,R1                        ; *,U.SAVE                      6212
000022  013700  000104'                        MOV     ICTLR,R0                        ;                               6213
000026  006300                                ASL     R0
000030  105260  000000G                        INCB    C.ERR.TBL(R0)
000034  032737  000001  001256'                BIT     #1,APT.MODE                     ;                               6215
000042  001405                                BEQ     1$
000044  012777  000001  001260'                MOV     #1,@MAIL.BOX.TESTNUM            ;                               6218
000052  005077  001262'                        CLR     @MAIL.BOX.SUBTST                ;                               6219
000056  013700  000076'            1$:         MOV     ICST.ADDR,R0                    ;                               6222
000062  016002  000006                        MOV     6(R0),R2
000066  000302                                SWAB    R2
000070  042702  177760                        BIC     #177760,R2
000074  010237  000000G                        MOV     R2,L$LUN
000100  104455                                TRAP    55                              ;                               6223
000102  000016                                .WORD   16
000104  000000G                               .WORD   EGD.14
000106  000000G                               .WORD   EMS.14
000110  010137  000000G                        MOV     R1,L$LUN                        ; U.SAVE,*                      6224
000114  013746  000104'                        MOV     ICTLR,-(SP)                     ;                               6225
000120  004737  000000G                        JSR     PC,DRV.CTLERR
000124  013716  000104'                        MOV     ICTLR,(SP)                      ;                               6227
000130  004737  000000G                        JSR     PC,GET.RETPKT
000134  010001                                MOV     R0,R1                           ; *,INDEX
000136  002007                                BGE     2$
000140  012716  000000G                        MOV     #DBM18,(SP)                     ;                               6229
000144  012746  000001                        MOV     #1,-(SP)
000150  010600                                MOV     SP,R0                           ; SP,*
000152  104417                                TRAP    17
000154  000424                                BR      3$                              ;                               6227
000156  010116                        2$:     MOV     R1,(SP)                         ; INDEX,*                       6232
000160  012746  000054                        MOV     #54,-(SP)
000164  004737  000000G                        JSR     PC,BL$MUL
000170  062700  000002G                        ADD     #RETPKT+2,R0
000174  112760  000003  000001               MOVB    #3,1(R0)
000202  013702  000104'                        MOV     ICTLR,R2                        ;                               6234
000206  042702  177760                        BIC     #177760,R2
000212  112710  000060                        MOVB    #60,(R0)
000216  150210                                BISB    R2,(R0)
000220  010116                                MOV     R1,(SP)                         ; INDEX,*                       6235
000222  004737  000000G                        JSR     PC,IN.IODQ
000226  022626                        3$:     CMP     (SP)+,(SP)+                     ;                               6205
000230  000207                                RTS     PC                              ;                               6189

; Routine Size:  77 words,      Routine Base:  $CODE$ + 26750
; Maximum stack depth per invocation:  7 words
```

M2


```
;    6239  1     GLOBAL routine POLL_CRING : novalue =
;    6240  1
;    6241  1     !+
;    6242  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    6243  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    6244  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S COMMAND RING AND CHECK FOR ANY
;    6245  1     !       COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY THE CONTROLLER. SUCH SLOTS
;    6246  1     !       HAVE BEEN RETURNED TO THE HOST, INDICATED BY A ZERO OWNERSHIP BIT. FOR
;    6247  1     !       EACH SLOT THAT HAS BEEN RETURNED TO THE HOST, THE CRING COUNT IS
;    6248  1     !       DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO THE NEXT SLOT IN
;    6249  1     !       THE COMMAND RING.
;    6250  1     !
;    6251  1     !       IMPLICIT INPUTS:
;    6252  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    6253  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    6254  1     !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
;    6255  1     !-
;    6256  1
;    6257  2         begin
;    6258  2         ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER    ZZZ
;    6259  2
;    6260  3         while ((.IDCT_ADDR [CRING_CNT] gtru 0) and                  ! WHILE # OF COMMANDS IN CRING > 0 AND
;    6261  2             not (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) do  ! CURRENT SLOT IS HOST-OWNED
;    6262  3             begin
;    6263  3             IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1;     ! DECREMENT # CMDs IN CRING
;    6264  3             IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;         ! ADVANCE TO NEXT SLOT TO POLL
;    6265  3
;    6266  3             if .IDCT_ADDR [CR_POLL] gtra .IDCT_ADDR [CR_END]        ! IF BEYOND END OF RING
;    6267  3             then
;    6268  3                 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];          ! SET POINTER TO TOP OF CRING
;    6269  3
;    6270  2             end;
;    6271  2
;    6272  2     !ZZZ    ICOM_ADDR [CMD_INT] = 0;                                    ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER
;    6273  1         end;


                                        .SBTTL  POLL.CRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G        POLL.CRING::
                                JSR     R1,$SAVE2                               ;                                               6239
000004  013700  000074'                 MOV     ICOM.ADDR,R0                    ;                                               6258
000010  005060  000004                  CLR     4(R0)
000014  013701  000100'                 MOV     IDCT.ADDR,R1                    ;                                               6260
000020  012702  000016                  MOV     #16,R2                          ;                                               6264
000024  060102                          ADD     R1,R2
000026  105711                1$:       TSTB    (R1)                            ;                                               6260
000030  001422                          BEQ     2$
000032  016100  000016                  MOV     16(R1),R0                       ;                                               6261
000036  016000  000002                  MOV     2(R0),R0
000042  042700  077777                  BIC     #77777,R0
000046  020027  100000                  CMP     R0,#-100000
000052  001411                          BEQ     2$
000054  105311                          DECB    (R1)                            ;                                               6263
```

N2


```
000056  062712  000004                          ADD     #4,(R2)                 ;                       6264
000062  021261  000012                          CMP     (R2),12(R1)             ;                       6266
000066  101757                                  BLOS    1$                      ;
000070  016112  000010                          MOV     10(R1),(R2)             ;                       6268
000074  000754                                  BR      1$                      ;                       6260
000076  000207                          2$:     RTS     PC                      ;                       6239

; Routine Size:  32 words,      Routine Base:  $CODE$ + 27202
; Maximum stack depth per invocation:  4 words
```

```
;    6274  1     GLOBAL routine POLL_RRING : novalue =
;    6275  1
;    6276  1
;    6277  1     !*
;    6278  1     !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    6279  1     !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    6280  1     !       ITS PURPOSE IS TO SCAN THE DEVICE'S RESPONSE RING AND CHECK FOR ANY
;    6281  1     !       SLOTS WHICH HAVE BEEN RETURNED TO THE HOST (OWNERSHIP BIT = 0). FOR
;    6282  1     !       EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS PROCESSED BASED ON ITS
;    6283  1     !       CONNECTION ID (DISK OR DUP). AFTER PROCESSING, THE MESSAGE PACKET
;    6284  1     !       IS RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET
;    6285  1     !       TO 1).
;    6286  1     !
;    6287  1     !       IMPLICIT INPUTS:
;    6288  1     !               ICTLR - NUMBER OF INTERRUPTING CONTROLLER
;    6289  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    6290  1     !-
;    6291  1
;    6292  2         begin
;    6293  2
;    6294  2         local
;    6295  2             TEMP : word;                                                ! MMM
;    6296  2
;Z   6297  2         ICOM_ADDR [RSP_INT] = 0;                                        ! CLR RESPONSE INTERRUPT WRD IN RING HEADER
;    6298  2         TEMP = .PAR_TSD;                                                ! MMM
;    6299  2
;    6300  2         while not (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) do     ! WHILE 0 = 0
;    6301  2             begin
;    6302  2             IPRT_ADDR = ..IDCT_ADDR [RR_POLL] - 10;                     ! ADDRESS OF RESPONSE PACKET
;    6303  3
;    6304  3
;    6305  4     IF NOT (.IPKT_ADDR [CONNID] EQL CID_DUP)                !           ZZZ
;    6306  3             THEN                                            !           ZZZ
;    6307  3             (CREDIT_BAL = .CREDIT_BAL + .IPKT_ADDR [CREDITS]);          !ZZZ
;    6308  3     !IT WAS NOTICE THAT DUP WAS SENDIND BACK CREDITS WHICH IT SHOULD NOT.!ZZZ
;    6309  3             selectoneu .IPKT_ADDR [CONNID] of
;    6310  3                 set
;    6311  3
;    6312  3                 [CID_DISK] :        DISK_RSP ();
;    6313  3
;    6314  3                 [CID_DUP]  :        DUP_RSP  ();                        !ZZZ
;    6315  3
;    6316  3                 [otherwise] :       PRINTF (DBM20, .IPKT_ADDR [CONNID], .IRDRX_ADDR);
;    6317  3                                                                         ! "BAD CONN ID = XXXXX FROM XXXXXX"
;    6318  3                 tes;
;    6319  3
;    6320  3
;    6321  3     !           if .PAR_TSD                                             ! MMM
;    6322  3     !           then
;    6323  3     !               if  not .TEMP
;    6324  3     !               then
;    6325  3     !                   return;                                         ! MMM
;    6326  3
```

C3


```
;   6327  3          IPKT_ADDR [MSGLEN] = MSG_LEN * 2;                          ! RE-INIT PKT FIELDS; MESSAGE LENGTH
;   6328  3          IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;            ! ADVANCE TO HI ORDER WORD OF RING SLOT
;   6329  3          .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];                ! RETURN SLOT TO CONTROLLER
;   6330  3          .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED_OWN or ED_FLAG; ! OWNERSHIP TOO
;   6331  3          IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;            ! ADVANCE TO NEXT RRING SLOT
;   6332  3
;   6333  3
;   6334  3          if .IDCT_ADDR [RR_POLL] gtra .IDCT_ADDR [RR_END]           ! IF BEYOND END OF RING
;   6335  3
;   6336  3          then
;   6337  3              IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG];             ! CYCLE TO TOP OF RING
;   6338  3
;   6339  2          end;                                                       ! WHILE LOOP
;   6340  2
;HEA6341  2    !ZZZ  ICOM_ADDR [RSP_INT] = 0;                                        ! CLR RESPONSE INTERRUPT WRD IN RING
;   6342  1      end;


                                        .SBTTL  POLL.RRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G                 POLL.RRING::
                                        JSR     R1,$SAVE3                   ;                                   6274
000004  013700  000074'                 MOV     ICOM.ADDR,R0                ;                                   6297
000010  005060  000006                  CLR     6(R0)
000014  013701  000100'                 MOV     IDCT.ADDR,R1                ;                                   6300
000020  062701  000014                  ADD     #14,R1
000024  011100                          1$:     MOV     (R1),R0
000026  016000  000002                  MOV     2(R0),R0
000032  042700  077777                  BIC     #77777,R0
000036  020027  100000                  CMP     R0,#-100000
000042  001504                          BEQ     6$
000044  017137  000000  000000G         MOV     @0(R1),IPKT.ADDR            ;                                   6302
000052  162737  000012  000000G         SUB     #12,IPKT.ADDR
000060  013700  000000G                 MOV     IPKT.ADDR,R0                ;                                   6305
000064  005002                          CLR     R2
000066  156002  000011                  BISB    11(R0),R2
000072  020227  000002                  CMP     R2,#2
000076  001406                          BEQ     2$
000100  116003  000010                  MOVB    10(R0),R3                   ;                                   6307
000104  042703  177760                  BIC     #177760,R3
000110  060337  000000G                 ADD     R3,CREDIT.BAL
000114  005702                          2$:     TST     R2                          ;                                   6312
000116  001003                          BNE     3$
000120  004737  000000V                 JSR     PC,DISK.RSP
000124  000421                          BR      5$                          ;                                   6309
000126  020227  000002                  3$:     CMP     R2,#2                       ;                                   6314
000132  001003                          BNE     4$
000134  004737  026456'                 JSR     PC,DUP.RSP
000140  000413                          BR      5$                          ;                                   6309
000142  013746  000000G                 4$:     MOV     IRDRX.ADDR,-(SP)            ;                                   6316
000146  010246                          MOV     R2,-(SP)
000150  012746  000000G                 MOV     #DBM20,-(SP)
000154  012746  000003                  MOV     #3,-(SP)
000160  010600                          MOV     SP,R0                       ; SP,*
```

D3


```
000162  104417                          TRAP    17
000164  062706  000010                  ADD     #10,SP
000170  013700  000000G         5$:     MOV     IPKT.ADDR,R0                    ;                       6327
000174  012760  000074  000006         MOV     #74,6(R0)
000202  013702  000100'                 MOV     IDCT.ADDR,R2                    ;                       6328
000206  010201                          MOV     R2,R1
000210  062701  000014                  ADD     #14,R1
000214  062711  000002                  ADD     #2,(R1)
000220  016071  000002  000000         MOV     2(R0),@0(R1)
000226  052771  140000  000000         BIS     #-40000,@0(R1)                  ;                       6329
000234  062711  000002                  ADD     #2,(R1)                         ;                       6330
000240  021162  000006                  CMP     (R1),6(R2)                      ;                       6331
000244  101667                          BLOS    1$                              ;                       6334
000246  016211  000004                  MOV     4(R2),(R1)                      ;                       6337
000252  000664                          BR      1$                              ;                       6300
000254  000207                  6$:     RTS     PC                              ;                       6274

; Routine Size:  87 words,      Routine Base:  $CODE$ + 27302
; Maximum stack depth per invocation:  10 words


;   6343  1
```

```
;    6344  1
;    6345  1
;    6346  1     GLOBAL routine DISK_RSP : novalue =
;    6347  1
;    6348  1
;    6349  1     !+
;    6350  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
;    6351  1     !       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
;    6352  1     !       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
;    6353  1     !       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
;    6354  1     !       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
;    6355  1     !
;    6356  1     !       IMPLICIT INPUTS:
;    6357  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    6358  1     !                           MESSAGE
;    6359  1     !-
;    6360  1
;    6361  1
;    6362  1
;    6363  1         selectoneu .IPKT_ADDR [MSGTYP] of
;    6364  1
;    6365  1
;    6366  1             set
;    6367  1
;    6368  1             [MT_SEQ] :      SEQUEN ();
;    6369  1
;    6370  1             [MT_DG] :       DATAGM ();
;    6371  1
;    6372  1             [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED"
;    6373  1             tes;


                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                  DISK.RSP::
                                        MOV     R1,-(SP)                        ;                       6346
000002  013700  000000G                 MOV     IPKT.ADDR,R0                    ;                       6363
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$                              ;                       6368
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      3$                              ;                       6363
000036  020127  000001          1$:     CMP     R1,#1                           ;                       6370
000042  001003                          BNE     2$
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      3$                              ;                       6363
000052  C10146                  2$:     MOV     R1,-(SP)                        ;                       6372
000054  012746  000000G                 MOV     #DBM21,-(SP)
000060  012746  000002                  MOV     #2,-(SP)
000064  010600                          MOV     SP,R0                           ; SP,*
```

F3


```
000066  104417                          TRAP    17
000070  062706  000006                  ADD     #6,SP
000074  012601                  3$:     MOV     (SP)+,R1                ;                       6346
000076  000207                          RTS     PC

; Routine Size:  32 words,      Routine Base:  $CODE$ + 27560
; Maximum stack depth per invocation:  6 words


;   6374  1
;   6375  1
;   6376  1
```

```
;  6377  1     GLOBAL routine SEQUEN : novalue =
;  6378  1
;  6379  1     !+
;  6380  1     !       THIS ROUTINE IS CALLED BY DISK RSP FOR EACH DISK MSCP RESPONSE MESSAGE
;  6381  1     !       WITH THE "SEQUENTIAL" MESSAGE TYPE. ITS GENERAL PURPOSE IS TO COPY THE
;  6382  1     !       CONTENTS OF THE MESSAGE PACKET INTO A RETURN PACKET SO THAT THE
;  6383  1     !       PACKET CAN BE RETURNED TO THE CONTROLLER.  IN ADDITION,
;  6384  1     !       IF THE COMMAND WAS AN I/O TRANSFER (READ, WRITE, OR ACCESS), THEN SOME
;  6385  1     !       FIELDS OF THE COMMAND PACKET ARE COPIED INTO THE RETURN PACKET.
;  6386  1     !
;  6387  1     !       IMPLICIT INPUTS:
;  6388  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;  6389  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;  6390  1     !-
;  6391  1
;  6392  2         begin
;  6393  2
;  6394  2         local
;  6395  2             P_INDEX : signed word initial (-1),                          ! ASSUME NO ASSOCIATED COMMAND PKT
;  6396  2             R_INDEX : signed word,
;  6397  2             SRC_ADDR,
;  6398  2             DST_ADDR,
;  6399  2             SAV_ADDR,
;  6400  2             R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;  6401  2
;  6402  2         builtin                                                          !MMM
;  6403  2
;  6404  2             HALT;                                                        !MMM
;  6405  2
;  6406  2         incr COUNT from 0 to PKT_CNT - 1 do
;  6407  2
;  6408  2             if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and  ! IF THIS IS THE ASSOC CMD
;  6409  2                (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;  6410  2                (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;  6411  2                ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;  6412  2                (.MSCP_PKT [.COUNT, MSGTYP] eql MT_SEQ) and
;  6413  2                ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END) and
;  6414  3                (.PKT_USE [.COUNT] eql .ICTLR)
;  6415  2             then
;  6416  3                 begin
;  6417  3                 P_INDEX = .COUNT;                                       ! SET PKT NUMBER
;  6418  3                 exitloop;
;  6419  3                 end;
;  6420  2
;  6421  2         if .P_INDEX lss 0                                                ! IF COMMAND NOT FOUND
;  6422  2         then
;  6423  3             begin
;  6424  3             PRINTF (DBM108, .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);   ! UNKNOWN COMMAND REF. NUMBER
;  6425  3             return;
;  6426  3             end;
;  6427  2
;  6428  2         if .MSCP_PKT [.P_INDEX, OPCODE] neq (.IPKT_ADDR [OPCODE] and (not OP_END))  ! IF OPCODE MISMATCH
;  6429  2         then
```

H3


```
;  6430  2          PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);
;  6431  2
;  6432  3          if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
;  6433  2              (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
;  6434  3             ((.IPKT_ADDR [STATUS_CODE] eql ST_SUC) and
;  6435  2              (.IPKT_ADDR [STATUS_SUBCODE] eql 0)) and
;  6436  3             ((.MSCP_PKT [.P_INDEX, BC_LO] neq .IPKT_ADDR [BC_LO]) or
;  6437  3              (.MSCP_PKT [.P_INDEX, BC_HI] neq .IPKT_ADDR [BC_HI]))
;  6438  2          then
;P 6439  2              PRINTF (DBM112,
;P 6440  2                      .MSCP_PKT [.P_INDEX, BC_HI], .MSCP_PKT [.P_INDEX, BC_LO], .IPKT_ADDR [BC_HI], .IPKT_ADDR [BC_LO],
;  6441  2                      .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);
;  6442  2
;  6443  2              if .MSCP_PKT [.P_INDEX, RSP_RECEIVED]
;  6444  2              then
;  6445  3                  begin
;  6446  3                  PRINTF (DBM120, .MSCP_PKT [.P_INDEX, CRN_HI], .MSCP_PKT [.P_INDEX, CRN_LO]);
;  6447  3                  PUT_PKT (.P_INDEX);
;  6448  3                  return;
;  6449  3                  end
;  6450  2              else
;  6451  2                  MSCP_PKT [.P_INDEX, RSP_RECEIVED] = TRUE;                    ! MARK RESPONSE RECEIVED
;  6452  2
;  6453  2              if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                     ! IF RETPKT IS NOT AVAILABLE
;  6454  2              then
;  6455  3                  begin
;  6456  3                  PRINTF (DBM22);                                          ! "SEQUEN: RETPKT NOT AVAILABLE"
;  6457  3                  PUT_PKT (.P_INDEX);
;  6458  3                  return;
;  6459  3                  end
;  6460  2              else
;  6461  3                  begin
;  6462  3                  SRC_ADDR = .IPKT_ADDR + 6;                               ! SET UP COPY (SKIP OVER PKT DESC)
;  6463  3                  R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);    ! START OF ALLOCATED RETPKT
;  6464  3
;  6465  3                  incr COUNT from 1 to RP_LEN do
;  6466  4                      begin
;  6467  4                      .DST_ADDR = ..SRC_ADDR;                              ! COPY 1 WORD
;  6468  4                      DST_ADDR = .DST_ADDR + 2;                            ! ADVANCE DESTINATION ADDR
;  6469  4                      SRC_ADDR = .SRC_ADDR + 2;                            ! ADVANCE SOURCE ADDR
;  6470  4
;  6471  5                      if .IPKT_ADDR [OPCODE] eql (OP_ONL or OP_END)        ! IF THIS IS THE ONLINE END MESSAGE
;  6472  4                      then
;  6473  4                          if .COUNT eql 10                                 ! SKIP OVER RESERVED WORDS
;  6474  4                          then
;  6475  4                              SRC_ADDR = .SRC_ADDR + 4;                    !      IN ONLINE END - MESSAGE
;  6476  3                      end;                                                 ! COPY LOOP
;  6477  3
;  6478  3                  R_ADDR [CTLR] = .ICTLR;                                  ! LOAD CONTROLLER NUMBER INTO PKT
;  6479  3
;  6480  3                  if .P_INDEX geq 0                                        ! IF ASSOC. CMD PKT WAS FOUND
;  6481  3                  then
;  6482  3
```

```
;   6483  3                 if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or                ! IF END MESSAGE IS
;   6484  3                    (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or               !      READ, WRITE, OR
;   6485  4                    (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))                  !      ACCESS
;   6486  3                 then
;   6487  4                     begin
;   6488  4                     R_ADDR [CMDMOD] = .MSCP_PKT [.P_INDEX, MODIFY];              ! COPY
;   6489  4                     R_ADDR [CBCNT_LO] = .MSCP_PKT [.P_INDEX, BC_LO];             ! RELEVANT
;   6490  4                     R_ADDR [CBCNT_HI] = .MSCP_PKT [.P_INDEX, BC_HI];             ! FIELDS
;   6491  4                     R_ADDR [LBN_LO] = .MSCP_PKT [.P_INDEX, LBN_L];               ! FROM
;   6492  4                     R_ADDR [LBN_HI] = .MSCP_PKT [.P_INDEX, LBN_H];               ! COMMAND
;   6493  4                     R_ADDR [BUFF_0] = .MSCP_PKT [.P_INDEX, BUF_0];               ! PACKET
;   6494  4                     R_ADDR [BUFF_1] = .MSCP_PKT [.P_INDEX, BUF_1];               ! TO RETPKT
;   6495  3                     end;                                                         ! IF ENDCODE WAS READ/WRITE/ACCESS
;   6496  3
;   6497  3                 IN_IODQ (.R_INDEX);                                              ! PUT RETPKT INDEX INTO IODQ
;   6498  2                 end;                                                             ! IF RETPKT WAS ALLOCATED
;   6499  2
;   6500  2             if (.IPKT_ADDR [STATUS_CODE] neq  ST_SUC) or
;   6501  3                (.IPKT_ADDR [STATUS_SUBCODE] neq 0)
;   6502  2             then
;   6503  2                 LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_OCCURED                    ! SAVE ERROR CONDITION
;   6504  2             else
;   6505  2                 LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_NOT_OCCURED;           !
;   6506  2
;   6507  2             LAST_PKT [.ICTLR, LAST_CRN_LO] = .IPKT_ADDR [CRN_LO];            ! SAVE COMMAND REFERENCE NUMBER
;   6508  2             LAST_PKT [.ICTLR, LAST_CRN_HI] = .IPKT_ADDR [CRN_HI];            !
;   6509  2         !MMM    SCAN_ERRLOG ();                                              ! PRINT ANY ASSOCIATED ERROR-LOGS
;   6510  2
;   6511  2             if (.R_ADDR [FLAGS] and %o'40') eql %o'40'                       ! MMM IF ERROR LOG(S) GENERATED
;   6512  2             then                                                             ! MMM PRINT RESPONSE PACKET
;   6513  3                 begin
;   6514  3                 PRINTB (DBM123);
;   6515  3                 EMS_R2 (.R_ADDR);
;   6516  3                 EMS_R1 (.R_ADDR);
;   6517  2                 end;
;   6518  2
; MMM6519  2         !MMM    SCAN_ERRLOG ( );                                                 ! PRINT ANY ASSOCIATED ERROR LOGS
;   6520  2
;   6521  2
;   6522  2             if .CSR_MEM and .TST_PAR                                 !MMM
;   6523  2             then
;   6524  3                 if ((..CSR_ADD and %o'100000') eqlu %o'100000')      !MMM
;   6525  2                 then
;   6526  3                     begin
;   6527  3                     PRINTF (DBM125, ..CSR_ADD);
;   6528  3                     .CSR_ADD = %o'40000';                                        !MMM PRINT EXTENDED CSR
;   6529  3                     PRINTF (DBM126, ..CSR_ADD);
;Y E6530  3                     HALT ();                                                     ! MMM HALT ON HOST MEMORY PARIT
;   6531  3                     end;
;   6532  2
;   6533  2             if .P_INDEX geq 0                                            ! IF ASSOC CMD PKT WAS FOUND
;   6534  2             then
;   6535  2                 PUT_PKT (.P_INDEX);                                      ! RETURN COMMAND PACKET TO POOL
```

J3


```
;   6536  2
;   6537  1          end;                                                    ! ROUTINE DISK_RSP

                                        .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G                 SEQUEN::JSR     R1,$SAVE5                       ;                                       6377
000004  005746                                  TST     -(SP)
000006  012746  177777                          MOV     #-1,-(SP)                       ; *,P.INDEX                             6392
000012  013701  000000G                         MOV     IPKT.ADDR,R1                    ;                                       6408
000016  005002                                  CLR     R2                              ; COUNT                                 6406
000020  010246                          1$:     MOV     R2,-(SP)                        ; COUNT,*                               6408
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(RC),14(R1)          ;                                       6409
000052  001024                                  BNE     2$
000054  026011  000000G                         CMP     MSCP.PKT(R0),(R1)               ;                                       6410
000060  001421                                  BEQ     2$
000062  105760  000022G                         TSTB    MSCP.PKT+22(R0)                 ;                                       6411
000066  100416                                  BMI     2$
000070  132760  000360  000010G                 BITB    #360,MSCP.PKT+10(R0)            ;                                       6412
000076  001012                                  BNE     2$
000100  105761  000022                          TSTB    22(R1)                          ;                                       6413
000104  100007                                  BPL     2$
000106  116200  000000G                         MOVB    PKT.USE(R2),R0                  ; *(COUNT),*                            6414
000112  020037  000104'                         CMP     R0,ICTLR
000116  001002                                  BNE     2$
000120  010216                                  MOV     R2,(SP)                         ; COUNT,P.INDEX                         6417
000122  000405                                  BR      3$                              ;                                       6416
000124  005202                          2$:     INC     R2                              ; COUNT                                 6406
000126  020227  000013                          CMP     R2,#13                          ; COUNT,*
000132  003732                                  BLE     1$
000134  005716                                  TST     (SP)                            ; P.INDEX                               6421
000136  002013                          3$:     BGE     4$
000140  016146  000012                          MOV     12(R1),-(SP)                    ;                                       6424
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$                              ;                                       6425
000166  011646                          4$:     MOV     (SP),-(SP)                      ; P.INDEX,*                             6428
000170  012746  000106                          MOV     #106,-(SP)
000174  004737  000000G                         JSR     PC,BL$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
```

K3


```
000226  020203                          CMP     R2,R3
000230  001422                          BEQ     5$
000232  016046  000012                  MOV     12(R0),-(SP)                    ;                               6430
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$:    MOV     IPKT.ADDR,R0                    ;                               6432
000302  126027  000022  000241         CMPB    22(R0),#241
000310  001404                          BEQ     6$
000312  126027  000022  000242         CMPB    22(R0),#242                     ;                               6433
000320  001045                          BNE     8$
000322  012702  000024          6$:    MOV     #24,R2                          ;                               6434
000326  060002                          ADD     R0,R2
000330  132712  000037                  BITB    #37,(R2)
000334  001037                          BNE     8$
000336  032712  177740                  BIT     #177740,(R2)                    ;                               6435
000342  001034                          BNE     8$
000344  026160  000026G 000026         CMP     MSCP.PKT+26(R1),26(R0)          ;                               6436
000352  001004                          BNE     7$
000354  026160  000030G 000030         CMP     MSCP.PKT+30(R1),30(R0)          ;                               6437
000362  001424                          BEQ     8$
000364  016046  000012          7$:    MOV     12(R0),-(SP)                    ;                               6441
000370  016046  000014                  MOV     14(R0),-(SP)
000374  016046  000026                  MOV     26(R0),-(SP)
000400  016046  000030                  MOV     30(R0),-(SP)
000404  016146  000026G                 MOV     MSCP.PKT+26(R1),-(SP)
000410  016146  000030G                 MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G                 MOV     #DBM112,-(SP)
000420  012746  000007                  MOV     #7,-(SP)
000424  010600                          MOV     SP,R0                           ; SP,*
000426  104417                          TRAP    17
000430  062706  000020                  ADD     #20,SP
000434  132761  000001  000005G  8$:    BITB    #1,MSCP.PKT+5(R1)               ;                               6443
000442  001422                          BEQ     10$
000444  016146  000012G                 MOV     MSCP.PKT+12(R1),-(SP)           ;                               6446
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,*                     6447
000474  004737  000000G                 JSR     PC,PUT.PKT
000500  062706  000010          9$:    ADD     #10,SP                          ;                               6448
000504  000137  031176'                 JMP     23$                             ;                               6445
000510  112761  000001  000005G 10$:    MOVB    #1,MSCP.PKT+5(R1)               ;                               6451
000516  013746  000104'                 MOV     ICTLR,-(SP)                     ;                               6453
```

L3


```
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  002010                                 BGE     11$
000542  012746  000000G                        MOV     #DBM22,-(SP)    ;                               6456
000546  012746  000001                         MOV     #1,-(SP)
000552  010600                                 MOV     SP,R0           ; SP,*
000554  104417                                 TRAP    17
000556  000137  031164'                       JMP     21$             ;                               6457
000562  013704  000000G             11$:       MOV     IPKT.ADDR,R4    ; *,SRC.ADDR                    6462
000566  062704  000006                         ADD     #6,R4           ; *,SRC.ADDR
000572  016646  000002                         MOV     2(SP),-(SP)     ; R.INDEX,*                     6463
000576  012746  000054                         MOV     #54,-(SP)
000602  004737  000000G                        JSR     PC,BL$MUL
000606  062700  000000G                        ADD     #RETPKT,R0
000612  010005                                 MOV     R0,R5           ; *,DST.ADDR
000614  010002                                 MOV     R0,R2           ; *,R.ADDR
000616  013700  000000G                        MOV     IPKT.ADDR,R0    ;                               6471
000622  012703  000001                         MOV     #1,R3           ; *,COUNT                       6465
000626  012425             12$:       MOV     (R4)+,(R5)+     ; SRC.ADDR,DST.ADDR             6467
000630  126027  000022  000211                CMPB    22(R0),#211     ;                               6471
000636  001005                                 BNE     13$
000640  020327  000012                         CMP     R3,#12          ; COUNT,*                       6473
000644  001002                                 BNE     13$
000646  062704  000004                         ADD     #4,R4           ; *,SRC.ADDR                    6475
000652  005203             13$:       INC     R3              ; COUNT                         6465
000654  020327  000026                         CMP     R3,#26          ; COUNT,*
000660  003762                                 BLE     12$
000662  013703  000104'                       MOV     ICTLR,R3        ;                               6478
000666  042703  177760                         BIC     #177760,R3
000672  142762  000017  000002                BICB    #17,2(R2)       ; *,*(R.ADDR)
000700  150362  000002                         BISB    R3,2(R2)        ; *,*(R.ADDR)
000704  005766  000004                         TST     4(SP)           ; P.INDEX                       6480
000710  002442                                 BLT     15$
000712  116000  000022                         MOVB    22(R0),R0       ;                               6483
000716  042700  177400                         BIC     #177400,R0
000722  020027  000241                         CMP     R0,#241
000726  001406                                 BEQ     14$
000730  020027  000242                         CMP     R0,#242         ;                               6484
000734  001403                                 BEQ     14$
000736  020027  000220                         CMP     R0,#220         ;                               6485
000742  001025                                 BNE     15$
000744  016162  000024G 000012      14$:       MOV     MSCP.PKT+24(R1),12(R2)  ; *,*(R.ADDR)           6488
000752  016162  000026G 000044                MOV     MSCP.PKT+26(R1),44(R2)  ; *,*(R.ADDR)           6489
000760  016162  000030G 000046                MOV     MSCP.PKT+30(R1),46(R2)  ; *,*(R.ADDR)           6490
000766  016162  000046G 000050                MOV     MSCP.PKT+46(R1),50(R2)  ; *,*(R.ADDR)           6491
000774  016162  000050G 000052                MOV     MSCP.PKT+50(R1),52(R2)  ; *,*(R.ADDR)           6492
001002  016162  000032G 000024                MOV     MSCP.PKT+32(R1),24(R2)  ; *,*(R.ADDR)           6493
001010  016162  000034G 000026                MOV     MSCP.PKT+34(R1),26(R2)  ; *,*(R.ADDR)           6494
001016  016616  000006             15$:       MOV     6(SP),(SP)      ; R.INDEX,*                     6497
001022  004737  000000G                        JSR     PC,IN.IODQ
001026  005726                                 TST     (SP)+           ;                               6461
```

M3


```
001030  013716  000104'                        MOV     ICTLR,(SP)                      ;                               6503
001034  012746  000006                         MOV     #6,-(SP)
001040  004737  000000G                        JSR     PC,BL$MUL
001044  013701  000000G                        MOV     IPKT.ADDR,R1                    ;                               6500
001050  012703  000024                         MOV     #24,R3
001054  060103                                 ADD     R1,R3
001056  132713  000037                         BITB    #37,(R3)
001062  001003                                 BNE     16$
001064  032713  177740                         BIT     #177740,(R3)                    ;                               6501
001070  001404                                 BEQ     17$
001072  012760  000001  000120'       16$:    MOV     #1,LAST.PKT(R0)                 ;                               6503
001100  000402                                 BR      18$                             ;                               6500
001102  005060  000120'               17$:    CLR     LAST.PKT(R0)                    ;                               6505
001106  016160  000012  000122'       18$:    MOV     12(R1),LAST.PKT+2(R0)           ;                               6507
001114  016160  000014  000124'               MOV     14(R1),LAST.PKT+4(R0)           ;                               6508
001122  132762  000040  000015               BITB    #40,15(R2)                      ; *,*(R.ADDR)                   6511
001130  001415                                 BEQ     19$
001132  012716  000000G                        MOV     #DBM123,(SP)                    ;                               6514
001136  012746  000001                         MOV     #1,-(SP)
001142  010600                                 MOV     SP,R0                           ; SP,*
001144  104414                                 TRAP    14
001146  010216                                 MOV     R2,(SP)                         ; R.ADDR,*                      6515
001150  004737  000000G                        JSR     PC,EMS.R2
001154  010216                                 MOV     R2,(SP)                         ; R.ADDR,*                      6516
001156  004737  000000G                        JSR     PC,EMS.R1
001162  005726                                 TST     (SP)+                           ;                               6513
001164  032737  000001  000000G       19$:    BIT     #1,CSR.MEM                      ;                               6522
001172  001441                                 BEQ     20$
001174  032737  000001  000000G               BIT     #1,TST.PAR
001202  001435                                 BEQ     20$
001204  017700  000000G                        MOV     @CSR.ADD,R0                     ;                               6524
001210  042700  077777                         BIC     #77777,R0
001214  020027  100000                         CMP     R0,#-100000
001220  001026                                 BNE     20$
001222  017716  000000G                        MOV     @CSR.ADD,(SP)                   ;                               6527
001226  012746  000000G                        MOV     #DBM125,-(SP)
001232  012746  000002                         MOV     #2,-(SP)
001236  010600                                 MOV     SP,R0                           ; SP,*
001240  104417                                 TRAP    17
001242  012777  040000  000000G               MOV     #40000,@CSR.ADD                 ;                               6528
001250  012716  040000                         MOV     #40000,(SP)                     ;                               6529
001254  012746  000000G                        MOV     #DBM126,-(SP)
001260  012746  000002                         MOV     #2,-(SP)
001264  010600                                 MOV     SP,R0                           ; SP,*
001266  104417                                 TRAP    17
001270  000000                                 HALT                                    ;                               6530
001272  062706  000010                         ADD     #10,SP                          ;                               6526
001276  005766  000004               20$:    TST     4(SP)                           ; P.INDEX                       6533
001302  002404                                 BLT     22$
001304  016616  000004               21$:    MOV     4(SP),(SP)                      ; P.INDEX,*                     6535
001310  004737  000000G                        JSR     PC,PUT.PKT
001314  022626                       22$:    CMP     (SP)+,(SP)+                     ;                               6392
001316  022626                       23$:    CMP     (SP)+,(SP)+                     ;                               6377
```

```
001320  000207                                  RTS     PC

; Routine Size:  361 words,     Routine Base:  $CODE$ + 27660
; Maximum stack depth per invocation:  18 words


;   6538  1
```

```
;  6539  1     GLOBAL routine DATAGM : novalue =
;  6540  1
;  6541  1     !+
;  6542  1     !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;  6543  1     !       THE RDRX
;  6544  1     !
;  6545  1     !       IMPLICIT INPUTS:
;  6546  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
;  6547  1     !                           MESSAGE
;  6548  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;  6549  1     !-
;  6550  1
;  6551  2         begin
;  6552  2
;  6553  2         local
;  6554  2             index : signed word initial (-1),
;  6555  2             SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;  6556  2             SRC_ADDR,
;  6557  2             DST_ADDR,
;  6558  2             TEMP_UNIT,
;  6559  2     !MMM    SFT_ERR_PRINTED : byte initial (byte (FALSE)),
;  6560  2             PACKET_LEN : word;
;  6561  2
;  6562  2         !
;  6563  2         ! FIND AN EMPTY SLOT IN THE ERROR-LOG PACKET SAVE AREA
;  6564  2         !
;  6565  2
;  6566  2             EL_FLUSH [.ICTLR] = TRUE;                                   ! MMM
;  6567  2
;  6568  2             incr COUNT from 0 to EP_CNT - 1 do
;  6569  2
;  6570  2                 if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY            ! IF EMPTY SLOT FOUND
;  6571  2                 then
;  6572  2                     begin
;  6573  2                     index = .COUNT;                                     ! SAVE INDEX INTO THE SAVE AREA
;  6574  2                     exitloop;
;  6575  2                     end;
;  6576  2
;  6577  2             if .index lss 0
;  6578  2             then
;  6579  2                 index = EP_CNT;                                         ! IF NO SLOT FOUND, USE LAST SPARE SLOT
;  6580  2
;  6581  2         !
;  6582  2         ! SAVE THE PACKET CONTENTS
;  6583  2         !
;  6584  2
;  6585  2             SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);               ! ADDRESS OF THE SAVE AREA
;  6586  2             SAVE_ADDR [EL_CONTENTS ] = FULL;                            ! MARK IT FULL
;  6587  2             SAVE_ADDR [EL_CNTR] = .ICTLR;                               ! OWNERSHIP
;  6588  2             SRC_ADDR = .IPKT_ADDR + 6;                                  ! SETUP COPY ADDRESSES
;  6589  2             DST_ADDR = .SAVE_ADDR + 2;                                  !
;  6590  2             PACKET_LEN = ((.IPKT_ADDR [MSGLEN] + 1) / 2) + 2;           ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
;  6591  2
```

```
;  6592  2          if .PACKET_LEN gtru EP_LEN - 1
;  6593  2          then
;  6594  2              PACKET_LEN = EP_LEN - 1;                                    ! ADJUST LENGTH, IF TOO LONG
;  6595  2
;  6596  2          incr COUNT from 1 to .PACKET_LEN do
;  6597  3              begin
;  6598  3              .DST_ADDR = ..SRC_ADDR;                                     ! COPY A WORD
;  6599  3              SRC_ADDR = .SRC_ADDR + 2;                                   ! UPDATE ADDRESS POINTERS
;  6600  3              DST_ADDR = .DST_ADDR + 2;                                   !
;  6601  2              end;
;  6602  2
;  6603  2  !       if (.CSR_MEM and .TST_PAR) and (not .PAR_TSD) and (.SAVE_ADDR [EL_FORMAT] eql 1)   ! MMM
;  6604  2  !       then                                                                               ! MMM
;  6605  2  !           begin                                                                          ! MMM
;  6606  2  !           WRT_RDRX (RCIP, RC_ALL, ALL_ONES);                                             ! MMM
;  6607  2  !           PAR_TSD = TRUE;                                                                ! MMM
;  6608  2  !           INIT_TEST ();                                                                  ! MMM
;  6609  2  !           return;                                                                        ! MMM
;  6610  2  !           end;                                                                           ! MMM
;  6611  2
;  6612  2
;  6613  2          if .SAVE_ADDR [EL_BRR] eql TRUE                                 ! TEST ERROR LOG MESSAGE FLAGS MMM
;  6614  2          then
;  6615  2             PRINTF (DBM127);                                              ! BAD BLOCK REPLACEMENT REQUEST FLAG SET MMM
;  6616  2
;¶ 6617  2          if .SAVE_ADDR [EL_EDR] eql TRUE                                 !ERROR DURING REPLACEMENT (BAD BLOCK) FLAG SET MM
;  6618  2          then
;  6619  2             PRINTF (DBM128);                                             !MMM
;  6620  2
;  6621  2          if .SAVE_ADDR [EL_SUCCESS] eql TRUE                                 !MMM
;  6622  2          then
;  6623  2             if .SAVE_ADDR [EL_FORMAT] lequ 4
;  6624  2             then
;  6625  3                 begin
;  6626  3  !DDD              SOFT_ERROR (.index);                                        ! UPDATE SOFT ERROR COUNT
;  6627  3                  TEMP_UNIT = .L$LUN;                                 ! SAVE UNIT NUMBER AS KNOWN TO DRS
;  6628  3
;  6629  3                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  6630  3
;  6631  3                     if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;  6632  4                        (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  6633  3                     then
;  6634  4                        begin
;  6635  4                        L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];        ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;  6636  4                        exitloop;
;  6637  3                        end;
;  6638  3
;  6639  3                     case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;  6640  3                        set
;  6641  3
;  6642  3                        [0] :  if .APT_MODE                              ! CONTROLLER ERROR
;  6643  3                               then
;  6644  3                                  ERR_SOFT_RTNE_APT (50, .index)
```

```
;  6645  3                        else
;  6646  3                           ERR_SOFT_RTNE (50);
;  6647  3
;  6648  3                     [1] : if .APT_MODE                              ! HOST MEMORY ACCESS ERROR
;  6649  3                        then
;  6650  3                           ERR_SOFT_RTNE_APT (51, .index)
;  6651  3                        else
;  6652  3                           ERR_SOFT_RTNE (51);
;  6653  3
;  6654  3                     [2] : if .APT_MODE                              ! DISK TRANSFER ERROR
;  6655  3                        then
;  6656  3                           ERR_SOFT_RTNE_APT (52, .index)
;  6657  3                        else
;  6658  3                           ERR_SOFT_RTNE (52);
;  6659  3
;  6660  3                     [3] : if .APT_MODE                              ! SDI ERROR
;  6661  3                        then
;  6662  3                           ERR_SOFT_RTNE_APT (53, .index)
;  6663  3                        else
;  6664  3                           ERR_SOFT_RTNE (53);
;  6665  3
;  6666  3                     [4] : if .APT_MODE                              ! SMALL DISK ERROR
;  6667  3                        then
;  6668  3                           ERR_SOFT_RTNE_APT (54, .index)
;  6669  3                        else
;  6670  3                           ERR_SOFT_RTNE (54);
;  6671  3                     tes;
;  6672  3
;  6673  3
;  6674  3                  L$LUN = .TEMP_UNIT;                               ! RESTORE UNIT NUMBER
;  6675  2                  eno;
;  6676  2
;  6677  2            !MMM     else
;  6678  2            !MMM        PRINTF (DBM109, .SAVE_ADDR [EL_FORMAT]);   ! ERROR LOG FORMAT UNKNOWN
;  6679  2
;  6680  2               PRINTB (CRLF);                                       ! EXTRA CARRIEGE-RETURN/LINE-FEED
;  6681  2            !DDD    EMS_EL (.index);                                !       ! PRINT PACKET CONTENTS
;  6682  1               end;                                                 !


                                        .SBTTL  DATAGM RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DATAGM::JSR     R1,$SAVE5                       ;                               6539
000004  012704  177777                  MOV     #-1,R4                          ; *,INDEX                       6551
000010  013700  000104'                 MOV     ICTLR,R0                        ;                               6566
000014  006300                          ASL     R0
000016  012760  000001  000106'         MOV     #1,EL.FLUSH(R0)
000024  005001                          CLR     R1                              ; COUNT                         6568
000026  010146                  1$:     MOV     R1,-(SP)                        ; COUNT,*                       6570
000030  012746  000102                  MOV     #102,-(SP)
000034  004737  000000G                 JSR     PC,BL$MUL
000040  022626                          CMP     (SP)+,(SP)+
000042  105760  000001G                 TSTB    ELOG.PKT+1(R0)
000046  001002                          BNE     2$
```

```
000050  010104                          MOV     R1,R4                   ; COUNT,INDEX                   6573
000052  000405                          BR      3$                      ;                               6572
000054  005201                  2$:     INC     R1                      ; COUNT                         6568
000056  020127  000013                  CMP     R1,#13                  ; COUNT,*
000062  003761                          BLE     1$
000064  005704                          TST     R4                      ; INDEX                         6577
000066  002002                  3$:     BGE     4$
000070  012704  000014                  MOV     #14,R4                  ; *,INDEX                       6579
000074  010446                  4$:     MOV     R4,-(SP)                ; INDEX,*                       6585
000076  012746  000102                  MOV     #102,-(SP)
000102  004737  000000G                 JSR     PC,BL$MUL
000106  062700  000000G                 ADD     #ELOG.PKT,R0
000112  010001                          MOV     R0,R1                   ; *,SAVE.ADDR
000114  111761  000001                  MOVB    (PC),1(R1)              ; *,*(SAVE.ADDR)                6586
000120  113711  000104'                 MOVB    ICTLR,(R1)              ; *,SAVE.ADDR                   6587
000124  013700  000000G                 MOV     IPKT.ADDR,R0            ;                               6588
000130  012705  000006                  MOV     #6,R5                   ; *,SRC.ADDR
000134  060005                          ADD     R0,R5                   ; *,SRC.ADDR
000136  012703  000002                  MOV     #2,R3                   ; *,DST.ADDR                    6589
000142  060103                          ADD     R1,R3                   ; SAVE.ADDR,DST.ADDR
000144  016016  000006                  MOV     6(R0),(SP)              ;                               6590
000150  005216                          INC     (SP)
000152  012746  000002                  MOV     #2,-(SP)
000156  004737  000000G                 JSR     PC,BL$DIV
000162  062700  000002                  ADD     #2,R0
000166  020027  000040                  CMP     R0,#40                  ; PACKET.LEN,*                  6592
000172  101402                          BLOS    5$
000174  012700  000040                  MOV     #40,R0                  ; *,PACKET.LEN                  6594
000200  005002                  5$:     CLR     R2                      ; COUNT                         6596
000202  000401                          BR      7$
000204  012523                  6$:     MOV     (R5)+,(R3)+             ; SRC.ADDR,DST.ADDR             6598
000206  005202                  7$:     INC     R2                      ; COUNT                         6596
000210  020200                          CMP     R2,R0                   ; COUNT,PACKET.LEN
000212  003774                          BLE     6$
000214  012702  000016                  MOV     #16,R2                  ;                               6613
000220  060102                          ADD     R1,R2                   ; SAVE.ADDR,*
000222  032712  020000                  BIT     #20000,(R2)
000226  001407                          BEQ     8$
000230  012716  000000G                 MOV     #DBM127,(SP)            ;                               6615
000234  012746  000001                  MOV     #1,-(SP)
000240  010600                          MOV     SP,R0                   ; SP,*
000242  104417                          TRAP    17
000244  005726                          TST     (SP)+
000246  032712  010000          8$:     BIT     #10000,(R2)             ;                               6617
000252  001407                          BEQ     9$
000254  012716  000000G                 MOV     #DBM128,(SP)            ;                               6619
000260  012746  000001                  MOV     #1,-(SP)
000264  010600                          MOV     SP,R0                   ; SP,*
000266  104417                          TRAP    17
000270  005726                          TST     (SP)+
000272  005712                  9$:     TST     (R2)                    ;                               6621
000274  100133                          BPL     27$
000276  121227  000004                  CMPB    (R2),#4                 ;                               6623
```

F4


```
000302  101130                          BHI     27$
000304  013705  000000G                 MOV     L$LUN,R5                ; *,TEMP.UNIT                   6627
000310  012703  000006                  MOV     #6,R3                   ; *,OFFSET                      6629
000314  010300                  10$:     MOV     R3,R0                   ; OFFSET,*                      6631
000316  063700  000076'                  ADD     ICST.ADDR,R0
000322  016146  000012                  MOV     12(R1),-(SP)            ; *(SAVE.ADDR),*
000326  111046                          MOVB    (R0),-(SP)
000330  042716  177760                  BIC     #177760,(SP)
000334  022626                          CMP     (SP)+,(SP)+
000336  001012                          BNE     11$
000340  032710  040000                  BIT     #40000,(R0)             ;                               6632
000344  001407                          BEQ     11$
000346  011046                          MOV     (R0),-(SP)              ;                               6635
000350  000316                          SWAB    (SP)
000352  042716  177760                  BIC     #177760,(SP)
000356  012637  000000G                 MOV     (SP)+,L$LUN
000362  000405                          BR      12$                     ;                               6634
000364  062703  000024          11$:    ADD     #24,R3                  ; *,OFFSET                      6629
000370  020327  000102                  CMP     R3,#102                 ; OFFSET,*
000374  003747                          BLE     10$
000376  005000                  12$:    CLR     R0                      ;                               6642
000400  153700  001256'                 BISB    APT.MODE,R0
000404  005001                          CLR     R1                      ;                               6639
000406  151201                          BISB    (R2),R1
000410  006301                          ASL     R1
000412  066107  000000'                 ADD     P.AAA(R1),PC            ; Case dispatch
000416  032700  000001          14$:    BIT     #1,R0                   ;                               6642
000422  001403                          BEQ     15$
000424  012716  000062                  MOV     #62,(SP)                ;                               6644
000430  000442                          BR      23$
000432  012716  000062          15$:    MOV     #62,(SP)                ;                               6646
000436  000446                          BR      25$
000440  032700  000001          16$:    BIT     #1,R0                   ;                               6648
000444  001403                          BEQ     17$
000446  012716  000063                  MOV     #63,(SP)                ;                               6650
000452  000431                          BR      23$
000454  012716  000063          17$:    MOV     #63,(SP)                ;                               6652
000460  000435                          BR      25$
000462  032700  000001          18$:    BIT     #1,R0                   ;                               6654
000466  001403                          BEQ     19$
000470  012716  000064                  MOV     #64,(SP)                ;                               6656
000474  000420                          BR      23$
000476  012716  000064          19$:    MOV     #64,(SP)                ;                               6658
000502  000424                          BR      25$
000504  032700  000001          20$:    BIT     #1,R0                   ;                               6660
000510  001403                          BEQ     21$
000512  012716  000065                  MOV     #65,(SP)                ;                               6662
000516  000407                          BR      23$
000520  012716  000065          21$:    MOV     #65,(SP)                ;                               6664
000524  000413                          BR      25$
000526  006000                  22$:    ROR     R0                      ;                               6666
000530  103007                          BCC     24$
000532  012716  000066                  MOV     #66,(SP)                ;                               6668
```

```
000536  010446                          23$:    MOV     R4,-(SP)                ; INDEX,*
000540  004737  000000V                        JSR     PC,ERR.SOFT.RTNE.APT
000544  005726                                TST     (SP)+
000546  000404                                BR      26$                     ;                               6666
000550  012716  000066                  24$:    MOV     #66,(SP)                ;                               6670
000554  004737  000000V                  25$:    JSR     PC,ERR.SOFT.RTNE
000560  010537  000000G                  26$:    MOV     R5,L$LUN                ; TEMP.UNIT,*                   6674
000564  012716  000000G                  27$:    MOV     #CRLF,(SP)              ;                               6680
000570  012746  000001                          MOV     #1,-(SP)
000574  010600                                MOV     SP,R0                   ; SP,*
000576  104414                                TRAP    14
000600  062706  000010                        ADD     #10,SP                  ;                               6551
000604  000207                                RTS     PC                      ;                               6539

; Routine Size:  195 words,    Routine Base:  $CODE$ + 31202
; Maximum stack depth per invocation:  12 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                        P.AAA:
000000  000000                  13$:    .WORD   0                       ; CASE Table for DATAGM+0412            6639
000002  000022                          .WORD   22                      ; [14$]
000004  000044                          .WORD   44                      ; [16$]
000006  000066                          .WORD   66                      ; [18$]
000010  000110                          .WORD   110                     ; [20$]
                                                                        ; [22$]



;    6683  1
;    6684  1
;    6685  1
;    6686  1
;    6687  1
;    6688  1
```

```
;  6689  1
;  6690  1     routine ERR_HRD_RTNE (ERRNUM) : novalue =
;  6691  1
;  6692  1     !+
;  6693  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;  6694  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;  6695  1     !-
;  6696  1
;  6697  2         begin
;  6698  2
;  6699  2         local
;  6700  2             CUR_PRIORITY : word;
;  6701  2
;  6702  2         builtin
;  6703  2             PC;
;  6704  2
;  6705  2         GETPRI (CUR_PRIORITY);
;  6706  2     !ZZZ    SETPRI (PRI04);                          ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;  6707  2         SETPRI (.BRLEVEL);                           ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW    ZZZ
;  6708  2
;  6709  2         if (.ERRNUM lequ 34) or                      ! FOR NON-BAD BLOCK TYPE ERRORS
;  6710  2            (.ERRNUM gtru 38) or
;  6711  2            (.ERRNUM eql 36) or
;  6712  3            (.ERRNUM eql 37)
;  6713  2         then
;  6714  2
;  6715  3             if BIT_TST (SWP_FLAGS, SWF_HRD)          ! IF ERRORS TO BE TREATED NORMALLY
;  6716  2             then
;  6717  2
;  6718  2     !ZZZ        case .ERRNUM from 31 to 45 of
;  6719  2                 case .ERRNUM from 31 to 73 of        !INCLUDE DUP NUMBERS (60-73)          ZZZ
;  6720  2             set
;  6721  2
;  6722  2                 [31]:   ERRHRD (31, EGH_30, EMS_30); ! INVALID COMMAND
;  6723  2
;  6724  2                 [32]:   ERRHRD (32, EGH_30, EMS_30); ! COMMAND ABORTED
;  6725  2
;  6726  2                 [33]:   ;                            !
;  6727  2
;  6728  2                 [34]:   ;                            !
;  6729  2
;  6730  2                 [35]:   ;                            ! MEDIA FORMAT ERROR
;  6731  2
;  6732  2                 [36]:   ERRHRD (36, EGH_30, EMS_30); ! WRITE PROTECTED
;  6733  2
;  6734  2                 [37]:   ERRHRD (37, EGH_30, EMS_30); ! COMPARE ERROR
;  6735  2
;  6736  2                 [38]:   ;                            ! DATA ERROR
;  6737  2
;  6738  2                 [39]:   ERRHRD (39, EGH_30, EMS_30); ! HOST BUFFER ACCESS ERROR
;  6739  2
;  6740  2                 [40]:   ERRHRD (40, EGH_30, EMS_30); ! CONTROLLER ERROR
;  6741  2
```

I4


```
;  6742  2                    [41]:   ERRHRD (41, EGH_30, EMS_30);    ! DRIVE ERROR
;  6743  2
;  6744  2                    [42]:   ERRHRD (42, EGH_30, 0);         ! HOST WRITE COMPARE ERROR
;  6745  2
;  6746  2                    [43]:   ERRHRD (43, EGH_30, EMS_30);    ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  6747  2
;  6748  2                    [44]:   ERRHRD (44, EGH_30, EMS_30);    ! DUPLICATE UNIT NUMBER
;  6749  2
;  6750  2                    [45]:   ERRHRD (45, EGH_30, EMS_30);    ! INVALID END CODE
;  6751  2
;  6752  2                    [46]:   ;                               !LEAVE ROOM FOR SOFT ERROR NUMBERS AND SOME PADDING  ZZZ
;  6753  2                    [47]:   ;                               !
;  6754  2                    [48]:   ;                               !
;  6755  2                    [49]:   ;                               !
;  6756  2                    [50]:   ;                               !
;  6757  2                    [51]:   ;                               !
;  6758  2                    [52]:   ;                               !
;  6759  2                    [53]:   ;                               !
;  6760  2                    [54]:   ;                               !
;  6761  2                    [55]:   ;                               !
;  6762  2                    [56]:   ;                               !
;  6763  2                    [57]:   ;                               !
;  6764  2                    [58]:   ;                               !
;  6765  2                    [59]:   ;                               !
;  6766  2
;  6767  2                    [60]:   ERRHRD (60, EH_12, EMS_30);     !NOT USED                                                   ZZZ
;  6768  2                    [61]:   ERRHRD (61, EH_13, EMS_30);     !SUCCESSFUL MESSAGE                                         ZZZ
;  6769  2                    [62]:   ERRHRD (62, EH_13, EMS_30);     !ILLEGAL UNIT NUMBER                                        ZZZ
;  6770  2                    [63]:   ERRHRD (63, EH_13, EMS_30);     !ILLEGAL RELATIVE OR PHYSICAL BLOCK                         ZZZ
;  6771  2                    [64]:   ERRHRD (64, EH_13, EMS_30);     !DEVICE ERROR                                               ZZZ
;  6772  2                    [65]:   ERRHRD (65, EH_13, EMS_30);     !ZERO LENGTH MESSAGE                                        ZZZ
;  6773  2                    [66]:   ERRHRD (66, EH_8,  EMS_30);     !DUP UNKNOWN STATUS CODE                                    ZZZ
;  6774  2                    [67]:   ERRHRD (67, EH_7,  EMS_30);     !INVALID COMMAND                                            ZZZ
;  6775  2                    [68]:   ERRHRD (68, EH_7,  EMS_30);     !NO REGION AVAILABLE                                        ZZZ
;  6776  2                    [69]:   ERRHRD (69, EH_7,  EMS_30);     !NO REGION SUITABLE                                         ZZZ
;  6777  2                    [70]:   ERRHRD (70, EH_7,  EMS_30);     !PROGRAM NOT KNOWN                                          ZZZ
;  6778  2                    [71]:   ERRHRD (71, EH_7,  EMS_30);     !LOAD FAILURE                                               ZZZ
;  6779  2                    [72]:   ERRHRD (72, EH_7,  EMS_30);     !STANDALONE                                                 ZZZ
;  6780  2                    [73]:   ERRHRD (73, EH_8,  EMS_30);     !DUP UNKNOWN STATUS CODE                                    ZZZ
;  6781  2
;  6782  2                        tes
;  6783  2                else
;  6784  3                    begin
;  6785  3                    !****increment error count              ! INCREMENT TOTAL ERROR COUNT
;  6786  3                    PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC); ! PRINT ERROR MESSAGE JUST LIKE DRS
;  6787  3
;  6788  3                    if .ERRNUM neq 42
;  6789  3                    then
;  6790  4                        begin
;  6791  4                        PRINTB (HRD_SUB);                   ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;  6792  4                        EMS_ERR ();                         ! PRINT REST OF THE INFORMATION
;  6793  3                        end;
;  6794  2                    end;
```

J4


```
;  6795  2
;  6796  2          if (.ERRNUM eql 35) or                          ! FOR BAD-BLOCK TYPE ERRORS
;  6797  3             (.ERRNUM eql 38)
;  6798  2          then
;  6799  2
;  6800  3              if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF ERRORS TO BE TREATED NORMALLY
;  6801  3              then
;  6802  3
;  6803  3                  selectoneu .ERRNUM of
;  6804  3                      set
;  6805  3
;  6806  3                      [35]:   ERRHRD (35, EGH_30, EMS_30);    ! MEDIA FORMAT ERROR
;  6807  3
;  6808  3                      [38]:   ERRHRD (38, EGH_30, EMS_30);    ! DATA ERROR
;  6809  3                      tes
;  6810  2              else
;  6811  3                  begin
;  6812  3                  !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;  6813  3                  PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR LINE JUST LIKE DRS
;  6814  3                  PRINTB (HRD_SUB);                           ! PRINT NEXT LINE TOO
;  6815  3                  EMS_ERR ();                                 ! PRINT REST OF THE INFORMATION
;  6816  2                  end;
;  6817  2
;  6818  2          SETPRI (.CUR_PRIORITY);                         ! PRIORITY BACK TO NORMAL
;  6819  2
;  6820  1          end;


                                        .SBTTL  ERR.HRD.RTNE RDRX INTERRUPT SERVICE ROUTINES
032010                                  .PSECT  $CODE$,  RO

000000  004137  000000G          ERR.HRD.RTNE:
                                        JSR     R1,$SAVE2
000004  104440                         TRAP    40                      ;                       6690
000006  010002                         MOV     R0,R2                   ;                       6705
000010  013700  C00000G                 MOV     BRLEVEL,R0              ; *,CUR.PRIORITY
000014  104441                         TRAP    41                      ;                       6707
000016  016601  000010                 MOV     10(SP),R1               ; ERRNUM,*              6709
000022  020127  000042                 CMP     R1,#42
000026  101411                         BLOS    1$
000030  020127  000046                 CMP     R1,#46                  ;                       6710
000034  101006                         BHI     1$
000036  020127  000044                 CMP     R1,#44                  ;                       6711
000042  001403                         BEQ     1$
000044  020127  000045                 CMP     R1,#45                  ;                       6712
000050  001176                         BNE     27$
000052  032737  010000  000000G  1$:    BIT     #10000,SWP.FLAGS        ;                       6715
000060  001002                         BNE     2$
000062  000137  032504'                JMP     31$
000066  010100                 2$:     MOV     R1,R0                   ;                       6719
000070  162700  000037                 SUB     #37,R0
000074  006300                         ASL     R0
000076  066007  000012'                 ADD     P.AAB(R0),PC            ; Case dispatch
```

K4


```
000102  104456                  4$:     TRAP    56                              ;                       6722
000104  000037                          .WORD   37
000106  000000G                         .WORD   EGH.30
000110  000000G                         .WORD   EMS.30
000112  000567                          BR      30$
000114  104456                  5$:     TRAP    56                              ;                       6719
000116  000040                          .WORD   40                              ;                       6724
000120  000000G                         .WORD   EGH.30
000122  000000G                         .WORD   EMS.30
000124  000562                          BR      30$
000126  104456                  6$:     TRAP    56                              ;                       6719
000130  000044                          .WORD   44                              ;                       6732
000132  000000G                         .WORD   EGH.30
000134  000000G                         .WORD   EMS.30
000136  000555                          BR      30$
000140  104456                  7$:     TRAP    56                              ;                       6719
000142  000045                          .WORD   45                              ;                       6734
000144  000000G                         .WORD   EGH.30
000146  000000G                         .WORD   EMS.30
000150  000550                          BR      30$
000152  104456                  8$:     TRAP    56                              ;                       6719
000154  000047                          .WORD   47                              ;                       6738
000156  000000G                         .WORD   EGH.30
000160  000000G                         .WORD   EMS.30
000162  000574                          BR      33$
000164  104456                  9$:     TRAP    56                              ;                       6719
000166  000050                          .WORD   50                              ;                       6740
000170  000000G                         .WORD   EGH.30
000172  000000G                         .WORD   EMS.30
000174  000567                          BR      33$
000176  104456                  10$:    TRAP    56                              ;                       6719
000200  000051                          .WORD   51                              ;                       6742
000202  000000G                         .WORD   EGH.30
000204  000000G                         .WORD   EMS.30
000206  000562                          BR      33$
000210  104456                  11$:    TRAP    56                              ;                       6719
000212  000052                          .WORD   52                              ;                       6744
000214  000000G                         .WORD   EGH.30
000216  000000                          .WORD   0
000220  000555                          BR      33$
000222  104456                  12$:    TRAP    56                              ;                       6719
000224  000053                          .WORD   53                              ;                       6746
000226  000000G                         .WORD   EGH.30
000230  000000G                         .WORD   EMS.30
000232  000550                          BR      33$
000234  104456                  13$:    TRAP    56                              ;                       6719
000236  000054                          .WORD   54                              ;                       6748
000240  000000G                         .WORD   EGH.30
000242  000000G                         .WORD   EMS.30
000244  000543                          BR      33$
000246  104456                  14$:    TRAP    56                              ;                       6719
000250  000055                          .WORD   55                              ;                       6750
000252  000000G                         .WORD   EGH.30
```

L4


```
000254  000000G                         .WORD   EMS.30
000256  000536                         BR      33$
000260  104456                 15$:    TRAP    56                                      ;6719
000262  000074                         .WORD   74                                      ;6767
000264  000000G                         .WORD   EH.12
000266  000000G                         .WORD   EMS.30
000270  000531                         BR      33$
000272  104456                 16$:    TRAP    56                                      ;6719
000274  000075                         .WORD   75                                      ;6768
000276  000000G                         .WORD   EH.13
000300  000000G                         .WORD   EMS.30
000302  000524                         BR      33$
000304  104456                 17$:    TRAP    56                                      ;6719
000306  000076                         .WORD   76                                      ;6769
000310  000000G                         .WORD   EH.13
000312  000000G                         .WORD   EMS.30
000314  000517                         BR      33$
000316  104456                 18$:    TRAP    56                                      ;6719
000320  000077                         .WORD   77                                      ;6770
000322  000000G                         .WORD   EH.13
000324  000000G                         .WORD   EMS.30
000326  000512                         BR      33$
000330  104456                 19$:    TRAP    56                                      ;6719
000332  000100                         .WORD   100                                     ;6771
000334  000000G                         .WORD   EH.13
000336  000000G                         .WORD   EMS.30
000340  000505                         BR      33$
000342  104456                 20$:    TRAP    56                                      ;6719
000344  000101                         .WORD   101                                     ;6772
000346  000000G                         .WORD   EH.13
000350  000000G                         .WORD   EMS.30
000352  000500                         BR      33$
000354  104456                 21$:    TRAP    56                                      ;6719
000356  000102                         .WORD   102                                     ;6773
000360  000000G                         .WORD   EH.8
000362  000000G                         .WORD   EMS.30
000364  000473                         BR      33$
000366  104456                 22$:    TRAP    56                                      ;6719
000370  000103                         .WORD   103                                     ;6774
000372  000000G                         .WORD   EH.7
000374  000000G                         .WORD   EMS.30
000376  000466                         BR      33$
000400  104456                 23$:    TRAP    56                                      ;6719
000402  000104                         .WORD   104                                     ;6775
000404  000000G                         .WORD   EH.7
000406  000000G                         .WORD   EMS.30
000410  000461                         BR      33$
000412  104456                 24$:    TRAP    56                                      ;6719
000414  000105                         .WORD   105                                     ;6776
000416  000000G                         .WORD   EH.7
000420  000000G                         .WORD   EMS.30
000422  000454                         BR      33$
000424  104456                 25$:    TRAP    56                                      ;6719
                                                                                       ;6777
```

M4


```
000426  000106                          .WORD   106
000430  000000G                         .WORD   EH.7
000432  000000G                         .WORD   EMS.30
000434  000447                          BR      33$                             ;                       6719
000436  104456                  26$:    TRAP    56                              ;                       6778
000440  000107                          .WORD   107
000442  000000G                         .WORD   EH.7
000444  000000G                         .WORD   EMS.30
000446  000442                  27$:    BR      33$                             ;                       6719
000450  104456                  28$:    TRAP    56                              ;                       6779
000452  000110                          .WORD   110
000454  000000G                         .WORD   EH.7
000456  000000G                         .WORD   EMS.30
000460  000435                          BR      33$                             ;                       6719
000462  104456                  29$:    TRAP    56                              ;                       6780
000464  000111                          .WORD   111
000466  000000G                         .WORD   EH.8
000470  000000G                         .WORD   EMS.30
000472  000430                  30$:    BR      33$                             ;                       6715
000474  010746                  31$:    MOV     PC,-(SP)                        ; PC,*                  6786
000476  013746  000000G                 MOV     L$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                          ;                       6788
000524  001411                          BEQ     32$
000526  012716  000000G                 MOV     #HRD.SUB,(SP)                   ;                       6791
000532  012746  000001                  MOV     #1,-(SP)
000536  010600                          MOV     SP,R0                           ; SP,*
000540  104414                          TRAP    14
000542  004737  000000G                 JSR     PC,EMS.ERR                      ;                       6792
000546  005726                          TST     (SP)+                           ;                       6790
000550  062706  000012          32$:    ADD     #12,SP                          ;                       6784
000554  020127  000043          33$:    CMP     R1,#43                          ;                       6796
000560  001403                          BEQ     34$
000562  020127  000046                  CMP     R1,#46                          ;                       6797
000566  001050                          BNE     37$
000570  032737  040000  000000G 34$:    BIT     #40000,SWP.FLAGS                ;                       6800
000576  001420                          BEQ     36$
000600  020127  000043                  CMP     R1,#43                          ;                       6806
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$                             ;                       6803
000620  020127  000046          35$:    CMP     R1,#46                          ;                       6808
000624  001031                          BNE     37$
000626  104456                          TRAP    56
000630  000046                          .WORD   46
000632  000000G                         .WORD   EGH.30
```

```
000634  000000G                         .WORD   EMS.30
000636  000424                          BR      37$
000640  010746                  36$:    MOV     PC,-(SP)                ;                               6803
000642  013746  000000G                 MOV     L$LUN,-(SP)             ; PC,*                          6813
000646  010146                          MOV     R1,-(SP)
000650  012746  000000G                 MOV     #HRD.MSG,-(SP)
000654  012746  000004                  MOV     #4,-(SP)
000660  010600                          MOV     SP,R0
000662  104414                          TRAP    14                      ; SP,*
000664  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                               6814
000670  012746  000001                  MOV     #1,-(SP)
000674  010600                          MOV     SP,R0
000676  104414                          TRAP    14                      ; SP,*
000700  004737  000000G                 JSR     PC,EMS.ERR              ;                               6815
000704  062706  000014                  ADD     #14,SP                  ;                               6811
000710  010200                  37$:    MOV     R2,R0                   ; CUR.PRIORITY,*                6818
000712  104441                          TRAP    41
000714  000207                          RTS     PC                      ;                               6690

; Routine Size:  231 words,    Routine Base:  $CODE$ + 32010
; Maximum stack depth per invocation:  11 words


000012                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAB:
000012  000000                  3$:     .WORD   0                       ; CASE Table for ERR.HRD.RTNE+0076      6719
000014  000012                          .WORD   12                      ; [4$]
000016  000452                          .WORD   452                     ; [5$]
000020  000452                          .WORD   452                     ; [33$]
000022  000452                          .WORD   452                     ; [33$]
000024  000024                          .WORD   24                      ; [33$]
000026  000036                          .WORD   36                      ; [6$]
000030  000452                          .WORD   452                     ; [7$]
000032  000050                          .WORD   50                      ; [33$]
000034  000062                          .WORD   62                      ; [8$]
000036  000074                          .WORD   74                      ; [9$]
000040  000106                          .WORD   106                     ; [10$]
000042  000120                          .WORD   120                     ; [11$]
000044  000132                          .WORD   132                     ; [12$]
000046  000144                          .WORD   144                     ; [13$]
000050  000452                          .WORD   452                     ; [14$]
000052  000452                          .WORD   452                     ; [33$]
000054  000452                          .WORD   452                     ; [33$]
000056  000452                          .WORD   452                     ; [33$]
000060  000452                          .WORD   452                     ; [33$]
000062  000452                          .WORD   452                     ; [33$]
000064  000452                          .WORD   452                     ; [33$]
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  000156                          .WORD   156             ; [15$]
000106  000170                          .WORD   170             ; [16$]
000110  000202                          .WORD   202             ; [17$]
000112  000214                          .WORD   214             ; [18$]
000114  000226                          .WORD   226             ; [19$]
000116  000240                          .WORD   240             ; [20$]
000120  000252                          .WORD   252             ; [21$]
000122  000264                          .WORD   264             ; [22$]
000124  000276                          .WORD   276             ; [23$]
000126  000310                          .WORD   310             ; [24$]
000130  000322                          .WORD   322             ; [25$]
000132  000334                          .WORD   334             ; [26$]
000134  000346                          .WORD   346             ; [28$]
000136  000360                          .WORD   360             ; [29$]
```

```
;    6821  1     routine ERR_SOFT_RTNE (ERRNUM) : novalue =
;    6822  1
;    6823  1     !+
;    6824  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6825  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6826  1     !-
;    6827  1
;    6828  2         begin
;    6829  2
;    6830  2         builtin
;    6831  2             PC;
;    6832  2
;    6833  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                 ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    6834  2         then
;    6835  2
;    6836  2             case .ERRNUM from 50 to 54 of
;    6837  2                 set
;    6838  2
;    6839  2                 [50]:       ERRSOFT (50, 0, 0);         ! CONTROLLER ERROR
;    6840  2
;    6841  2                 [51]:       ERRSOFT (51, 0, 0);         ! HOST MEMORY ACCESS ERROR
;    6842  2
;    6843  2                 [52]:       ERRSOFT (52, 0, 0);         ! DISK TRANSFER ERROR
;    6844  2
;    6845  2                 [53]:       ERRSOFT (53, 0, 0);         ! SDI ERROR
;    6846  2
;    6847  2                 [54]:       ERRSOFT (54, 0, 0);         ! SMALL DISK ERROR
;    6848  2                 tes
;    6849  2         else
;    6850  3             begin
;    6851  3             !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;    6852  3             PRINTB (SFT_MSG, .ERRNUM, .L$LUN, .PC);     ! PRINT ERROR LINE JUST LIKE DRS
;    6853  2             end;
;    6854  2
;    6855  1         end;


                                        .SBTTL  ERR.SOFT.RTNE RDRX INTERRUPT SERVICE ROUTINES
032726                                  .PSECT  $CODE$,  RO

000000  032737  020000  000000G ERR.SOFT.RTNE:
                                BIT     #20000,SWP.FLAGS                ;                       6833
000006  001440                  BEQ     7$
000010  016600  000002          MOV     2(SP),RO                        ; ERRNUM,*              6836
000014  162700  000062          SUB     #62,RO
000020  006300                  ASL     RO
000022  066007  000140'          ADD     P.AAC(RO),PC                    ; Case dispatch
000026  104457              2$: TRAP    57                              ;                       6839
000030  000062                  .WORD   62
000032  000000                  .WORD   0
000034  000000                  .WORD   0
000036  000207                  RTS     PC                              ;                       6836
000040  104457              3$: TRAP    57                              ;                       6841
```

```
000042  000063                          .WORD   63
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  000207                          RTS     PC                              ;                       6836
000052  104457                  4$:     TRAP    57                              ;                       6843
000054  000064                          .WORD   64
000056  000000                          .WORD   0
000060  000000                          .WORD   0
000062  000207                          RTS     PC                              ;                       6836
000064  104457                  5$:     TRAP    57                              ;                       6845
000066  000065                          .WORD   65
000070  000000                          .WORD   0
000072  000000                          .WORD   0
000074  000207                          RTS     PC                              ;                       6836
000076  104457                  6$:     TRAP    57                              ;                       6847
000100  000066                          .WORD   66
000102  000000                          .WORD   0
000104  000000                          .WORD   0
000106  000207                          RTS     PC                              ;                       6833
000110  010746                  7$:     MOV     PC,-(SP)                        ; PC,*                  6852
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                          ;                       6850
000142  000207                          RTS     PC                              ;                       6821

; Routine Size:  50 words,      Routine Base:  $CODE$ + 32726
; Maximum stack depth per invocation:  7 words


000140                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAC:
000140  000000                 1$:     .WORD   0                               ; CASE Table for ERR.SOFT.RTNE+0022     6836
000142  000012                         .WORD   12                              ; [2$]
000144  000024                         .WORD   24                              ; [3$]
000146  000036                         .WORD   36                              ; [4$]
000150  000050                         .WORD   50                              ; [5$]
                                                                                ; [6$]
```

```
;    6856  1     routine ERR_HRD_RTNE_APT (ERRNUM) : novalue =
;    6857  1
;    6858  1
;    6859  1     !+
;    6860  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    6861  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6862  1     !-
;    6863  1
;    6864  1
;    6865  2         begin
;    6866  2
;    6867  2
;    6868  2         local
;    6869  2             CUR_PRIORITY;
;    6870  2
;    6871  2
;    6872  2         builtin
;    6873  2             PC;
;    6874  2
;    6875  2         GETPRI (CUR_PRIORITY);
;    6876  2     !ZZZ    SETPRI (PRI04);                          ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    6877  2         SETPRI (.BRLEVEL);                           ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW     ZZZ
;    6878  2
;    6879  2
;    6880  2         if .APT_MODE
;    6881  2         then
;    6882  2
;    6883  3             begin
;    6884  3             .MAIL_BOX_TESTNUM = .RP_ADDR [LBN_LO];   ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    6885  3             .MAIL_BOX_SUBTST = .RP_ADDR [DISK];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER UNDER APT ONLY
;    6886  2             end;
;    6887  2
;    6888  2
;    6889  2         if (.ERRNUM lequ 34) or                      ! FOR NON-BAD BLOCK TYPE ERRORS
;    6890  2            (.ERRNUM gtru 38) or
;    6891  2            (.ERRNUM eql 36) or
;    6892  3            (.ERRNUM eql 37)
;    6893  3
;    6894  2         then
;    6895  2
;    6896  3             if BIT_TST (SWP_FLAGS, SWF_HRD)          ! IF ERRORS TO BE TREATED NORMALLY
;    6897  2             then
;    6898  2
;    6899  2                 case .ERRNUM from 31 to 45 of
;    6900  2                     set
;    6901  2
;    6902  2                     [31]:   ERRDF (31, EGH_30, EMS_30);     ! INVALID COMMAND
;    6903  2
;    6904  2                     [32]:   ERRDF (32, EGH_30, EMS_30);     ! COMMAND ABORTED
;    6905  2
;    6906  2                     [33]:   ;                               !
;    6907  2
;    6908  2                     [34]:   ;                               !
```

```
;  6909  2
;  6910  2                    [35]:   ;                               ! MEDIA FORMAT ERROR
;  6911  2
;  6912  2                    [36]:   ERRDF (36, EGH_30, EMS_30);     ! WRITE PROTECTED
;  6913  2
;  6914  2                    [37]:   ERRDF (37, EGH_30, EMS_30);     ! COMPARE ERROR
;  6915  2
;  6916  2                    [38]:   ;                               ! DATA ERROR
;  6917  2
;  6918  2                    [39]:   ERRDF (39, EGH_30, EMS_30);     ! HOST BUFFER ACCESS ERROR
;  6919  2
;  6920  2                    [40]:   ERRDF (40, EGH_30, EMS_30);     ! CONTROLLER ERROR
;  6921  2
;  6922  2                    [41]:   ERRDF (41, EGH_30, EMS_30);     ! DRIVE ERROR
;  6923  2
;  6924  2                    [42]:   ERRDF (42, EGH_30, 0);          ! HOST WRITE COMPARE ERROR
;  6925  2
;  6926  2                    [43]:   ERRDF (43, EGH_30, EMS_30);     ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  6927  2
;  6928  2                    [44]:   ERRDF (44, EGH_30, EMS_30);     ! DUPLICATE UNIT NUMBER
;  6929  2
;  6930  2                    [45]:   ERRDF (45, EGH_30, EMS_30);     ! INVALID END CODE
;  6931  2                    tes
;  6932  2
;  6933  2                else
;  6934  2
;  6935  3                    begin
;  6936  3                    !****increment error count              ! INCREMENT TOTAL ERROR COUNT
;  6937  3                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);  ! PRINT ERROR MESSAGE JUST LIKE DRS
;  6938  3
;  6939  3
;  6940  3                    if .ERRNUM neq 42
;  6941  3
;  6942  3                    then
;  6943  4                        begin
;  6944  4                        PRINTB (HRD_SUB);                   ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;  6945  4                        EMS_ERR ();                         ! PRINT REST OF THE INFORMATION
;  6946  3                        end;
;  6947  2                    end;
;  6948  2
;  6949  2                if (.ERRNUM eql 35) or                      ! FOR BAD-BLOCK TYPE ERRORS
;  6950  3                   (.ERRNUM eql 38)
;  6951  3
;  6952  2                then
;  6953  2
;  6954  3                    if BIT_TST (SWP_FLAGS, SWF_BLK)         ! IF ERRORS TO BE TREATED NORMALLY
;  6955  2                    then
;  6956  2
;  6957  2                        selectoneu .ERRNUM of
;  6958  2                            set
;  6959  2
;  6960  2                            [35]:   ERRDF (35, EGH_30, EMS_30);     ! MEDIA FORMAT ERROR
;  6961  2
```

```
;  6962  2                    [38]:   ERRDF (38, EGH_30, EMS_30);     ! DATA ERROR
;  6963  2                    tes
;  6964  2
;  6965  2                else
;  6966  2
;  6967  3                    begin
;  6968  3                    !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;  6969  3                    PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR LINE JUST LIKE DRS
;  6970  3                    PRINTB (HRD_SUB);                           ! PRINT NEXT LINE TOO
;  6971  3                    EMS_ERR ();                                 ! PRINT REST OF THE INFORMATION
;  6972  2                    end;
;  6973  2
;  6974  2
;  6975  2            SETPRI (.CUR_PRIORITY);                             ! PRIORITY BACK TO NORMAL
;  6976  2
;  6977  2
;  6978  1            end,


033072                                      .SBTTL  ERR.HRD.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
                                            .PSECT  $CODE$,  RO

000000  004137  000000G                ERR.HRD.RTNE.APT:
                                            JSR     R1,$SAVE2                       ;                       6856
000004  104440                              TRAP    40                              ;                       6875
000006  010002                              MOV     R0,R2                           ; *,CUR.PRIORITY
000010  013700  000000G                     MOV     BRLEVEL,R0                      ;                       6877
000014  104441                              TRAP    41
000016  032737  000001  001256'             BIT     #1,APT.MODE                     ;                       6880
000024  001412                              BEQ     1$
000026  013700  000000G                     MOV     RP.ADDR,R0                      ;                       6884
000032  016077  000050  001260'             MOV     50(R0),@MAIL.BOX.TESTNUM
000040  013700  000000G                     MOV     RP.ADDR,R0                      ;                       6885
000044  016077  000010  001262'             MOV     10(R0),@MAIL.BOX.SUBTST
000052  016601  000010              1$:     MOV     10(SP),R1                       ; ERRNUM,*              6889
000056  020127  000042                      CMP     R1,#42
000062  101411                              BLOS    2$
000064  020127  000046                      CMP     R1,#46                          ;                       6890
000070  101006                              BHI     2$
000072  020127  000044                      CMP     R1,#44                          ;                       6891
000076  001403                              BEQ     2$
000100  020127  000045                      CMP     R1,#45                          ;                       6892
000104  001131                              BNE     17$
000106  032737  010000  000000G     2$:     BIT     #10000,SWP.FLAGS                ;                       6896
000114  001475                              BEQ     15$
000116  010100                              MOV     R1,R0                           ;                       6899
000120  162700  000037                      SUB     #37,R0
000124  006300                              ASL     R0
000126  066007  000152'                      ADD     P.AAD(R0),PC                    ; Case dispatch
000132  104455                      4$:     TRAP    55                              ;                       6902
000134  000037                              .WORD   37
000136  000000G                             .WORD   EGH.30
000140  000000G                             .WORD   EMS.30
```

H5


```
000142  000512                          BR      17$                             ;                       6899
000144  104455                  5$:     TRAP    55                              ;                       6904
000146  000040                          .WORD   40
000150  000000G                         .WORD   EGH.30
000152  000000G                         .WORD   EMS.30
000154  000505                          BR      17$
000156  104455                  6$:     TRAP    55                              ;                       6899
000160  000044                          .WORD   44                              ;                       6912
000162  000000G                         .WORD   EGH.30
000164  000000G                         .WORD   EMS.30
000166  000500                          BR      17$
000170  104455                  7$:     TRAP    55                              ;                       6899
000172  000045                          .WORD   45                              ;                       6914
000174  000000G                         .WORD   EGH.30
000176  000000G                         .WORD   EMS.30
000200  000473                          BR      17$
000202  104455                  8$:     TRAP    55                              ;                       6899
000204  000047                          .WORD   47                              ;                       6918
000206  000000G                         .WORD   EGH.30
000210  000000G                         .WORD   EMS.30
000212  000466                          BR      17$
000214  104455                  9$:     TRAP    55                              ;                       6899
000216  000050                          .WORD   50                              ;                       6920
000220  000000G                         .WORD   EGH.30
000222  000000G                         .WORD   EMS.30
000224  000461                          BR      17$
000226  104455                  10$:    TRAP    55                              ;                       6899
000230  000051                          .WORD   51                              ;                       6922
000232  000000G                         .WORD   EGH.30
000234  000000G                         .WORD   EMS.30
000236  000454                          BR      17$
000240  104455                  11$:    TRAP    55                              ;                       6899
000242  000052                          .WORD   52                              ;                       6924
000244  000000G                         .WORD   EGH.30
000246  000000                          .WORD   0
000250  000447                          BR      17$
000252  104455                  12$:    TRAP    55                              ;                       6899
000254  000053                          .WORD   53                              ;                       6926
000256  000000G                         .WORD   EGH.30
000260  000000G                         .WORD   EMS.30
000262  000442                          BR      17$
000264  104455                  13$:    TRAP    55                              ;                       6899
000266  000054                          .WORD   54                              ;                       6928
000270  000000G                         .WORD   EGH.30
000272  000000G                         .WORD   EMS.30
000274  000435                          BR      17$
000276  104455                  14$:    TRAP    55                              ;                       6899
000300  000055                          .WORD   55                              ;                       6930
000302  000000G                         .WORD   EGH.30
000304  000000G                         .WORD   EMS.30
000306  000430                          BR      17$                             ;                       6896
000310  010746                  15$:    MOV     PC,-(SP)                        ; PC,*                  6937
000312  013746  000000G                 MOV     L$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                  ;                       6940
000340  001411                          BEQ     16$
000342  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                       6944
000346  012746  000001                  MOV     #1,-(SP)
000352  010600                          MOV     SP,R0                   ; SP,*
000354  104414                          TRAP    14
000356  004737  000000G                 JSR     PC,EMS.ERR              ;                       6945
000362  005726                          TST     (SP)+                   ;                       6943
000364  062706  000012          16$:    ADD     #12,SP                  ;                       6935
000370  020127  000043          17$:    CMP     R1,#43                  ;                       6949
000374  001403                          BEQ     18$
000376  020127  000046                  CMP     R1,#46                  ;                       6950
000402  001050                          BNE     21$
000404  032737  040000  000000G 18$:    BIT     #40000,SWP.FLAGS        ;                       6954
000412  001420                          BEQ     20$
000414  020127  000043                  CMP     R1,#43                  ;                       69
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$
000434  020127  000046          19$:    CMP     R1,#46                  ;                       6957
000440  001031                          BNE     21$                     ;                       6962
000442  104455                          TRAP    55
000444  000046                          .WORD   46
000446  000000G                         .WORD   EGH.30
000450  000000G                         .WORD   EMS.30
000452  000424                          BR      21$
000454  010746                  20$:    MOV     PC,-(SP)                ;                       6957
000456  013746  000000G                 MOV     L$LUN,-(SP)             ; PC,*                  6969
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)           ;                       6970
000504  012746  000001                  MOV     #1,-(SP)
000510  010600                          MOV     SP,R0                   ; SP,*
000512  104414                          TRAP    14
000514  004737  000000'                 JSR     PC,EMS.ERR              ;                       6971
000520  062706  00001^                  ADD     #14,SP                  ;                       6967
000524  010200                  21$:    MOV     R2,R0                   ; CUR.PRIORITY,*        6975
000526  104441                          TRAP    41
000530  000207                          RTS     PC                      ;                       6856

; Routine Size:  173 words,    Routine Base:  $CODE$ + 33072
; Maximum stack depth per invocation:  11 words
```

```
000152                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAD:
000152  000000                3$:     .WORD   0               ; CASE Table for ERR.HRD.RTNE.AP+0126   6899
000154  000012                        .WORD   12              ; [4$]
000156  000236                        .WORD   236             ; [5$]
000160  000236                        .WORD   236             ; [17$]
000162  000236                        .WORD   236             ; [17$]
000164  000024                        .WORD   24              ; [17$]
000166  000036                        .WORD   36              ; [6$]
000170  000236                        .WORD   236             ; [7$]
000172  000050                        .WORD   50              ; [17$]
000174  000062                        .WORD   62              ; [8$]
000176  000074                        .WORD   74              ; [9$]
000200  000106                        .WORD   106             ; [10$]
000202  000120                        .WORD   120             ; [11$]
000204  000132                        .WORD   132             ; [12$]
000206  000144                        .WORD   144             ; [13$]
                                                               ; [14$]


;   6979  1
;   6980  1
```

K5


```
;    6981  1     routine ERR_SOFT_RTNE_APT (EPRNUM, index) : novalue =
;    6982  1
;    6983  1     !+
;    6984  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6985  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6986  1     !-
;    6987  1
;    6988  2         begin
;    6989  2
;    6990  2         local
;    6991  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;    6992  2
;    6993  2         builtin
;    6994  2             PC;
;    6995  2
;    6996  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);       ! ADDRESS OF THE SAVED ERROR-LOG INFORMATION
;    6997  2
;    6998  2         if .APT_MODE
;    6999  2         then
;    7000  3             begin
;    7001  3             .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];      ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    7002  3             .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
;    7003  2             end;
;    7004  2
;    7005  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                     ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    7006  2         then
;    7007  2
;    7008  2             case .ERRNUM from 50 to 54 of
;    7009  2                 set
;    7010  2
;    7011  2                 [50]:       ERRDF (50, 0, 0);               ! CONTROLLER ERROR
;    7012  2
;    7013  2                 [51]:       ERRDF (51, 0, 0);               ! HOST MEMORY ACCESS ERROR
;    7014  2
;    7015  2                 [52]:       ERRDF (52, 0, 0);               ! DISK TRANSFER ERROR
;    7016  2
;    7017  2                 [53]:       ERRDF (53, 0, 0);               ! SDI ERROR
;    7018  2
;    7019  2                 [54]:       ERRDF (54, 0, 0);               ! SMALL DISK ERROR
;    7020  2                 tes
;    7021  2             else
;    7022  3                 begin
;    7023  3                 !****increment error count                  ! INCREMENT TOTAL ERROR COUNT
;    7024  3                 PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);      ! PRINT ERROR LINE JUST LIKE DRS
;    7025  2                 end;
;    7026  2
;    7027  1         end;


                                                .SBTTL  ERR.SOFT.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
033624                                          .PSECT  $CODE$,  RO

000000  016646  000002                     ERR.SOFT.RTNE.APT:
```

L5


```
000004  012746  000102                        MOV     2(SP),-(SP)             ; INDEX,*                       6996
000010  004737  000000G                       MOV     #102,-(SP)
000014  062700  000000G                       JSR     PC,BL$MUL
000020  032737  000001  001256'               ADD     #ELOG.PKT,R0
000026  001406                                BIT     #1,APT.MODE             ;                               6998
000030  016077  000056  001260'               BEQ     1$
000036  016077  000012  001262'               MOV     56(R0),@MAIL.BOX.TESTNUM ; *(ELOG.ADDR),*               7001
000044  032737  020000  000000G       1$:     MOV     12(R0),@MAIL.BOX.SUBTST ; *(ELOG.ADDR),*                7002
000052  001440                                BIT     #20000,SWP.FLAGS        ;                               7005
000054  016600  000010                        BEQ     8$
000060  162700  000062                        MOV     10(SP),R0               ; ERRNUM,*                      7008
000064  006300                                SUB     #62,R0
000066  066007  000210'                        ASL     R0
000072  104455                        3$:     ADD     P.AAE(R0),PC            ; Case dispatch
000074  000062                                TRAP    55                      ;                               7011
000076  000000                                .WORD   62
000100  000000                                .WORD   0
000102  000441                                .WORD   0
000104  104455                        4$:     BR      9$                      ;                               7008
000106  000063                                TRAP    55                      ;                               7013
000110  000000                                .WORD   63
000112  000000                                .WORD   0
000114  000434                                .WORD   0
000116  104455                        5$:     BR      9$                      ;                               7008
000120  000064                                TRAP    55                      ;                               7015
000122  000000                                .WORD   64
000124  000000                                .WORD   0
000126  000427                                .WORD   0
000130  104455                        6$:     BR      9$                      ;                               7008
000132  000065                                TRAP    55                      ;                               7017
000134  000000                                .WORD   65
000136  000000                                .WORD   0
000140  000422                                .WORD   0
000142  104455                        7$:     BR      9$                      ;                               7008
000144  000066                                TRAP    55                      ;                               7019
000146  000000                                .WORD   66
000150  000000                                .WORD   0
000152  000415                                .WORD   0
000154  010716                        8$:     BR      9$                      ;                               7005
000156  013746  000000G                       MOV     PC,(SP)                 ; PC,*                          7024
000162  016646  000012                        MOV     L$LUN,-(SP)
000166  012746  000000G                       MOV     12(SP),-(SP)            ; ERRNUM,*
000172  012746  000004                        MOV     #DF.MSG,-(SP)
000176  010600                                MOV     #4,-(SP)
000200  104414                                MOV     SP,R0                   ; SP,*
000202  062706  000010                        TRAP    14
000206  022626                        9$:     ADD     #10,SP                  ;                               7022
000210  000207                                CMP     (SP)+,(SP)+             ;                               6988
                                              RTS     PC                      ;                               6981

; Routine Size:  69 words,      Routine Base:  $CODE$ + 33624
; Maximum stack depth per invocation:  8 words
```

M5


```
000210                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAE:                                  ; CASE Table for ERR.SOFT.RTNE.A+0066   7008
000210  000000                  2$:     .WORD   0                       ; [3$]
000212  000012                          .WORD   12                      ; [4$]
000214  000024                          .WORD   24                      ; [5$]
000216  000036                          .WORD   36                      ; [6$]
000220  000050                          .WORD   50                      ; [7$]


;   7028  1
;   7029  1
;   7030  1     end
;   7031  1
;   7032  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
;        $GGG$                    356       RO ,  I  ,  LCL,  REL,  CON
;        $CODE$                  7183       RO ,  I  ,  LCL,  REL,  CON
;        $PLIT$                    73       RO ,  D  ,  LCL,  REL,  CON


;                           Library Statistics
;
;                                       -------- Symbols --------      Pages       Processing
;       File                            Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER:[DUNCAN.RELEASE]ZRQDA0.L16;1
;                                        412       337        81        21          00:00.1



;                               COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQDA0.BL2/LIST=ZRQDA0.LS2/OBJECT=ZRQDA0.OB2/SOURCE=PAGE:53
```

```
;    7033  0     module ZRQDM4 (
;    7034  0
;    7035  0     %title 'RD/RX EXERCISER'
;    7036  0                 ident = 'V01.9',
;    7037  0                 addressing_mode (absolute),
;    7038  0                 environment (noeis)
;    7039  0                 ) =
;    7040  0
;    7041  1     begin
;    7042  1
;    7043  1     %sbttl 'LASTAD AND SETUP'
;    7044  1
;    7045  1     library 'ZRQDA0.L16';
;    7046  1
;    7047  1     !MMM require 'BLSMAC.REQ';                  ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    7048  1     require 'HSAXA0.BLB';                       ! DIAGNOSTIC SUPERVISOR LIBRARY      ZZZ
;    8789  1
;    8790  2     LASTAD
;    8791  2
;    8792  2     BGNSETUP (4)                                                                     !ZZZ
;    8793  2
;P   8794  2             BGNPTAB
;P   8795  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000020', 0, 0, RD52_MAX_LBN, 0  !ZZZ
;P   8796  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;P   8797  2
;    8798  2             ENDPTAB
;    8799  2
;P   8800  2             BGNPTAB
;P   8801  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000001', 0, 0, RX50_MAX_LBN, 0  !ZZZ
;P   8802  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    8803  2             ENDPTAB
;    8804  2
;P   8805  2             BGNPTAB
;P   8806  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000002', 0, 0, RX50_MAX_LBN, 0  !ZZZ
;P   8807  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    8808  2             ENDPTAB
;P   8809  2             BGNPTAB
;P   8810  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'000003', 0, 0, RX50_MAX_LBN, 0  !ZZZ
;P   8811  2                             !HERE'S ONE FOR THE 4TH DRIVE                                            !ZZZ
;    8812  2             ENDPTAB                                                                                  !ZZZ
;    8813  2
;    8814  1     ENDSETUP


                                        .TITLE  ZRQDM4 RD/RX EXERCISER
                                        .IDENT  /V01.9/
                                        .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  000020                  .WORD   20
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
```

C6


```
                                .SBTTL  $END.LINK LASTAD AND SETUP
000000  000207          $END.LINK::
                                RTS     PC                                      ;                                       8788

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0126
; Maximum stack depth per invocation:  0 words

;   8815  1     end
;   8816  1
;   8817  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$USER:[DUNCAN.RELEASE]ZRQDA0.L16;1
;                                     412         7         1          21          00:00.1



;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQDA0.BL2/LIST=ZRQDA0.LS2/OBJECT=ZRQDA0.OB2/SOURCE=PAGE:53

; Size:          7184 code + 472 data words
; Run Time:        02:28.7
; Elapsed Time:    03:35.4
; Lines/CPU Min:    3557
; Lexemes/CPU-Min: 31063
; Memory Used:  563 pages
; Compilation Complete
```

D6


```
;  0001  0     !*****************************************************************************
;  0002  0     !                                                                            *
;  0003  0     !                       L I T E R A L S                                      *
;  0004  0     !                                                                            *
;  0005  0     !*****************************************************************************
;  0006  0     !
;  0007  0     LITERAL
;  0008  0     !
;  0009  0     !****** ODT TRAP VECTOR LOCATION
;  0010  0     !
;  0011  0             O_TVEC          = %o'14',
;  0012  0     !
;  0013  0     !****** HARDWARE ADDRESSES ETC.
;  0014  0     !
;  0015  0             INIT_INTR_VECT  = %o'154',                          ! VECTOR ADDRESS
;  0016  0             INIT_IP_ADDR    = %o'172150',                       ! IP REGISTER ADDRESS
;  0017  0             INIT_BR_LEVEL   = %o'4',                            ! BUS REQUEST LEVEL
;  0018  0     !
;  0019  0             LINE_CLOCK      = %o'177546',                       ! LINE-CLOCK ADDRESS
;  0020  0     !
;  0021  0     !****** HARDWARE LIMITS
;  0022  0     !
;  0023  0             MAX_CTLR        = 1,                                ! MAXIMUM NUMBER OF LCP CONTROLLERS ALLOWED
;  0024  0             UNITS_PER_CNTR  = 4,                                ! MAXIMUM UNITS PER CONTROLLER
;  0025  0             MAX_UNITS       = MAX_CTLR * UNITS_PER_CNTR,        ! MAXIMUM NUMBER OF UNITS TO TEST
;  0026  0
;  0027  0     !                                                           !HIGHEST RD51 LBN = 52137 OCT
;  0028  0             RD51_MAX_TRACK  = 1200,                             ! MAXIMUM NUMBER OF TRACKS FOR RD51
;  0029  0             RD51_SEC_PER_TRK= 18,                               ! NUMBER OF SECTORS PER TRACK FOR RD51
;  0030  0             RD51_MAX_LBN    = RD51_MAX_TRACK * RD51_SEC_PER_TRK - 1,! MAX LBN FOR RD51
;  0031  0
;  0032  0             RD52_MAX_TRACK  = 2976,                             ! MAXIMUM NUMBER OF TRACKS FOR RD52
;  0033  0             RD52_SEC_PER_TRK= 18,                               ! NUMBER OF SECTORS PER TRACK FOR RD52
;  0034  0             RD52_MAX_LBN    = RD52_MAX_TRACK * RD52_SEC_PER_TRK - 1,! MAX LBN FOR RD52
;  0035  0
;  0036  0             RX50_MAX_TRACK  = 80,                               ! MAXIMUM NUMBER OF TRACKS FOR RX50
;  0037  0             RX50_SEC_PER_TRK= 10,                               ! NUMBER OF SECTORS PER TRACK FOR RX50
;  0038  0             RX50_MAX_LBN    = RX50_MAX_TRACK * RX50_SEC_PER_TRK - 1,! MAX LBN FOR RX50
;  0039  0
;  0040  0     !                                                           !HIGHEST RD53 LBN = 416660 OCT (2/016660)
;  0041  0     !                                                           !RD51  SEAGATE 4 HEADS
;  0042  0     !                                                           !RD52A QUANTUM 8 HEADS 98%
;  0043  0     !                                                           !RD52B ATASI   7 HEADS
;  0044  0     !                                                           !RD53  MICROPOLIS 8 HEADS
;  0045  0     !                                                           !RQDX2 18 BKS/TK
;  0046  0     !                                                           !RQDX2 17 BKS/TK
;  0047  0
;  0048  0             BYTES_PER_SECT  = 512,                              ! BYTES/SECTOR (AT PRESENT SAME FOR RDs AND RXs)
;  0049  0     !       MAX_XFER_SIZE   = 2 * BYTES_PER_SECT,               ! ARTBITRARY MAX SIZE OF EACH DISK I/O
;  0050  0             MAX_XFER_SIZE   = BYTES_PER_SECT * 3 / 2,
;  0051  0     !
;  0052  0     ! NOTE - BOTH OF THESE NUMBERS ARE NOW ARBITRARILY CHOSEN AS THE NUMBER OF LBNS CONTAINED PER UNIT/10 .
;  0053  0
```

```
;    0054  0     !
;    0055  0     !****** RING SIZES
;    0056  0     !
;    0057  0             CR_LOG          = 2,                                    ! LOG2 LENGTH OF COMMAND RING
;    0058  0             RR_LOG          = 2,                                    ! LOG2 LENGTH OF RESPONSE RING
;    0059  0             CRING_LEN       = 1 ^ CR_LOG,                           ! COMMAND RING LENGTH
;    0060  0             RRING_LEN       = 1 ^ RR_LOG,                           ! RESPONSE RING LENGTH
;    0061  0     !
;    0062  0     !****** OFFSETS (IN WORDS)
;    0063  0     !
;    0064  0             OF_UN           = 3,                                    ! OFFSET FROM START OF CST TO FIRST UNIT
;    0065  0             OF_DATA         = 0,                                    ! OFFSET TO DISK UNIT FLAGS   WITHIN UNIT'S CST
;    0066  0             OF_BEG          = 1,                                    ! OFFSET TO BEGINNING BLK NO. WITHIN UNIT'S CST
;    0067  0             OF_BEG1         = 2,        !OFFSET TO START BK HI      ZZZ
;    0068  0             OF_END          = 3,        !OFFSET TO END BLOCK LO     ZZZ
;    0069  0             OF_END1         = 4,        !OFFSET TO END BK HI        ZZZ
;    0070  0             OF_NAME_0       = 5,        !OFFSET TO 1st 2 CHARS OF NAME  ZZZ
;    0071  0             OF_NAME_2       = 6,        !OFFSET TO 2nd 2 CHARS OF NAME  ZZZ
;    0072  0             OF_DUPFLAGS     = 8,        !OFFSET TO DUP FLAGS        ZZZ
;    0073  0             OF_COUNT        = 9,        !OFFSET TO MSCP FUNCTION COUNTER ZZZ
;    0074  0             OF_DBN          = 8,        !OFFSET TO RELATIVE DBN     ZZZ
;    0075  0     !
;    0076  0     !
;    0077  0     !****** TABLE AND OTHER STRUCTURE SIZES
;    0078  0     !
;ZZZ0079  0             LBNADR_LEN      = 2,                                    !MAX LBN'S ARE 2 WD ADDRESSES
;ZZZ0080  0             HWPT_LEN        = 8,                                    ! SIZE (WORDS) OF HW P-TABLE
;    0081  0             COMM_LEN        = (RRING_LEN * 2) + (CRING_LEN * 2) + 4,! SIZE (WORDS) OF COMMUNICATION AREA PER CONTROLLER
;ZZZ0082  0             UNIT_SIZE       = 10,                                   ! SIZE (WORDS) OF CST UNIT ENTRY
;    0083  0             CST_LEN         = UNITS_PER_CNTR * UNIT_SIZE + OF_UN,   ! SIZE (WORDS) OF A CONTROLLER STATUS TABLE
;    0084  0             TALLY_CLEAR     = 7,                                    ! SIZE (WORDS) OF STATISTICS TBL CLEARED EVERY PASS
;ZZZ0085  0             TALLY_TOTALS    = 20,                                   ! SIZE (WORDS) OF STATISTICS TABLE FOR TOTALS
;    0086  0             TALLY_LEN       = TALLY_CLEAR + TALLY_TOTALS,           ! SIZE (WORDS) OF A STATISTICS TABLE
;    0087  0             C_ERR_LEN       = 1,                                    ! SIZE (WORDS) OF CONTROLLER ERROR TABLE
;    0088  0             RP_LEN          = 22,                                   ! SIZE (WORDS) OF A RETURN PACKET
;    0089  0             MSG_LEN         = 30,                                   ! SIZE (WORDS) OF AN MSCP MESSAGE (TEXT PORTION)
;    0090  0             PKT_LEN         = MSG_LEN + 5,                          ! SIZE (WORDS) OF AN MSCP PACKET
;    0091  0             DCT_LEN         = 9,                                    ! SIZE (WORDS) OF A DRIVER CONTROLLER TABLE
;    0092  0             RDM_LEN         = 16,                                   ! SIZE (WORDS) OF THE RANDOM NUMBER TABLE
;    0093  0             MAX_UDP_CNT     = 16,                                   ! MAX SIZE OF USER DATA PATTERN
;    0094  0             MAX_BUF_CNT     = (CRING_LEN * 2) * MAX_CTLR,           ! MAX NO. OF I/O BUFFERS (BUFF_ADDR & BUFF_OWN)
;    0095  0             PKT_CNT         = ((CRING_LEN * 2) + RRING_LEN) * MAX_CTLR,
;    0096  0
;    0097  0             RP_CNT          = PKT_CNT - (RRING_LEN * MAX_CTLR),     ! NO. OF MSCP PACKETS IN POOL
;    0098  0             IODQ_LEN        = RP_CNT,                               ! NO. OF RETURN PACKETS IN POOL
;    0099  0             OUTC_CNT        = CRING_LEN * 2,                        ! NO. OF ENTRIES IN I/O DONE QUEUE (IODQ)
;    0100  0             DP_CNT          = 21,                                   ! NO. OF ENTRIES/CONTROLLER'S OUTSTANDING CMD LIST
;    0101  0             EP_CNT          = MAX_CTLR * RRING_LEN * 3,             ! NO. OF PRE-DEFINED DATA PATTERNS
;    0102  0             EP_LEN          = PKT_LEN - 3 + 1,                      ! NO. OF ERROR-LOG PACKET SAVE BUFFERS
;    0103  0             LAST_PKT_LEN    = 3,                                    ! LENGTH OF EACH ERROR-LOG SAVE BUFFER
;    0104  0             DESC_SIZ        = 4,                                    ! BUFFER LENGTH TO SAVE INFO. ABOUT LAST RESPONSE
;    0105  0     !                                                               !NO. OF BYTES IN A PACKET DESCRIPTOR          ZZZ
;    0106  0     !****** SW P-TABLE FLAGS (SWP_FLAGS)
```

F6


```
0     !
0     !ZZZ    SWF_TRC        = %o'000001',                    ! DIAGNOSTIC TRACE
0             SWF_APT        = %o'000001',                    !RUNNING UNDER A.P.T. MONITOR                    ZZZ
0             SWF_RDM        = %o'000002',                    ! RANDOM SEEK MODE
0             SWF_CRC        = %o'000004',                    ! READ-COMPARE AT CONTROLLER
0     !       SWF_DCC        = %o'000010',                    ! DRIVE COMPLEMENT COMPLETE
0             SWF_CWC        = %o'000020',                    ! WRITE-COMPARE AT CONTROLLER
0             SWF_HWC        = %o'000040',                    ! WRITE-COMPARE AT HOST
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             MAX_DBN        = 63,                            !HIGHEST RELATIVE DBN NUMBER                     ZZZ
```

G6


```
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  (THESE FLAGS AREN'T USED. INSTEAD,)   ZZZ
0          RD_51            = 1,                        !D_TYPE FLAG = 1 FOR RD51  (D_TYPE = 1 FOR FIXED, 0 FOR REMOV)   ZZZ
0          RD_52            = 2,                        !              2 FOR RD52                                         ZZZ
0
0    !
0    !****** MSCP PACKET DESCRIPTOR
0    !
0          ED_OWN           = %o'100000',                         ! OWNERSHIP BIT
0          ED_FLAG          = %o'040000',                         ! FLAG BIT
0    !
0    !****** MSCP COMMAND PACKET OPCODES
0    !
0          OP_MSK           = %o'177',                            ! OPCODE MASK
0          OP_END           = %o'200',                            ! ENDCODE DESIGNATOR
0          OP_ACC           = %o'20',                             ! ACCESS COMMAND
0          OP_ONL           = %o'11',                             ! ONLINE COMMAND
0          OP_RD            = %o'41',                             ! READ COMMAND
0          OP_SCC           = %o'4',                              ! SET CONTROLLER CHARACTERISTICS COMMAND
0          OP_WRT           = %o'42',                             ! WRITE COMMAND
0          OP_GDS           = %o'1',                    !get dust status          ZZZ
0          OP_ESP           = %o'2',                    !execute supplied prog    ZZZ
0          OP_ELP           = %o'3',                    !execute local program    ZZZ
0          OP_SDD           = %o'4',                    !send data                ZZZ
0          OP_RCD           = %o'5',                    !receive data             ZZZ
0          OP_ABT           = %o'6',                    !abort program            ZZZ
0
0    !
0    !****** PACKET SIZES
0    !
0          SZ_ACC = %decimal '32',                                ! ACCESS
0          SZ_ONL = %decimal '36',                                ! ON LINE COMMAND
0          SZ_RD  = %decimal '32',                                ! READ
0          SZ_SCC = %decimal '32',                                ! SET CONTROLLER CHARACTERISTICS
0          SZ_WRT = %decimal '32',                                ! WRITE
0          SZ_GEN = %decimal '32',                                ! GENERAL PACKET SIZE
0          SZ_REC = %DECIMAL '28',                    !                     ZZZ
0          SZ_SEN = %DECIMAL '28',                    !                     ZZZ
0          SZ_ELP = %DECIMAL '18',                    !                     ZZZ
0          SZ_ABT = %DECIMAL '12',                    !                     ZZZ
0          SZ_GDS = %DECIMAL '12',                    !                     ZZZ
0    !
0    !****** MSCP COMMAND MODIFIERS
0    !
0          MD_CMP           = %o'040000',                         ! COMPARE
```

```
;    0213  0          MD_EXP          = %o'100000',                          ! EXPRESS REQUEST
;    0214  0      !
;    0215  0      !****** CONNECTION ID VALUES (MSCP PKT, RETPKT)
;    0216  0      !               (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
;    0217  0      !
;    0218  0          CID_DISK        = 0,                                   ! DISK MSCP
;    0219  0          CID_MSCP        = 0,                                   !DISK MSCP
;    0220  0          CID_TAPE        = 1,                                   ! TAPE MSCP
;    0221  0          CID_DUP         = 2,                                   ! DIAGNOSTIC AND UTILITIES PROTOCOL
;    0222  0          CID_DRIVER      = 3,                                   ! EXERCISER "DRIVER"
;    0223  0      !
;    0224  0      !****** MESSAGE TYPE VALUES
;    0225  0      !
;    0226  0          MT_SEQ          = 0,                                   ! SEQUENTIAL (FROM PORT)
;    0227  0          MT_DG           = 1,                                   ! DATAGRAM (FROM PORT)
;    0228  0          MT_CRD          = 2,                                   ! CREDIT NOTIFICATION (FROM PORT)
;    0229  0          MT_FATAL        = 3,                                   ! FATAL DEVICE ERROR (FROM "DRIVER")
;    0230  0          MT_TIMEOUT      = 4,                                   ! COMMAND TIMEOUT (FROM "DRIVER")
;    0231  0      !
;    0232  0      !****** CONTROLLER FLAGS
;    0233  0      !               (IN SET CONTROLLER CHARACTERISTICS COMMAND AND RESPONSE)
;    0234  0      !
;    0235  0          CF_ATN          = %o'000200',                          ! ENABLE ATTENTION MESSAGES
;    0236  0          CF_MSC          = %o'000100',                          ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
;    0237  0          CF_OTH          = %o'000040',                          ! ENABLE OTHER HOST'S ERROR LOG MESSAGES
;    0238  0          CF_THS          = %o'000020',                          ! ENABLE THIS HOST'S ERROR LOG MESSAGES
;    0239  0      !   CF_MASK         = CF_ATN or CF_MSC or CF_THS,
;    0240  0          CF_MASK         = CF_MSC or CF_THS,                    ! RELEVENT BITS IN CTLR FLAGS WORD
;    0241  0      !
;    0242  0      !****** UNIT FLAGS
;    0243  0      !               (IN ONLINE COMMAND AND RESPONSE)
;    0244  0      !
;    0245  0          UF_REMOVABLE    = %o'000200',                          ! REMOVABLE MEDIA
;    0246  0          UF_WPH          = %o'020000',                          ! WRITE PROTECT (HARDWARE)
;    0247  0      !
;    0248  0      !****** STATUS / EVENT CODE DEFINITIONS
;    0249  0      !
;    0250  0          ST_SUC          = %o'0',                               ! SUCCESS
;    0251  0          ST_CMD          = %o'1',                               ! INVALID COMMAND
;    0252  0          ST_ABO          = %o'2',                               ! COMMAND ABORTED
;    0253  0          ST_OFL          = %o'3',                               ! UNIT OFFLINE
;    0254  0          ST_AVL          = %o'4',                               ! DRIVE AVAILABLE
;    0255  0          ST_MFE          = %o'5',                               ! MEDIA FORMAT ERROR
;    0256  0          ST_WPT          = %o'6',                               ! WRITE PROTECTED
;    0257  0          ST_CMP          = %o'7',                               ! COMPARE ERROR
;    0258  0          ST_DAT          = %o'10',                              ! DATA ERROR
;    0259  0          ST_HST          = %o'11',                              ! HOST BUFFER ACCESS ERROR
;    0260  0          ST_CNT          = %o'12',                              ! CONTROLLER ERROR
;    0261  0          ST_DRV          = %o'13',                              ! DRIVE ERROR
;    0262  0          ST_BRC          = %o'24',                              ! BAD BLOCK REPLACEMENT COMPLETION MMM
;    0263  0          ST_DIA          = %o'37',                              ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    0264  0      !
;    0265  0      !****** END MESSAGE FLAGS
```

```
;  0266  0     !
;  0267  0             EF_BBR          = %o'200',          ! BAD BLOCK REPORTED
;  0268  0             EF_BBU          = %o'100',          ! BAD BLOCK NOT REPORTED
;  0269  0     !
;  0270  0     !****** RDRX LITERALS
;  0271  0     !
;  0272  0             RCIP            = 0,                ! IP REGISTER
;  0273  0             RCSA            = 1,                ! SA REGISTER
;  0274  0     !
;  0275  0     !****** COMMON SA REGISTER BIT DEFINITIONS
;  0276  0     !
;  0277  0             SA_S1           = %o'004000',       ! STEP 1 STATUS BIT
;  0278  0             SA_S2           = %o'010000',       !   :  2
;  0279  0             SA_S3           = %o'020000',       !   :  3
;  0280  0             SA_S4           = %o'040000',       !   V  4
;  0281  0             SA_ERR          = %o'100000',       ! ERROR INDICATOR
;  0282  0             SA_INT          = %o'000200',       ! INTERRUPT ENABLE DURING INITIALIZATION
;  0283  0             SA_GO           = %o'000001',       ! GO BIT TO START FIRMWARE
;  0284  0     !
;  0285  0     !****** INITIALIZATION STEP READ MASKS
;  0286  0     !
;  0287  0             S1_MASK         = %o'176000',       ! STEP 1 READ BITS
;  0288  0             S2_MASK         = %o'174377',       !   :  2
;  0289  0             S3_MASK         = %o'174377',       !   :  3
;  0290  0             S4_MASK         = %o'174000',       !   V  4
;  0291  0     !
;  0292  0     !****** COMMAND TYPES
;  0293  0     !
;  0294  0             IMM_CMD         = 0,                ! IMMEDIATE COMMAND
;  0295  0             SEQ_CMD         = 1,                ! SEQUENTIAL COMMAND
;  0296  0             NON_SEQ_CMD     = 2,                ! NON-SEQUENTIAL COMMAND
;  0297  0     !
;  0298  0     !****** ERROR-LOG FORMAT TYPES
;  0299  0     !
;  0300  0             FORMAT_CNTR     = %o'0',            ! CONTROLLER ERROR
;  0301  0             FORMAT_HOST     = %o'1',            ! HOST MEMORY ACCESS ERROR
;  0302  0             FORMAT_XFER     = %o'2',            ! DISK TRANSFER ERROR
;  0303  0             FORMAT_SDI      = %o'3',            ! 'STANDARD DISK INTECONNECT' ERROR
;  0304  0             FORMAT_SDE      = %o'4',            ! SMALL DISK ERROR
;  0305  0             FORMAT_BRA      = %o'11',           ! BAD BLOCK REPLACEMENT ATTEMPT MMM
;  0306  0     !
;  0307  0     !****** ERROR-LOG BLOCK NUMBER INFORMATION
;  0308  0     !
;  0309  0             TYPE_LBN        = %o'0000',         ! LOGICAL BLOCK NUMBER
;  0310  0             TYPE_RBN        = %o'0110',         ! REPLACEMENT BLOCK NUMBER
;  0311  0     !
;  0312  0     !****** MSCP DISK MODEL CODES
;  0313  0     !
;  0314  0     !           MODEL_RX50      = 7,            ! RX50  THESE ARE NO LONGER USED. THE
;  0315  0     !           MODEL_RD51      = 6,            ! RD51  MODEL IS DETERMINED ANOTHER WAY.
;  0316  0     !           MODEL_RD52      = 8,            ! RD52
;  0317  0     !
;  0318  0     !****** LITERALS FOR READABILITY
```

```
;    0319  0     !
;    0320  0           YES             = 1,
;    0321  0           NO              = 0,
;    0322  0           TRUE            = 1,
;    0323  0           FALSE           = 0,
;    0324  0           SUCCESS         = 1,
;    0325  0           FAILURE         = 0,
;    0326  0           FOUND           = 1,
;    0327  0           NOT_FOUND       = 0,
;    0328  0           PRESENT         = 1,                ! DISK IS PRESENT IN CONTROLLER
;    0329  0           NOT_PRESENT     = 0,                ! DISK IS NOT PRESENT IN CONTROLLER
;    0330  0           UNPROTECTED     = 1,                ! DISK HAS UNPROTECTED CUSTOMER LBN'S
;    0331  0           PROTECTED       = 0,                ! DISK HAS PROTECTED CUSTOMER LBN'S
;    0332  0           ONLINE          = 1,
;    0333  0           OFFLINE         = 0,
;    0334  0           IDLE            = 0,                !IDLE
;    0335  0           ACTIVE          = 1,                !ACTIVE
;    0336  0           FULL            = 1,                ! ERROR-LOG SAVE PACKET FILLED
;    0337  0           EMPTY           = 0,                ! ERROR-LOG SAVE PACKET PRINTED
;    0338  0           HRD_OCCURED     = 1,                ! HARD ERROR DETECTED IN RESPONSE PACKET
;    0339  0           HRD_NOT_OCCURED = 0,                ! HARD ERROR NOT DETECTED
;    0340  0           ALL_ONES        = %o'177777';
```

```
;    0341  0     !*****************************************************************************
;    0342  0     !                                                                            *
;    0343  0     !                          F I E L D S                                       *
;    0344  0     !                                                                            *
;    0345  0     !*****************************************************************************
;    0346  0     !
;    0347  0     FIELD
;    0348  0     !
;    0349  0     !****** HARDWARE P-TABLE FIELDS
;    0350  0     !
;    0351  0     HWP_FIELDS =
;    0352  0             set
;    0353  0             HWP_IP_ADDR     = [0, 0, 16, 0],                ! IP ADDRESS
;    0354  0             HWP_VECTOR      = [1, 0, 16, 0],                ! VECTOR ADDRESS
;    0355  0             HWP_BR_LEVEL    = [2, 0, 16, 0],                ! BUS REQUEST LEVEL
;    0356  0             HWP_DISK        = [3, 0, 16, 0],                ! DISK (ALL FIELDS)
;    0357  0             HWP_DISK_NUM    = [3, 0, 4, 0],                 ! DISK NUMBER
;    0358  0             HWP_DISK_TYPE   = [3, 4, 1, 0],                 ! DISK TYPE
;    0359  0             HWP_DISK_DUPEX  = [3, 5, 1, 0],                 !RUN DUP EXERCISER              !ZZZ
;    0360  0             HWP_DISK_DUPWT  = [3, 6, 1, 0],                 !DUP WRITE FLAG                 !ZZZ
;    0361  0             HWP_ENTIRE      = [3, 7, 1, 0],                 !TEST ENTIRE DISK               !ZZZ
;    0362  0             HWP_DISK_CP     = [3, 15, 1, 0],                ! PROTECT CUSTOMER DATA BIT
;    0363  0             HWP_BEG_TRK     = [4, 0, 16, 0],                ! BEGINNING TRACK LO            !ZZZ
;    0364  0             HWP_BEG_TRK1    = [5, 0, 16, 0],                ! BEGINNING TRACK HI            !ZZZ
;    0365  0             HWP_END_TRK     = [6, 0, 16, 0],                ! ENDING TRACK LO               !ZZZ
;    0366  0             HWP_END_TRK1    = [7, 0, 16, 0]                 ! ENDING TRACK HI               !ZZZ
;    0367  0             tes,
;    0368  0     !
;    0369  0     !****** COMMUNICATION AREA HEADER FIELDS
;    0370  0     !
;    0371  0     COM_FIELDS =
;    0372  0             set
;    0373  0             ADAP_CH         = [1, 8, 8, 0],                 ! ADAPTER CHANNEL NUMBER FOR PURGES
;    0374  0             CMD_INT         = [2, 0, 16, 0],                ! COMMAND RING INTERRUPT
;    0375  0             RSP_INT         = [3, 0, 16, 0]                 ! RESPONSE RING INTERRUPT
;    0376  0             tes,
;    0377  0
;    0378  0     !                                                                                ZZZ
;    0379  0     !       DUP BUFFER FIELD                                                         ZZZ
;    0380  0     !                                                                                ZZZ
;    0381  0     DP_FIELDS =                                                                     !ZZZ
;    0382  0             SET                                                                     !ZZZ
;    0383  0             DUPBF0          = [0, 0, 16, 0],                                        !ZZZ
;    0384  0             DUPBF1          = [1, 0, 16, 0],                                        !ZZZ
;    0385  0             DUPBF2          = [2, 0, 16, 0],                                        !ZZZ
;    0386  0             DUPTYPE         = [0, 12, 4, 0],                                        !ZZZ
;    0387  0             DUPMESG         = [0, 0, 12, 0]                                         !ZZZ
;    0388  0             TES,                                                                    !ZZZ
;    0389  0
;    0390  0
;    0391  0     !****** CONTROLLER STATUS TABLE (CST) FIELDS
;    0392  0     !
;    0393  0     CST_FIELDS =
```

L6


```
;  0394  0          set
;  0395  0          IP_ADDR         = [0, 0, 16, 0],                ! IP ADDRESS
;  0396  0          VEC_ADDR        = [1, 0, 9, 0],                 ! VECTOR ADDRESS
;  0397  0          STATE           = [1, 15, 1, 0],                ! CONTROLLER STATUS
;  0398  0          BR_LEV          = [2, 0, 8, 0],                 ! BUS REQUEST LEVEL
;  0399  0          U_CNT           = [2, 8, 8, 0]                  ! NUMBER OF UNITS (DISKS) FOR THIS CONTROLLER
;  0400  0
;  0401  0          DO_ALL          = [3, 0, 16, 0],                ! DISK 0 (ALL FIELDS)
;  0402  0          DO_DISK_NUM     = [3, 0, 4, 0],                 ! DISK NUMBER
;  0403  0          DO_TYPE         = [3, 4, 1, 0],                 ! DISK TYPE
;  0404  0          DO_UNIT         = [3, 8, 4, 0],                 ! DISK 0 UNIT NUMBER (DRS UNIT)
;  0405  0          DO_FATAL        = [3, 12, 1, 0],                ! DISK 0 FATAL ERROR BIT
;  0406  0          DO_STAT         = [3, 13, 1, 0],                ! DISK 0 STATUS BIT
;  0407  0          DO_PRES         = [3, 14, 1, 0],                ! DISK 0 PRESENT BIT
;  0408  0          DO_PROT         = [3, 15, 1, 0],                ! DK 0 PROTECT CUSTOMER DATA
;  0409  0          DO_BEG0         = [4, 0, 16, 0],        !DK 0 BEGIN TK LO       ZZZ
;  0410  0          DO_BEG1         = [5, 0, 16, 0],        !DK 0 BEGIN TK HI       ZZZ
;  0411  0          DO_END0         = [6, 0, 16, 0],        !DK 0 END TK LO         ZZZ
;  0412  0          DO_END1         = [7, 0, 16, 0],        !DK 0 END TK HI         ZZZ
;  0413  0          DO_NAME0        = [8, 0, 8, 0],         !DK 0 MANE BYTE 0       ZZZ
;  0414  0          DO_NAME1        = [8, 8, 8, 0],         !DK 0 NAME BYTE 1       ZZZ
;  0415  0          DO_NAME2        = [9, 0, 8, 0],         !DK 0 NAME BYTE 2       ZZZ
;  0416  0          DO_NAME3        = [9, 8, 8, 0],         !DK 0 NAME BYTE 3       ZZZ
;  0417  0          DO_NUL          = [10, 0, 16, 0]        !NUL AFTER NAME         ZZZ
;  0418  0          DO_DBN          = [11, 0, 8, 0],        !DK 0 RELATIVE DBN      ZZZ
;  0419  0          DO_WRITE        = [11, 12, 1, 0],       !DK 0 DUP WRITE FLAG    ZZZ
;  0420  0          DO_ACTIVE       = [11, 13, 1, 0],       !DK 0 ACTIVE FLAG       ZZZ
;  0421  0          DO_DUPERROR     = [11, 14, 1, 0],       !DK 0 DUP ERROR FLAG    ZZZ
;  0422  0          DONODUPMED      = [11, 15, 1, 0],       !DK 0 NO DUP MEDIA FLAG ZZZ
;  0423  0          DO_COUNT        = [12, 0, 16, 0],       !DK 0 RELATIVE MSCP FUN-ZZZ
;  0424  0                                                  !     CTION COUNTER     ZZZ
;  0425  0                                                  !                       ZZZ
;  0426  0          ! REPEAT WORDS 3 THROUGH 12 ABOVE AS: !                         ZZZ
;  0427  0          ! WORDS 13 THROUGH 21 FOR DRIVE 1     !                         ZZZ
;  0428  0          ! WORDS 22 THROUGH 30 FOR DRIVE 2     !                         ZZZ
;  0429  0          ! WORDS 31 THROUGH 39 FOR DRIVE 3     !                         ZZZ
;  0430  0          !                                     !                         ZZZ
;  0431  0          !                                                               ZZZ
;  0432  0          tes,
;  0433  0          !
;  0434  0          !****** MSCP PACKET FIELDS
;  0435  0          !               (NOTE: BASE ADDRESS OF PACKET REFERENCES THE PACKET'S OWN
;  0436  0          !               BUFFER DESCRIPTOR, RATHER THAN THE MESSAGE BODY (TEXT + 0).
;  0437  0          !               SEE DOCUMENTATION FOR LAYOUT OF MSCP PACKETS.)
;  0438  0          !
;  0439  0     PKT_FIELDS =
;  0440  0          set
;  0441  0          !
;  0442  0          !       HEADER FIELDS
;  0443  0          !
;  0444  0          PKT_LO          = [0, 0, 16, 0],                ! PACKET DESCRIPTOR (LO ORDER)
;  0445  0          PKT_HI          = [1, 0, 16, 0],                ! PACKET DESCRIPTOR (HI ORDER - ALL FIELDS)
;  0446  0          PKT_U           = [1, 0, 2, 0],                 ! PACKET DESCRIPTOR (HI ORDER UNIBUS BITS)
```

```
;  0447  0                PKT_Q           = [1, 2, 4, 0],                 ! PACKET DESCRIPTOR (HI ORDER Q-BUS BITS)
;  0448  0                PKT_F           = [1, 14, 1, 0],                ! PACKET DESCRIPTOR FLAG BIT
;  0449  0                PKT_O           = [1, 15, 1, 0],                ! PACKET DESCRIPTOR OWNERSHIP BIT
;  0450  0                CMD_TYPE        = [2, 0, 8, 0],                 ! COMMAND TYPE
;  0451  0                RSP_RECEIVED    = [2, 8, 8, 0],                 ! FLAG SET IF RESPONSE TO COMMAND RECEIVED
;  0452  0                MSGLEN          = [3, 0, 16, 0],                ! MESSAGE LENGTH
;  0453  0                CREDITS         = [4, 0, 4, 0],                 ! CREDITS
;  0454  0                MSGTYP          = [4, 4, 4, 0],                 ! MESSAGE TYPE
;  0455  0                CONNID          = [4, 8, 8, 0],                 ! CONNECTION ID
;  0456  0        !
;  0457  0        !!      GENERIC COMMAND PACKET AND END PACKET HEADER FILEDS
;  0458  0        !
;  0459  0                CRN_LO          = [5, 0, 16, 0],                ! COMMAND REF NUMBER (LO ORDER)
;  0460  0                CRN_HI          = [6, 0, 16, 0],                ! COMMAND REF NUMBER (HI ORDER)
;  0461  0                DK_NUM          = [7, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0462  0                OPCODE          = [9, 0, 8, 0],                 ! OPCODE AND ENDCODE
;  0463  0                MODIFY          = [10, 0, 16, 0],               ! COMMAND MODIFIERS
;  0464  0                STATUS_CODE     = [10, 0, 5, 0],                ! STATUS (PART OF RESPONSE PACKET)
;  0465  0                STATUS_SUBCODE  = [10, 5, 11, 0],               ! SUBCODE (PART OF RESPONSE PACKET)
;  0466  0        !
;  0467  0        !!      READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)
;  0468  0        !
;  0469  0                BC_LO           = [11, 0, 16, 0],               ! BYTE COUNT (LO ORDER)
;  0470  0                BC_HI           = [12, 0, 16, 0],               ! BYTE COUNT (HI ORDER)
;  0471  0                BUF_0           = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR
;  0472  0                BUF_1           = [14, 0, 16, 0],               !       :
;  0473  0                BUF_2           = [15, 0, 16, 0],               !       :
;  0474  0                BUF_3           = [16, 0, 16, 0],               !       :
;  0475  0                BUF_4           = [17, 0, 16, 0],               !       :
;  0476  0                BUF_5           = [18, 0, 16, 0],               !       V
;  0477  0                LBN_L           = [19, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0478  0                LBN_H           = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0479  0
;  0480  0        !
;  0481  0        !!      DUP PROGRAM LETTER FIELDS (FOR EXECUTE LOCAL PROGRAM CMD)               ZZZ
;  0482  0        !                                                                               ZZZ
;  0483  0                L1              = [11, 0, 8, 0],        !LETTER NO 1                   ZZZ
;  0484  0                L2              = [11, 8, 8, 0],        !LETTER NO 2                   ZZZ
;  0485  0                L3              = [12, 0, 8, 0],        !LETTER NO 3                   ZZZ
;  0486  0                L4              = [12, 8, 8, 0],        !LETTER NO 4                   ZZZ
;  0487  0                L5              = [13, 0, 8, 0],        !LETTER NO 5                   ZZZ
;  0488  0                L6              = [13, 8, 8, 0],        !LETTER NO 6                   ZZZ
;  0489  0        !
;  0490  0        !!      SET CONTROLLER CHARACTERISTICS COMMAND FIELDS
;  0491  0        !
;  0492  0                C_FLAGS         = [12, 0, 16, 0],               ! CONTROLLER FLAGS
;  0493  0        !
;  0494  0        !!      ONLINE COMMAND FIELDS
;  0495  0        !
;  0496  0                U_FLAGS         = [12, 0, 16, 0],               ! UNIT FLAGS
;  0497  0                DDPAR           = [19, 0, 16, 0]                ! DEVICE-DEPENDENT PARAMETERS
;  0498  0                tes,
;  0499  0        !
```

```
;  0500  0     !****** RETURN PACKET (RETPKT) FIELDS
;  0501  0     !                (SIMILAR, BUT NOT IDENTICAL, TO MSCP PACKET FIELDS)
;  0502  0     !
;  0503  0     RP_FIELDS =
;  0504  0             set
;  0505  0     !
;  0506  0     !       COMMON TO ALL RETURN PACKETS FROM DISK MSCP
;  0507  0     !
;  0508  0             MESLEN          = [0, 0, 16, 0],                ! MESSAGE LENGTH
;  0509  0             CTLR            = [1, 0, 4, 0],                 ! CONTROLLER NUMBER (CREDITS OVERWRITTEN)
;  0510  0             MESTYP          = [1, 4, 4, 0],                 ! MESSAGE TYPE
;  0511  0             CONID           = [1, 8, 8, 0],                 ! CONNECTION ID
;  0512  0             CRF_LO          = [2, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (LO ORDER)
;  0513  0             CRF_HI          = [3, 0, 16, 0],                ! COMMAND REFERENCE NUMBER (HI ORDER)
;  0514  0             DISR            = [4, 0, 16, 0],                ! DISK ADDRESS (RD/RX DISK NUMBER)
;  0515  0             CMDMOD          = [5, 0, 16, 0],                ! COMMAND MODIFIERS
;  0516  0             ENDCOD          = [6, 0, 8, 0],                 ! END CODE
;  0517  0             FLAGS           = [6, 8, 8, 0],                 ! FLAGS
;  0518  0             STATUS          = [7, 0, 16, 0],                ! STATUS AND SUB-CODE
;  0519  0             STSCOD          = [7, 0, 5, 0],                 ! STATUS CODE
;  0520  0             SUBCOD          = [7, 5, 11, 0],                ! SUB-CODE
;  0521  0     !
;  0522  0     !       READ, WRITE, AND ACCESS COMMAND RETURN PACKETS
;  0523  0     !
;  0524  0             BCNT_LO         = [8, 0, 16, 0],                ! BYTE COUNT (LO ORDER)
;  0525  0             BCNT_HI         = [9, 0, 16, 0],                ! BYTE COUNT (HI ORDER)
;  0526  0             BUFF_0          = [10, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 0)
;  0527  0             BUFF_1          = [11, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 1)
;  0528  0             BUFF_2          = [12, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 2)
;  0529  0             BUFF_3          = [13, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 3)
;  0530  0             BUFF_4          = [14, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 4)
;  0531  0             BUFF_5          = [15, 0, 16, 0],               ! I/O BUFFER DESCRIPTOR (WORD 5)
;  0532  0             BBLK_LO         = [16, 0, 16, 0],               ! FIRST BAD BLOCK (LO ORDER)
;  0533  0             BBLK_HI         = [17, 0, 16, 0],               ! FIRST BAD BLOCK (HI ORDER)
;  0534  0             CBCNT_LO        = [18, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (LO ORDER)
;  0535  0             CBCNT_HI        = [19, 0, 16, 0],               ! BYTE COUNT FROM CMD PACKET (HI ORDER)
;  0536  0             LBN_LO          = [20, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (LO ORDER)
;  0537  0             LBN_HI          = [21, 0, 16, 0],               ! LOGICAL BLOCK NUMBER (HI ORDER)
;  0538  0     !
;  0539  0     !       SET CONTROLLER CHARACTERISTICS RETURN PACKET
;  0540  0     !
;  0541  0             C_FLGS          = [9, 0, 16, 0],                ! CONTROLLER FLAGS
;  0542  0             C_TIME          = [10, 0, 16, 0],               ! CONTROLLER TIMEOUT
;  0543  0     !
;  0544  0     !       UNIT ONLINE RETURN PACKET
;  0545  0     !
;  0546  0             U_FLGS          = [9, 0, 16, 0],                ! UNIT FLAGS
;  0547  0             R_MODEL         = [13, 0, 8, 0],                ! 2 DIGIT MODEL NUMBER                          ZZZ
;  0548  0             NAME_NUM        = [14, 0, 6, 0],                ! MODEL NAME - 2 DIGIT NUMBER
;  0549  0             NAME_1_LO       = [14, 12, 4, 0],               ! MODEL NAME - 2ND CHARACTER (LOW  ORDER 4 BITS)
;  0550  0             NAME_1_HI       = [15, 0, 1, 0],                ! MODEL NAME - 2ND CHARACTER (HIGH ORDER 1 BIT)
;  0551  0             NAME_0          = [15, 1, 5, 0],                ! MODEL NAME - 1ST CHARACTER
;  0552  0     !ZZZ    USIZ_LO         = [18, 0, 16, 0],               ! UNIT SIZE (LO ORDER)
```

```
;    0553  0     !ZZZ    USIZ_HI         = [19, 0, 16, 0],       ! UNIT SIZE (HI ORDER)
;    0554  0             SIZE0           = [18, 0, 16, 0],       !LOWER WD OF MAX LBNS OR UNIT SIZE      ZZZ
;    0555  0             SIZE1           = [19, 0, 16, 0]        !UPPER WD     "       "       "         ZZZ
;    0556  0             tes,
;    0557  0     !
;    0558  0     !****** STATISTICS TABLE (TALLY) FIELDS
;    0559  0     !
;    0560  0     T_FIELDS =
;    0561  0             set
;    0562  0             BYTES_READ_LO   = [0, 0, 16, 0],        ! NUMBER OF BYTES READ (LO ORDER)
;    0563  0             BYTES_READ_HI   = [1, 0, 16, 0],        ! NUMBER OF BYTES READ (HI ORDER)
;    0564  0             MBYTES_READ     = [2, 0, 16, 0],        ! MEGABYTES READ
;    0565  0             BYTES_WRIT_LO   = [3, 0, 16, 0],        ! NUMBER OF BYTES WRITTEN (LO ORDER)
;    0566  0             BYTES_WRIT_HI   = [4, 0, 16, 0],        ! NUMBER OF BYTES WRITTEN (HI ORDER)
;    0567  0             MBYTES_WRIT     = [5, 0, 16, 0],        ! MEGABYTES WRITTEN
;    0568  0             ERR_HARD        = [6, 0, 16, 0],        ! NUMBER OF HARD ERRORS
;    0569  0     !
;    0570  0             TOT_READS_LO    = [7, 0, 16, 0],        ! TOTAL NUMBER OF READS (LO ORDER)
;    0571  0             TOT_READS_HI    = [8, 0, 16, 0],        ! TOTAL NUMBER OF READS (HI ORDER)
;    0572  0             TOT_WRITES_LO   = [10, 0, 16, 0],       ! TOTAL NUMBER OF WRITES (LO ORDER)
;    0573  0             TOT_WRITES_HI   = [11, 0, 16, 0],       ! TOTAL NUMBER OF WRITES (HI ORDER)
;    0574  0             TOT_BYT_RED_LO  = [13, 0, 16, 0],       ! TOTAL BYTES READ (LO ORDER)
;    0575  0             TOT_BYT_RED_HI  = [14, 0, 16, 0],       ! TOTAL BYTES READ (HI ORDER)
;    0576  0             MTOT_BYT_RED    = [15, 0, 16, 0],       ! TOTAL MEGABYTES READ
;    0577  0             TOT_BYT_WRT_LO  = [16, 0, 16, 0],       ! TOTAL BYTES WRITTEN (LO ORDER)
;    0578  0             TOT_BYT_WRT_HI  = [17, 0, 16, 0],       ! TOTAL BYTES WRITTEN (HI ORDER)
;    0579  0             MTOT_BYT_WRT    = [18, 0, 16, 0],       ! TOTAL MEGABYTES WRITTEN
;    0580  0             ERR_HRD_SEK     = [19, 0, 8, 0],        ! TOTAL HARD ERRORS - SEEK
;    0581  0             ERR_HRD_DAT     = [19, 8, 8, 0],        ! TOTAL HARD ERRORS - DATA
;    0582  0             ERR_HRD_DRV     = [20, 0, 8, 0],        ! TOTAL HARD ERRORS - DRIVE
;    0583  0             ERR_HRD_HST     = [20, 8, 8, 0],        ! TOTAL HARD ERRORS - HOST
;    0584  0             ERR_SFT_SEK     = [21, 0, 8, 0],        ! TOTAL SOFT ERRORS - SEEK
;    0585  0             ERR_SFT_DAT     = [21, 8, 8, 0],        ! TOTAL SOFT ERRORS - DATA
;    0586  0             ERR_SFT_DRV     = [22, 0, 8, 0],        ! TOTAL SOFT ERRORS - DRIVE
;    0587  0             ERR_SFT_HST     = [22, 8, 8, 0],        ! TOTAL SOFT ERRORS - HOST
;    0588  0             T_BLK_WT        = [23, 0,16, 0],        !                       ZZZ
;    0589  0             T_DBN_WT        = [24, 0,16, 0],        !DBNS WRITTEN           ZZZ
;    0590  0             T_BLK_RD        = [25, 0,16, 0],        !                       ZZZ
;    0591  0             T_DBN_RD        = [26, 0,16, 0]         !DBNS READ              ZZZ
;    0592  0
;    0593  0             tes,
;    0594  0     !
;    0595  0     !****** CONTROLLER ERROR TALLY FIELDS
;    0596  0     !
;    0597  0     C_ERR_FIELDS =
;    0598  0             set
;    0599  0             C_ERR_HRD       = [0, 0, 8, 0],         ! HARD ERRORRS
;    0600  0             C_ERR_SFT       = [0, 8, 8, 0]          ! SOFT ERRORS
;    0601  0             tes,
;    0602  0     !
;    0603  0     !****** DRIVER CONTROLLER TABLE (DCT) FIELDS
;    0604  0     !
;    0605  0     DCT_FIELDS =
```

```
;  0606  0              set
;  0607  0              WORD0           = [0, 0, 16, 0],
;  0608  0              CRING_CNT       = [0, 0, 8, 0],
;  0609  0              IG_INT          = [0, 14, 1, 0],
;  0610  0              STAT            = [0, 15, 1, 0],
;  0611  0              SA_SAVE         = [1, 0, 16, 0],
;  0612  0              RR_BEG          = [2, 0, 16, 0],
;  0613  0              RR_END          = [3, 0, 16, 0],
;  0614  0              CR_BEG          = [4, 0, 16, 0],
;  0615  0              CR_END          = [5, 0, 16, 0],
;  0616  0              RR_POLL         = [6, 0, 16, 0],
;  0617  0              CR_POLL         = [7, 0, 16, 0],
;  0618  0              CR_NEXT         = [8, 0, 16, 0]
;  0619  0              tes,
;  0620  0      !
;  0621  0      !****** ERROR LOG PACKET SAVE AREA FIELDS
;  0622  0      !
;  0623  0      EP_FIELDS =
;  0624  0              set
;  0625  0              EL_CNTR         = [0, 0, 8, 0],
;  0626  0              EL_CONTENTS     = [0, 8, 8, 0],
;  0627  0              EL_MSGLEN       = [1, 0, 16, 0],
;  0628  0              EL_CRN_LO       = [3, 0, 16, 0],
;  0629  0              EL_CRN_HI       = [4, 0, 16, 0],
;  0630  0              EL_DK_NUM       = [5, 0, 16, 0],
;  0631  0              EL_FORMAT       = [7, 0, 8, 0],
;  0632  0              EL_EDR          = [7, 12, 1, 0],
;  0633  0              EL_BRR          = [7, 13, 1, 0],
;  0634  0              EL_CONTINUE     = [7, 14, 1, 0],
;  0635  0              EL_SUCCESS      = [7, 15, 1, 0],
;  0636  0              EL_CODE         = [8, 0, 5, 0],
;  0637  0              EL_SUBCODE      = [8, 5, 11, 0],
;  0638  0              EL_RETRY        = [20, 8, 8, 0],
;  0639  0              EL_BLOCK        = [23, 0, 16, 0],
;  0640  0              EL_BLOCK_HI     = [24, 0, 12, 0],
;  0641  0              EL_BLOCK_TYPE   = [24, 12, 4, 0]
;  0642  0              tes,
;  0643  0      !
;  0644  0      !****** INFORMATION ABOUT LAST RESPONSE PACKET
;  0645  0      !
;  0646  0      LAST_PKT_FIELDS =
;  0647  0              set
;  0648  0              LAST_HRD_ERR    = [0, 0, 16, 0],
;  0649  0              LAST_CRN_LO     = [1, 0, 16, 0],
;  0650  0              LAST_CRN_HI     = [2, 0, 16, 0]
;  0651  0              tes,
;  0652  0      !
;  0653  0      !****** RDRX REGISTER FIELDS
;  0654  0      !
;  0655  0      RC_REG =
;  0656  0              set
;  0657  0              RC_ALL          = [0, 16, 0]
;  0658  0              tes;
```

```
! ALL FIELDS IN WORD 0
! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
! IGNORE INTERRUPT BIT
! ONLINE / OFFLINE STATUS
! SA REGISTER SAVE WORD
! FIXED ADDRESSES OF START AND
!       END OF EACH RING
!              :
!              V
! ADDR OF NEXT RRING SLOT TO BE POLLED
! ADDR OF NEXT CRING SLOT TO BE POLLED
! ADDR OF NEXT AVAIL CRING SLOT

! CONTROLLER NUMBER
! FLAG INDICATES IF PACKET CONTENTS ALREADY PRINTED
! PACKET LENGTH
! COMMAND REFERENCE NUMBER
!
! DISK ADDRESS (RD/RX DISK NUMBER)
! FORMAT
! ERROR DURING REPLACEMENT FLAG MMM
! BAD BLOCK REPLACEMENT REQUEST FLAG  MMM
! CONTINUE FLAG
! SUCCESS FLAG
! ERROR CODE
! SUB CODE
! RETRY COUNT
! BLOCK NUMBER
!HIGH BITS OF BLOCK NUMBER                    ZZZ
! TYPE OF BLOCK NUMBER INFO RETURNED

! FLAG INDICATES IF HARD ERROR OCCURED
! COMMAND REFERENCE NUMBER
!

! DEFINE ALL BITS
```

D7


```
;  0659  0     !*****************************************************************************
;  0660  0     !                                                                            *
;  0661  0     !                          M A C R O S                                       *
;  0662  0     !                                                                            *
;  0663  0     !*****************************************************************************
;  0664  0     !
;  0665  0     macro
;  0666  0     !
;  0667  0     !****** CST FIELDS. MODEL FOR WDS 3-12, 13-21, 22-30, AND 31-39.       ZZZ
;  0668  0     !
;  0669  0             D_ALL           = 0, 16, 0%,                    ! ALL FIELDS
;  0670  0             D_DISK_NUM      = 0, 4, 0%,                     ! DISK ADDRESS
;  0671  0             D_TYPE          = 4, 1, 0%,                     !DISK TYPE - 1 BIT      ZZZ
;  0672  0             D_UNIT          = 8, 4, 0%,                     ! DISK UNIT NUMBER (DRS UNIT)
;  0673  0             D_FATAL         = 12, 1, 0%,                    ! FATAL ERROR BIT
;  0674  0             D_STAT          = 13, 1, 0%,                    ! DISK STATUS BIT
;  0675  0             D_PRES          = 14, 1, 0%,                    ! DISK PRESENT BIT
;  0676  0             D_PROT          = 15, 1, 0%,                    ! DISK PROTECTION BIT
;  0677  0             D_BEGO          = 0, 16, 0%,            !BEGIN TRACK LO         ZZZ
;  0678  0             D_BEG1          = 0, 16, 0%,            !BEGIN TRACK HI         ZZZ
;  0679  0             D_ENDO          = 0, 16, 0%,            !END TRACK LO           ZZZ
;  0680  0             D_END1          = 0, 16, 0%,            !END TRACK HI           ZZZ
;  0681  0             D_NAME_0        = 0, 8, 0%,             ! NAME (FIRST  CHARACTER)
;  0682  0             D_NAME_1        = 8, 8, 0%,             ! NAME (SECOND CHARACTER)
;  0683  0             D_NAME_2        = 0, 8, 0%,             ! NAME (THIRD  CHARACTER)
;  0684  0             D_NAME_3        = 8, 8, 0%,             ! NAME (FOURTH CHARACTER)
;  0685  0     !       D_NUL           = 0, 16, 0%,            !NUL AFTER NAME         ZZZ
;  0686  0             D_DBN           = 0, 8, 0%,             !RELATIVE DBN           ZZZ
;  0687  0             DOPWRITE        = 12, 1, 0%,            !DUP WRITE FLAG         ZZZ
;  0688  0             D_ACTIVE        = 13, 1, 0%,            !ACTIVE STATE           ZZZ
;  0689  0             DOPERROR        = 14, 1, 0%,            !DUP ERROR FLAG         ZZZ
;  0690  0             NODUPMEDIA      = 15, 1, 0%,            !NO DUP MEDIA           ZZZ
;  0691  0             D_COUNT         = 0, 16, 0%,            !MSCP FUNCTION COUNTER  ZZZ
;  0692  0
;  0693  0
;  0694  0     !*********** BST FIELDS *************                                      ZZZ
;  0695  0     !                                                                       ZZZ
;  0696  0             HI_WRD          = 1, 0, 16, 0%,         !HI LBN                 ZZZ
;  0697  0             LO_WRD          = 0, 0, 16, 0%,         !LO LBN                 ZZZ
;  0698  0     !
;  0699  0     !****** BIT TEST
;  0700  0     !                   (CAUTION: THE FIRST ARGUMENT IS THE ADDRESS AND NOT THE CONTENTS)
;  0701  0     !
;M 0702  0     BIT_TST (ADDR, EXPECTED) =
;M 0703  0         (if (.ADDR and EXPECTED) eql EXPECTED
;M 0704  0         then
;M 0705  0             TRUE
;M 0706  0         else
;  0707  0             FALSE )%,
;  0708  0     !
;  0709  0     !****** RDRX WRITE
;  0710  0     !
;M 0711  0     WRT_RDRX (O, FIELDNAM, IMAGE) =
```

E7


```
;  M 0712  0          begin
;  M 0713  0          local
;  M 0714  0              RC_REG;
;  M 0715  0          RC_REG <%fieldexpand (FIELDNAM)> = IMAGE;
;  M 0716  0          (.RDRX_ADDR + (%upval * 0)) = .RC_REG;
;    0717  0          end%;
```

F7


```
;  0718  0     !*****************************************************************************
;  0719  0     !                                                                            *
;  0720  0     !                          S T R U C T U R E S                               *
;  0721  0     !                                                                            *
;  0722  0     !*****************************************************************************
;  0723  0     !
;  0724  0     !****** NIBBLE (4-BIT) VECTOR STRUCTURE
;  0725  0     !
;  0726  0     !structure
;  0727  0     !    NIBVECTOR [I; N] =
;  0728  0     !        [(N + 1) / 2]
;  0729  0     !        (NIBVECTOR + I / 2) <(I * 2) and 4, 4>;
;  0730  0     !
;  0731  0     !****** RDRX ACCESS ALGORITHM
;  0732  0     !
;  0733  0     structure
;  0734  0         RDRX [O, P, S, E] =
;  0735  1             begin
;  0736  1             local
;  0737  1                 RC_REG;
;  0738  1             RC_REG = .(RDRX + %upval * 0) <0, %bpval, 0>;
;  0739  1             RC_REG
;  0740  1             end
;  0741  0             <P, S, E>;
```

```
;                                   COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQDA0.REQ/LIST=ZRQDA0.LIS/LIBRARY=ZRQDA0.L16/SOURCE=PAGE:53

; Run Time:          00:04.5
; Elapsed Time:      00:06.9
; Lines/CPU Min:      9836
; Lexemes/CPU-Min: 51955
; Memory Used:  72 pages
; Library Precompilation Complete
```

```
Partition name : DUMMY
Identification : V02.3
Task  UIC      : [203,4]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 17792. words
Task address limits: 002000 107333
R-W disk blk limits: 000002 000107 000106 00070.

*** Root segment: ZRQDA0


R/W mem  limits: 002000 107331 105332 35546.
Disk blk limits: 000002 000107 000106 00070.


Memory allocation synopsis:

Section                                          Title  Ident  File
-------                                          -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON)      002000 000000 00000.
$CODE$:(RO,I,LCL,REL,CON)      002000 075120 31312.
                               002000 024172 10362.  ZRQDM1 V02.3  ZRQDA0.OB1;1
                               026172 014244 06308.  ZRQDM2 V02.3  ZRQDA0.OB1;1
                               042436 034036 14366.  ZRQDM3 V02.3  ZRQDA0.OB2;1
                               076474 000316 00206.  B16MUL 2.8    NOEIS.OLB;1
                               077012 000106 00070.  B16SAV 2.4    NOEIS.OLB;1
$FFF$ :(RW,D,GBL,REL,CON)      077120 006064 03124.
                               077120 006064 03124.  ZRQDM1 V02.3  ZRQDA0.OB1;1
$GGG$ :(RO,I,LCL,REL,CON)      105204 001310 00712.
                               105204 001310 00712.  ZRQDM3 V02.3  ZRQDA0.OB2;1
$OWN$ :(RW,D,LCL,REL,CON)      106514 000244 00164.
                               106514 000244 00164.  ZRQDM2 V02.3  ZRQDA0.OB1;1
$PLIT$:(RO,D,LCL,REL,CON)      106760 000222 00146.
                               106760 000222 00146.  ZRQDM3 V02.3  ZRQDA0.OB2;1
$XYZ$ :(RO,I,LCL,REL,CON)      107202 000130 00088.
                               107202 000130 00088.  ZRQDM4 V01.9  ZRQDA0.OB2;1


Global symbols:

ADDR.V 105203-R  BIT06  000100   BIT4   000020   BSI    077300-R  CRLF   026024-R  DBM101 006310-R  DBM126 007342-R
ADR    000020    BIT07  000200   BIT5   000040   BUFF.A 104760-R  CRN.HI 105072-R  DBM104 006336-R  DBM127 007422-R
ASTERI 026036-R  BIT08  000400   BIT6   000100   BUFF.O 105000-R  CRN.LO 105070-R  DBM105 006400-R  DBM128 007500-R
AZINT  070714-R  BIT09  001000   BIT7   000200   BYTS.P 105050-R  CSR.AD 105160-R  DBM107 006436-R  DBM15  005266-R
AZINTC 070676-R  BIT1   000002   BIT8   000400   CCTLR  105030-R  CSR.ME 105156-R  DBM108 006474-R  DBM18  005316-R
BAL.IN 105136-R  BIT10  002000   BIT9   001000   CDISK  105032-R  CST    077120-R  DBM109 006564-R  DBM19  005370-R
BIT0   000001    BIT11  004000   BL$DIV 076720-R  CER.01 017424-R  CST.AD 077246-R  DBM111 006644-R  DBM20  005454-R
BIT00  000001    BIT12  010000   BL$LAS 107202-R  CER.02 017470-R  CTLR.C 105036-R  DBM112 006744-R  DBM21  005530-R
BIT01  000002    BIT13  020000   BL$MOD 076732-R  CLK.PR 105114-R  CTLR.I 044434-R  DBM12  005212-R  DBM22  005612-R
BIT02  000004    BIT14  040000   BL$MUL 076474-R  CLK.TI 105106-R  CUOFF  105034-R  DBM120 007046-R  DBM23  005656-R
BIT03  000010    BIT15  100000   BL$SHF 076744-R  CMD.TI 105064-R  C.ERR. 100722-R  DBM121 007140-R  DBM25  005714-R
BIT04  000020    BIT2   000004   BOE    000400   CNTR.E 024734-R  DASH   026030-R  DBM123 007230-R  DBM26  005762-R
BIT05  000040    BIT3   000010   BRLEVE 105172-R  CREDIT 105100-R  DATAGM 073640-R  DBM125 007272-R  DBM27  006014-R
```

H7


```
DBM28A 006066-R  EGD.21 012502-R  EX.BDW 016722-R  GP$2    [illegible]564-R  IN.IOD 035260-R  L$NDSW 0261[illegible]2-R  PTCH3  002262-R
DBM28B 006126-R  EGD.22 012614-R  EX.CBC 016310-R  GP$26   [illegible]572-R  IODQ   105010-R  L$PRIO 002042-R  PTCH4  002334-R
DBM29  006166-R  EGD.23 012654-R  EX.CBR 016354-R  GP$27   026604-R  IODQ.I 105020-R  L$PROT 026164-R  PTCH5  002406-R
DBM32  006234-R  EGD.24 012716-R  EX.CBW 016426-R  GP$28   026620-R  IODQ.O 105022-R  L$PRT  002112-R  PUTA.B 035162-R
DBM5   005164-R  EGH.30 012762-R  EX.CMD 015526-R  GP$29   026630-R  IO.RET 064274-R  L$REPP 002062-R  PUT.IO 035116-R
DCT    077250-R  EGS.01 011746-R  EX.CMP 015566-R  GP$3    026256-R  IPKT.A 102434-R  L$REV  002010-R  PUT.PK 034560-R
DCT.AD 077272-R  EGS.02 011766-R  EX.LB  017146-R  GP$30   026644-R  IRDRX. 077276-R  L$RPT  030036-R  PUT.RE 035026-R
DFPTBL 026046-R  EH.0   013440-R  EX.LBN 016032-R  GP$31   026662-R  ISR    000100    L$SFIL 026430-R  P.INDE 105166-R
DF.MSG 025472-R  EH.1   013476-R  EX.LBR 016132-R  GP$32   026674-R  IXE    004000    L$SOFT 026432-R  QIO    105044-R
DIO.RE 063670-R  EH.10  014052-R  EX.LBW 016200-R  GP$33   026712-R  LOE    040000    L$SPC  002056-R  QIO.FU 056264-R
DISK.R 072216-R  EH.12  014102-R  EX.ONL 015602-R  GP$34   026720-R  LOI    000010    L$SPCP 002020-R  QIO.GE 053672-R
DRIVER 044102-R  EH.13  014136-R  EX.OP  015626-R  GP$4    026270-R  L$ACP  002110-R  L$SPTP 002024-R  QIO.LB 062230-R
DROP.C 035346-R  EH.2   013534-R  EX.PBN 016072-R  GP$5    026302-R  L$APT  002036-R  L$STA  002030-R  QIO.OK 053412-R
DRV.CT 035454-R  EH.3   013574-R  EX.RBN 016250-R  GP$6    026312-R  L$AU   033614-R  L$SW   026076-R  QIO.OU 053620-R
DR.ERR 051630-R  EH.4   013632-R  EX.RD  015546-R  GP$7    026322-R  L$AUT  002070-R  L$SWLE 026074-R  QIO.SI 063032-R
DR.RET 070562-R  EH.5   013656-R  EX.RP  017012-R  GP$8    026332-R  L$AUTO 032616-R  L$TEST 002114-R  QIO.UN 054660-R
DUP    057152-R  EH.6   013706-R  EX.SA  015432-R  GP$9    026344-R  L$CCP  002106-R  L$TIML 002014-R  RANDOM 100662-R
DUPCOM 061636-R  EH.7   013736-R  EX.SB  015504-R  HARD.E  064670-R  L$CLEA 033116-R  L$UNIT 002012-R  RANDY  054402-R
DUPIDL 062004-R  EH.8   013766-R  EX.SBO 015500-R  HARD.I  046112-R  L$CO   002032-R  MAN.TS 026156-R  RDM.CN 100660-R
DUPPKT 077652-R  EH.9   014022-R  EX.SC  015450-R  HOE     100000    L$DEPO 002011-R  MD.INI 053046-R  RDRX.A 077274-R
DUPRED 060574-R  ELG.FM 015416-R  EX.TIM 017110-R  HOE.FL  105115-R  L$DESC 026212-R  MINUTE 105104-R  RDRX.E 025454-R
DUPROU 026154-R  ELG.00 015002-R  EX.WRD 017100-R  HOST.W  067756-R  L$DESP 002076-R  MODULA 036206-R  RD.COU 105170-R
DUPWRT 057516-R  ELOG.P 103226-R  EX.WRT 015556-R  HOURS   105103-R  L$DEVP 002060-R  MSCP.P 100724-R  REG.EX 045062-R
DUP.CO 064176-R  EMS.BL 040256-R  FATAL. 071406-R  HRD.MS  025570-R  L$DISP 002124-R  MSG.01 010334-R  RETPKT 102452-R
DUP.FL 105026-R  EMS.CM 041202-R  FER.BC 105200-R  HRD.SU  025766-R  L$DLY  002116-R  MSG.02 010366-R  ROUND. 067412-R
DUP.RS 071114-R  EMS.DB 040060-R  FER.LB 105176-R  HWPTE0  026062-R  L$DTP  002040-R  MSG.03 010422-R  RPS.RE 070374-R
DUR    105040-R  EMS.ER 041570-R  FERO.L 105110-R  HWPTE1  026064-R  L$DTYP 002034-R  MULTI. 052402-R  RPT1   010454-R
DU.MSG 007566-R  EMS.R1 041136-R  FER1.L 105112-R  HWPTS0  026056-R  L$DU   033522-R  NAME.H 026070-R  RPT10  011164-R
DU.RSN 010306-R  EMS.R2 041006-R  FILL.B 063242-R  HWPTS1  026060-R  L$DUT  002072-R  NAME.L 026066-R  RPT11  011252-R
D$PCNT 002122-R  EMS.01 042000-R  FORCED 105175-R  HWPT.B  026052-R  L$DVTY 026172-R  NEX    105066-R  RPT12  011320-R
D.FAIL 105174-R  EMS.10 042036-R  FORCE. 105146-R  HWPT.D  026054-R  L$E    002052-R  NEXT.P 105102-R  RPT13  011366-R
EBD.10 013050-R  EMS.12 042100-R  FREE.M 105046-R  HWPT.I  026046-R  L$ENVT 002044-R  NEX.TR 033624-R  RPT14  011466-R
EBD.12 013110-R  EMS.13 042136-R  FSET.U 064544-R  HWPT.V  026050-R  L$ERRT 002126-R  NULL   005162-R  RPT15  011564-R
EBD.13 013156-R  EMS.14 042200-R  GET.IO 035042-R  HWQ1    002460-R  L$ETP  002102-R  OFF    000002    RPT16  011664-R
EBD.14 013210-R  EMS.18 042242-R  GET.PK 034174-R  HWQ10   003260-R  L$EXP1 002046-R  OF.RC  105060-R  RPT2   010540-R
EBD.18 013250-R  EMS.21 042310-R  GET.RA 054272-R  HWQ11   003310-R  L$EXP4 002064-R  ON     000001    RPT3   010604-R
EBD.19 013304-R  EMS.22 042334-R  GET.RE 034720-R  HWQ2    002474-R  L$EXP5 002066-R  OUT.IO 035222-R  RPT4   010670-R
EBD.24 013364-R  EMS.24 042352-R  GP$DIS 026600-R  HWQ3    002504-R  L$HARD 026236-R  OVF.CH 067336-R  RPT5   010734-R
EBS.01 013006-R  EMS.30 042420-R  GP$1   026236-R  HWQ4    002546-R  L$HIME 002120-R  PARITY 033634-R  RPT6   011022-R
EF.CON 000036    ENTRY. 105024-R  GP$10  026356-R  HWQ5    002564-R  L$HPCP 002016-R  PKT.US 102436-R  RPT7   011066-R
EF.NEW 000035    EOP.FL 105025-R  GP$11  026372-R  HWQ6A   002634-R  L$HPTP 002022-R  PNT    001000    RPT8   011104-R
EF.PWR 000034    ERRBLK 002134-R  GP$12  026406-R  HWQ6B   002706-R  L$HRDL 02[illegible]234-R  POLL.C 071640-R  RPT9   011132-R
EF.RES 000037    ERRMSG 002132-R  GP$13  026416-R  HWQ7A   002762-R  L$HW   026046-R  POLL.R 071740-R  RP.ADD 103224-R
EF.STA 000040    ERRNBR 002130-R  GP$14  026432-R  HWQ7B   003032-R  L$HWLE 026044-R  PRI    002000    RP.IND 103222-R
EGD.10 012060-R  ERRTYP 002126-R  GP$15  026444-R  HWQ8    003102-R  L$ICP  002104-R  PRI00  000000    RP.USE 103212-R
EGD.11 012120-R  ERR.CO 014744-R  GP$16  026456-R  HWQ9    003160-R  L$INIT 032604-R  PRI01  000040    SA.REG 105062-R
EGD.12 012144-R  ERR.00 014210-R  GP$17  026470-R  IBE     010000    L$LADP 002026-R  PRI02  000100    SB.COD 105054-R
EGD.13 012172-R  EVL    000004    GP$18  026502-R  IDU     000040    L$LAST 107206-R  PRI03  000140    SC.CLK 022672-R
EGD.14 012220-R  EX.ACC 015614-R  GP$19  026510-R  IER     020000    L$LOAD 002100-R  PRI04  000200    SC.CON 017570-R
EGD.15 012250-R  EX.BBU 015752-R  GP$2   026246-R  INIT.I  053206-R  L$LUN  002074-R  PRI05  000240    SC.CTO 022134-R
EGD.16 012266-R  EX.BB1 015700-R  GP$20  026516-R  INIT.O  105202-R  L$MREV 002050-R  PRI06  000300    SC.DIS 020166-R
EGD.17 012316-R  EX.BB2 015632-R  GP$21  026524-R  INIT.T  043744-R  L$NAME 002000-R  PRI07  000340    SC.DST 020576-R
EGD.18 012334-R  EX.BC  016500-R  GP$22  026532-R  INI.CT  044736-R  L$NDHR 026426-R  PROC.R 063474-R  SC.DS2 020652-R
EGD.19 012354-R  EX.BD  016554-R  GP$23  026540-R  INI.RR  046762-R  L$NDHW 026072-R  PTCH1  002136-R  SC.DUP 017612-R
EGD.20 012414-R  EX.BDR 016632-R  GP$24  026554-R  INT.GE  045622-R  L$NDSF 026726-R  PTCH2  002210-R  SC.ECC 020724-R
```

17


```
SC.ECD 021006-R  SC.INR 017704-R  SC.RSP 022720-R  SFPTBL 026076-R  SWQ11  003660-R  SWQ7   003466-R  UPD.IO 066470-R
SC.EC1 021256-R  SC.INV 017736-R  SC.SAF 021726-R  SFT.MS 025666-R  SWQ12  003712-R  SWQ9   003540-R  VEC.BR 045274-R
SC.EC2 021306-R  SC.IOP 020114-R  SC.SDI 017544-R  SPACE4 026020-R  SWQ13  004010-R  S.DUPP 105164-R  WAIT   036170-R
SC.EC3 021336-R  SC.ISH 020436-R  SC.SDS 022206-R  STEP   105056-R  SWQ14  004066-R  S.PATT 105162-R  XX13   017212-R
SC.EC4 021370-R  SC.IS2 020516-R  SC.SON 017664-R  ST.COD 105052-R  SWQ15  004140-R  TALLY  077320-R  XX23   017234-R
SC.EC5 021420-R  SC.NBB 023142-R  SC.SRI 022466-R  SWEEP  070242-R  SWQ17  004210-R  TEMP1  105074-R  XX32   017270-R
SC.EC6 021450-R  SC.NXM 022044-R  SC.SRT 022374-R  SWM1   004770-R  SWQ19  004306-R  TEMP2  105076-R  XX33   017316-R
SC.EC7 021500-R  SC.ODA 021772-R  SC.SUR 022766-R  SWP.DP 026104-R  SWQ2   003364-R  TIME   033716-R  XX34   017354-R
SC.EC8 021532-R  SC.ODB 022022-R  SC.SWP 021626-R  SWP.ER 026076-R  SWQ20  004376-R  TRK.SG 100654-R  $END.L 107330-R
SC.EC9 021564-R  SC.ONL 017642-R  SC.UNK 017754-R  SWP.FL 026102-R  SWQ21  004464-R  TST.PA 026160-R  $SAVE2 077012-R
SC.EDC 022264-R  SC.PAR 022100-R  SC.VOL 020034-R  SWP.RA 026106-R  SWQ22  004550-R  TYPER  105116-R  $SAVE3 077026-R
SC.FCT 021210-R  SC.POE 022554-R  SC.576 021134-R  SWP.TI 026110-R  SWQ23  004610-R  TYPEW  105126-R  $SAVE4 077044-R
SC.FER 020260-R  SC.PSP 023016-R  SEND   035532-R  SWP.UC 026112-R  SWQ24  004662-R  T$FREE 107326-R  $SAVE5 077064-R
SC.FE2 020346-R  SC.RCT 021040-R  SEQUEN 072316-R  SWP.UD 026114-R  SWQ26  004726-R  T$PTHV 000004
SC.FUL 021060-R  SC.RDY 022610-R  SET.CP 034000-R  SWP.XF 026100-R  SWQ27  005066-R  T.ADDR 077650-R
SC.HWP 021666-R  SC.REF 023212-R  SET.CT 047064-R  SWQI   003342-R  SWQ28  005112-R  T1     043730-R
SC.IDS 022324-R  SC.REP 023100-R  SET.UP 034074-R  SWQI0  003614-R  SWQ4   003444-R  UAM    000200
```

*** Task builder statistics:

Total work file references: 163811.
Work file reads: 0.
Work file writes: 0.
Size of core pool: 23176. words (90. pages)
Size of work file: 5120. words (20. pages)

Elapsed time:00:00:26

ZRQDAG CREATED BY TKB ON 3-JAN-86 AT 09:24 PAGE 1 SEQ 0501

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...
ADDR.V	105203-R	# ZRQDM1 ZRQDM2 ZRQDM3
ADR	000020	# ZRQDM1 # ZRQDM2 # ZRQDM3
ASTERI	026036-R	# ZRQDM1 ZRQDM2
AZINT	070714-R	# ZRQDM3
AZINTO	070676-R	# ZRQDM3
BAL.IN	105136-R	# ZRQDM1 ZRQDM2 ZRQDM3
BIT0	000001	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT00	000001	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT01	000002	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT02	000004	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT03	000010	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT04	000020	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT05	000040	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT06	000100	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT07	000200	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT08	000400	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT09	001000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT1	000002	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT10	002000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT11	004000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT12	010000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT13	020000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT14	040000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT15	100000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT2	000004	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT3	000010	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT4	000020	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT5	000040	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT6	000100	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT7	000200	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT8	000400	# ZRQDM1 # ZRQDM2 # ZRQDM3
BIT9	001000	# ZRQDM1 # ZRQDM2 # ZRQDM3
BL$DIV	076720-R	# B16MUL ZRQDM2 ZRQDM3
BL$LAS	107202-R	# ZRQDM4
BL$MOD	076732-R	# B16MUL ZRQDM2 ZRQDM3
BL$MUL	076474-R	# B16MUL ZRQDM2 ZRQDM3
BL$SHF	076744-R	# B16MUL ZRQDM3
BOE	000400	# ZRQDM1 # ZRQDM2 # ZRQDM3
BRLEVE	105172-R	# ZRQDM1 ZRQDM2 ZRQDM3
BST	077300-R	# ZRQDM1 ZRQDM2 ZRQDM3
BUFF.A	104760-R	# ZRQDM1 ZRQDM2 ZRQDM3
BUFF.O	105000-R	# ZRQDM1 ZRQDM2 ZRQDM3
BYTS.P	105050-R	# ZRQDM1 ZRQDM2 ZRQDM3
CCTLR	105030-R	# ZRQDM1 ZRQDM2 ZRQDM3
CDISK	105032-R	# ZRQDM1 ZRQDM2 ZRQDM3
CER.01	017424-R	# ZRQDM1 ZRQDM2
CER.02	017470-R	# ZRQDM1 ZRQDM2
CLK.PR	105114-R	# ZRQDM1 ZRQDM2 ZRQDM3
CLK.TI	105106-R	# ZRQDM1 ZRQDM2 ZRQDM3
CMD.TI	105064-R	# ZRQDM1 ZRQDM2 ZRQDM3
CNTR.E	024734-R	# ZRQDM1 ZRQDM2
CREDIT	105100-R	# ZRQDM1 ZRQDM2 ZRQDM3

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
CRLF	026024-R	# ZRQDM1	ZRQDM2	ZRQDM3
CRN.HI	105072-R	# ZRQDM1	ZRQDM2	ZRQDM3
CRN.LO	105070-R	# ZRQDM1	ZRQDM2	ZRQDM3
CSR.AD	105160-R	# ZRQDM1	ZRQDM2	ZRQDM3
CSR.ME	105156-R	# ZRQDM1	ZRQDM2	ZRQDM3
CST	077120-R	# ZRQDM1	ZRQDM2	ZRQDM3
CST.AD	077246-R	# ZRQDM1	ZRQDM2	ZRQDM3
CTLR.C	105036-R	# ZRQDM1	ZRQDM2	ZRQDM3
CTLR.I	044434-R	# ZRQDM3		
CUOFF	105034-R	# ZRQDM1	ZRQDM2	ZRQDM3
C.ERR.	100722-R	# ZRQDM1	ZRQDM2	ZRQDM3
DASH	026030-R	# ZRQDM1	ZRQDM2	
DATAGM	073640-R	# ZRQDM3		
DBM101	006310-R	# ZRQDM1		
DBM104	006336-R	# ZRQDM1		
DBM105	006400-R	# ZRQDM1		
DBM107	006436-R	# ZRQDM1	ZRQDM2	
DBM108	006474-R	# ZRQDM1	ZRQDM3	
DBM109	006564-R	# ZRQDM1	ZRQDM3	
DBM111	006644-R	# ZRQDM1	ZRQDM3	
DBM112	006744-R	# ZRQDM1	ZRQDM3	
DBM12	005212-R	# ZRQDM1	ZRQDM3	
DBM120	007046-R	# ZRQDM1	ZRQDM3	
DBM121	007140-R	# ZRQDM1	ZRQDM3	
DBM123	007230-R	# ZRQDM1	ZRQDM3	
DBM125	007272-R	# ZRQDM1	ZRQDM2	ZRQDM3
DBM126	007342-R	# ZRQDM1	ZRQDM2	ZRQDM3
DBM127	007422-R	# ZRQDM1	ZRQDM3	
DBM128	007500-R	# ZRQDM1	ZRQDM3	
DBM15	005266-R	# ZRQDM1		
DBM18	005316-R	# ZRQDM1	ZRQDM3	
DBM19	005370-R	# ZRQDM1	ZRQDM3	
DBM20	005454-R	# ZRQDM1	ZRQDM3	
DBM21	005530-R	# ZRQDM1	ZRQDM3	
DBM22	005612-R	# ZRQDM1	ZRQDM3	
DBM23	005656-R	# ZRQDM1	ZRQDM3	
DBM25	005714-R	# ZRQDM1	ZRQDM3	
DBM26	005762-R	# ZRQDM1	ZRQDM3	
DBM27	006014-R	# ZRQDM1	ZRQDM3	
DBM28A	006066-R	# ZRQDM1		
DBM28B	006126-R	# ZRQDM1		
DBM29	006166-R	# ZRQDM1	ZRQDM3	
DBM32	006234-R	# ZRQDM1		
DBM5	005164-R	# ZRQDM1	ZRQDM2	
DCT	077250-R	# ZRQDM1	ZRQDM2	ZRQDM3
DCT.AD	077272-R	# ZRQDM1	ZRQDM2	ZRQDM3
DFPTBL	026046-R	# ZRQDM1		
DF.MSG	025472-R	# ZRQDM1	ZRQDM3	
DIO.RE	063670-R	# ZRQDM3		
DISK.R	072216-R	# ZRQDM3		
DRIVER	044102-R	# ZRQDM3		
DROP.C	035346-R	# ZRQDM2	ZRQDM3	

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
DRV.CT	035454-R	# ZRQDM2	ZRQDM3	
DR.ERR	051630-R	# ZRQDM3		
DR.RET	070562-R	# ZRQDM3		
DUP	057152-R	# ZRQDM3		
DUPCOM	061636-R	# ZRQDM3		
DUPIDL	062004-R	# ZRQDM3		
DUPPKT	077652-R	# ZRQDM1	ZRQDM2	ZRQDM3
DUPRED	060574-R	# ZRQDM3		
DUPROU	026154-R	# ZRQDM1	ZRQDM3	
DUPWRT	057516-R	# ZRQDM3		
DUP.CO	064176-R	# ZRQDM3		
DUP.FL	105026-R	# ZRQDM1	ZRQDM2	ZRQDM3
DUP.RS	071114-R	# ZRQDM3		
DUR	105040-R	# ZRQDM1	ZRQDM2	ZRQDM3
DU.MSG	007566-R	# ZRQDM1	ZRQDM2	
DU.RSN	010306-R	# ZRQDM1	ZRQDM2	
D$PCNT	002122-R	# ZRQDM1		
D.FAIL	105174-R	# ZRQDM1	ZRQDM2	ZRQDM3
EBD.10	013050-R	# ZRQDM1	ZRQDM2	
EBD.12	013110-R	# ZRQDM1	ZRQDM2	
EBD.13	013156-R	# ZRQDM1	ZRQDM2	
EBD.14	013210-R	# ZRQDM1	ZRQDM2	
EBD.18	013250-R	# ZRQDM1	ZRQDM2	
EBD.19	013304-R	# ZRQDM1	ZRQDM2	
EBD.24	013364-R	# ZRQDM1	ZRQDM2	
EBS.01	013006-R	# ZRQDM1	ZRQDM2	
EF.CON	000036	# ZRQDM1	# ZRQDM2	# ZRQDM3
EF.NEW	000035	# ZRQDM1	# ZRQDM2	# ZRQDM3
EF.PWR	000034	# ZRQDM1	# ZRQDM2	# ZRQDM3
EF.RES	000037	# ZRQDM1	# ZRQDM2	# ZRQDM3
EF.STA	000040	# ZRQDM1	# ZRQDM2	# ZRQDM3
EGD.10	012060-R	# ZRQDM1	ZRQDM3	
EGD.11	012120-R	# ZRQDM1	ZRQDM3	
EGD.12	012144-R	# ZRQDM1	ZRQDM3	
EGD.13	012172-R	# ZRQDM1	ZRQDM3	
EGD.14	012220-R	# ZRQDM1	ZRQDM3	
EGD.15	012250-R	# ZRQDM1	ZRQDM3	
EGD.16	012266-R	# ZRQDM1	ZRQDM3	
EGD.17	012316-R	# ZRQDM1	ZRQDM3	
EGD.18	012334-R	# ZRQDM1	ZRQDM3	
EGD.19	012354-R	# ZRQDM1	ZRQDM3	
EGD.20	012414-R	# ZRQDM1	ZRQDM3	
EGD.21	012502-R	# ZRQDM1	ZRQDM3	
EGD.22	012614-R	# ZRQDM1	ZRQDM3	
EGD.23	012654-R	# ZRQDM1	ZRQDM3	
EGD.24	012716-R	# ZRQDM1	ZRQDM3	
EGH.30	012762-R	# ZRQDM1	ZRQDM3	
EGS.01	011746-R	# ZRQDM1	ZRQDM2	
EGS.02	011766-R	# ZRQDM1	ZRQDM3	
EH.0	013440-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.1	013476-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.10	014052-R	# ZRQDM1	ZRQDM2	ZRQDM3

M7

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
EH.12	014102-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.13	014136-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.2	013534-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.3	013574-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.4	013632-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.5	013656-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.6	013706-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.7	013736-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.8	013766-R	# ZRQDM1	ZRQDM2	ZRQDM3
EH.9	014022-R	# ZRQDM1	ZRQDM2	ZRQDM3
ELG.FM	015416-R	# ZRQDM1	ZRQDM2	
ELG.00	015002-R	# ZRQDM1	ZRQDM2	
ELOG.P	103226-R	# ZRQDM1	ZRQDM2	ZRQDM3
EMS.BL	040256-R	# ZRQDM2		
EMS.CM	041202-R	# ZRQDM2	ZRQDM3	
EMS.DB	040060-R	# ZRQDM2		
EMS.ER	041570-R	# ZRQDM2	ZRQDM3	
EMS.R1	041136-R	# ZRQDM2	ZRQDM3	
EMS.R2	041006-R	# ZRQDM2	ZRQDM3	
EMS.01	042000-R	# ZRQDM2		
EMS.10	042036-R	# ZRQDM2	ZRQDM3	
EMS.12	042100-R	# ZRQDM2	ZRQDM3	
EMS.13	042136-R	# ZRQDM2	ZRQDM3	
EMS.14	042200-R	# ZRQDM2	ZRQDM3	
EMS.18	042242-R	# ZRQDM2	ZRQDM3	
EMS.21	042310-R	# ZRQDM2	ZRQDM3	
EMS.22	042334-R	# ZRQDM2	ZRQDM3	
EMS.24	042352-R	# ZRQDM2	ZRQDM3	
EMS.30	042420-R	# ZRQDM2	ZRQDM3	
ENTRY.	105024-R	# ZRQDM1	ZRQDM2	ZRQDM3
EOP.FL	105025-R	# ZRQDM1	ZRQDM2	ZRQDM3
ERRBLK	002134-R	# ZRQDM1		
ERRMSG	002132-R	# ZRQDM1		
ERRNBR	002130-R	# ZRQDM1		
ERRTYP	002126-R	# ZRQDM1		
ERR.CO	014744-R	# ZRQDM1	ZRQDM2	
ERR.00	014210-R	# ZRQDM1	ZRQDM2	
EVL	000004	# ZRQDM1	# ZRQDM2	# ZRQDM3
EX.ACC	015614-R	# ZRQDM1	ZRQDM2	
EX.BBU	015752-R	# ZRQDM1	ZRQDM2	
EX.BB1	015700-R	# ZRQDM1	ZRQDM2	
EX.BB2	015632-R	# ZRQDM1	ZRQDM2	
EX.BC	016500-R	# ZRQDM1	ZRQDM2	
EX.BD	016554-R	# ZRQDM1	ZRQDM2	
EX.BDR	016632-R	# ZRQDM1	ZRQDM2	
EX.BDW	016722-R	# ZRQDM1	ZRQDM2	
EX.CBC	016310-R	# ZRQDM1	ZRQDM2	
EX.CBR	016354-R	# ZRQDM1	ZRQDM2	
EX.CBW	016426-R	# ZRQDM1	ZRQDM2	
EX.CMD	015526-R	# ZRQDM1	ZRQDM2	
EX.CMP	015566-R	# ZRQDM1	ZRQDM2	
EX.LB	017146-R	# ZRQDM1	ZRQDM2	

```
SYMBOL  VALUE      REFERENCES...

EX.LBN  016032-R  # ZRQDM1    ZRQDM2
EX.LBR  016132-R  # ZRQDM1    ZRQDM2
EX.LBW  016200-R  # ZRQDM1    ZRQDM2
EX.ONL  015602-R  # ZRQDM1    ZRQDM2
EX.OP   015626-R  # ZRQDM1    ZRQDM2
EX.PBN  016072-R  # ZRQDM1    ZRQDM2
EX.RBN  016250-R  # ZRQDM1    ZRQDM2
EX.RD   015546-R  # ZRQDM1    ZRQDM2
EX.RP   017012-R  # ZRQDM1    ZRQDM2
EX.SA   015432-R  # ZRQDM1    ZRQDM2
EX.SB   015504-R  # ZRQDM1    ZRQDM2
EX.SBO  015500-R  # ZRQDM1    ZRQDM2
EX.SC   015450-R  # ZRQDM1    ZRQDM2
EX.TIM  017110-R  # ZRQDM1    ZRQDM2
EX.WRD  017100-R  # ZRQDM1    ZRQDM2
EX.WRT  015556-R  # ZRQDM1    ZRQDM2
FATAL.  071406-R  # ZRQDM3
FER.BC  105200-R  # ZRQDM1    ZRQDM2    ZRQDM3
FER.LB  105176-R  # ZRQDM1    ZRQDM2
FER0.L  105110-R  # ZRQDM1    ZRQDM2    ZRQDM3
FER1.L  105112-R  # ZRQDM1    ZRQDM2    ZRQDM3
FILL.B  063242-R  # ZRQDM3
FORCED  105175-R  # ZRQDM1    ZRQDM2    ZRQDM3
FORCE.  105146-R  # ZRQDM1    ZRQDM2    ZRQDM3
FREE.M  105046-R  # ZRQDM1    ZRQDM2    ZRQDM3
FSET.U  064544-R  # ZRQDM3
GET.IO  035042-R  # ZRQDM2    ZRQDM3
GET.PK  034174-R  # ZRQDM2    ZRQDM3
GET.RA  054272-R  # ZRQDM3
GET.RE  034720-R  # ZRQDM2    ZRQDM3
GP$DIS  026600-R  # ZRQDM2
GP$1    026236-R  # ZRQDM2
GP$10   026356-R  # ZRQDM2
GP$11   026372-R  # ZRQDM2
GP$12   026406-R  # ZRQDM2
GP$13   026416-R  # ZRQDM2
GP$14   026432-R  # ZRQDM2
GP$15   026444-R  # ZRQDM2
GP$16   026456-R  # ZRQDM2
GP$17   026470-R  # ZRQDM2
GP$18   026502-R  # ZRQDM2
GP$19   026510-R  # ZRQDM2
GP$2    026246-R  # ZRQDM2
GP$20   026516-R  # ZRQDM2
GP$21   026524-R  # ZRQDM2
GP$22   026532-R  # ZRQDM2
GP$23   026540-R  # ZRQDM2
GP$24   026554-R  # ZRQDM2
GP$25   026564-R  # ZRQDM2
GP$26   026572-R  # ZRQDM2
GP$27   026604-R  # ZRQDM2
GP$28   026620-R  # ZRQDM2
```

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
GP$29	026630-R	# ZRQDM2		
GP$3	026256-R	# ZRQDM2		
GP$30	026644-R	# ZRQDM2		
GP$31	026662-R	# ZRQDM2		
GP$32	026674-R	# ZRQDM2		
GP$33	026712-R	# ZRQDM2		
GP$34	026720-R	# ZRQDM2		
GP$4	026270-R	# ZRQDM2		
GP$5	026302-R	# ZRQDM2		
GP$6	026312-R	# ZRQDM2		
GP$7	026322-R	# ZRQDM2		
GP$8	026332-R	# ZRQDM2		
GP$9	026344-R	# ZRQDM2		
HARD.E	064670-R	# ZRQDM3		
HARD.I	046112-R	# ZRQDM3		
HOE	100000	# ZRQDM1	# ZRQDM2	# ZRQDM3
HOE.FL	105115-R	# ZRQDM1	ZRQDM2	ZRQDM3
HOST.W	067756-R	# ZRQDM3		
HOURS	105103-R	# ZRQDM1	ZRQDM2	ZRQDM3
HRD.MS	025570-R	# ZRQDM1	ZRQDM3	
HRD.SU	025766-R	# ZRQDM1	ZRQDM3	
HWPTE0	026062-R	# ZRQDM1		
HWPTE1	026064-R	# ZRQDM1		
HWPTS0	026056-R	# ZRQDM1		
HWPTS1	026060-R	# ZRQDM1		
HWPT.B	026052-R	# ZRQDM1		
HWPT.D	026054-R	# ZRQDM1		
HWPT.I	026046-R	# ZRQDM1		
HWPT.V	026050-R	# ZRQDM1		
HWQ1	002460-R	# ZRQDM1	ZRQDM2	
HWQ10	003260-R	# ZRQDM1	ZRQDM2	
HWQ11	003310-R	# ZRQDM1	ZRQDM2	
HWQ2	002474-R	# ZRQDM1	ZRQDM2	
HWQ3	002504-R	# ZRQDM1	ZRQDM2	
HWQ4	002546-R	# ZRQDM1	ZRQDM2	
HWQ5	002564-R	# ZRQDM1	ZRQDM2	
HWQ6A	002634-R	# ZRQDM1	ZRQDM2	
HWQ6B	002706-R	# ZRQDM1	ZRQDM2	
HWQ7A	002762-R	# ZRQDM1	ZRQDM2	
HWQ7B	003032-R	# ZRQDM1	ZRQDM2	
HWQ8	003102-R	# ZRQDM1	ZRQDM2	
HWQ9	003160-R	# ZRQDM1	ZRQDM2	
IBE	010000	# ZRQDM1	# ZRQDM2	# ZRQDM3
IDU	000040	# ZRQDM1	# ZRQDM2	# ZRQDM3
IER	020000	# ZRQDM1	# ZRQDM2	# ZRQDM3
INIT.I	053206-R	# ZRQDM3		
INIT.O	105202-R	# ZRQDM1	ZRQDM2	ZRQDM3
INIT.T	043744-R	# ZRQDM3		
INI.CT	044736-R	# ZRQDM3		
INI.RR	046762-R	# ZRQDM3		
INT.GE	045622-R	# ZRQDM3		
IN.IOD	035260-R	# ZRQDM2	ZRQDM3	

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
IODQ	105010-R	# ZRQDM1	ZRQDM2	ZRQDM3
IODQ.I	105020-R	# ZRQDM1	ZRQDM2	ZRQDM3
IODQ.O	105022-R	# ZRQDM1	ZRQDM2	ZRQDM3
IO.RET	064274-R	# ZRQDM3		
IPKT.A	102434-R	# ZRQDM1	ZRQDM2	ZRQDM3
IRDRX.	077276-R	# ZRQDM1	ZRQDM2	ZRQDM3
ISR	000100	# ZRQDM1	# ZRQDM2	# ZRQDM3
IXE	004000	# ZRQDM1	# ZRQDM2	# ZRQDM3
LOE	040000	# ZRQDM1	# ZRQDM2	# ZRQDM3
LOT	000010	# ZRQDM1	# ZRQDM2	# ZRQDM3
L$ACP	002110-R	# ZRQDM1		
L$APT	002036-R	# ZRQDM1		
L$AU	033614-R	ZRQDM1	# ZRQDM2	
L$AUT	002070-R	# ZRQDM1		
L$AUTO	032616-R	ZRQDM1	# ZRQDM2	
L$CCP	002106-R	# ZRQDM1		
L$CLEA	033116-R	ZRQDM1	# ZRQDM2	
L$CO	002032-R	# ZRQDM1		
L$DEPO	002011-R	# ZRQDM1		
L$DESC	026212-R	ZRQDM1	# ZRQDM2	
L$DESP	002076-R	# ZRQDM1		
L$DEVP	002060-R	# ZRQDM1		
L$DISP	002124-R	# ZRQDM1		
L$DLY	002116-R	# ZRQDM1	ZRQDM2	ZRQDM3
L$DTP	002040-R	# ZRQDM1		
L$DTYP	002034-R	# ZRQDM1		
L$DU	033522-R	ZRQDM1	# ZRQDM2	
L$DUT	002072-R	# ZRQDM1		
L$DVTY	026172-R	ZRQDM1	# ZRQDM2	
L$EF	002052-R	# ZRQDM1		
L$ENVI	002044-R	# ZRQDM1		
L$ERRT	002126-R	# ZRQDM1		
L$ETP	002102-R	# ZRQDM1		
L$EXP1	002046-R	# ZRQDM1		
L$EXP4	002064-R	# ZRQDM1		
L$EXP5	002066-R	# ZRQDM1		
L$HARD	026236-R	ZRQDM1	# ZRQDM2	
L$HIME	002120-R	# ZRQDM1	ZRQDM2	
L$HPCP	002016-R	# ZRQDM1		
L$HPTP	002022-R	# ZRQDM1		
L$HRDL	026234-R	# ZRQDM2		
L$HW	026046-R	# ZRQDM1		
L$HWLE	026044-R	# ZRQDM1		
L$ICP	002104-R	# ZRQDM1		
L$INIT	032604-R	ZRQDM1	# ZRQDM2	
L$LADP	002026-R	# ZRQDM1		
L$LAST	107206-R	ZRQDM1	# ZRQDM4	
L$LOAD	002100-R	# ZRQDM1		
L$LUN	002074-R	# ZRQDM1	ZRQDM2	ZRQDM3
L$MREV	002050-R	# ZRQDM1		
L$NAME	002000-R	# ZRQDM1		
L$NDHR	026426-R	# ZRQDM2		

D8

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL VALUE REFERENCES...

```
L$NDHW  026072-R  # ZRQDM1
L$NDSF  026726-R  # ZRQDM2
L$NDSW  026162-R  # ZRQDM1
L$PRIO  002042-R  # ZRQDM1
L$PROT  026164-R  # ZRQDM1
L$PRT   002112-R  # ZRQDM1
L$REPP  002062-R  # ZRQDM1
L$REV   002010-R  # ZRQDM1
L$RPT   030036-R    ZRQDM1  # ZRQDM2
L$SFTL  026430-R  # ZRQDM2
L$SOFT  026432-R    ZRQDM1  # ZRQDM2
L$SPC   002056-R  # ZRQDM1
L$SPCP  002020-R  # ZRQDM1
L$SPTP  002024-R  # ZRQDM1
L$STA   002030-R  # ZRQDM1
L$SW    026076-R  # ZRQDM1
L$SWLE  026074-R  # ZRQDM1
L$TEST  002114-R  # ZRQDM1
L$TIML  002014-R  # ZRQDM1
L$UNIT  002012-R  # ZRQDM1    ZRQDM2    ZRQDM3
MAN.TS  026156-R  # ZRQDM1    ZRQDM3
MD.INI  053046-R  # ZRQDM3
MINUTE  105104-R  # ZRQDM1    ZRQDM2    ZRQDM3
MODULA  036206-R  # ZRQDM2    ZRQDM3
MSCP.P  100724-R  # ZRQDM1    ZRQDM2    ZRQDM3
MSG.01  010334-R  # ZRQDM1    ZRQDM2
MSG.02  010366-R  # ZRQDM1    ZRQDM3
MSG.03  010422-R  # ZRQDM1    ZRQDM3
MULTI.  052402-R  # ZRQDM3
NAME.H  026070-R  # ZRQDM1
NAME.L  026066-R  # ZRQDM1
NEX     105066-R  # ZRQDM1    ZRQDM2    ZRQDM3
NEXT.P  105102-R  # ZRQDM1    ZRQDM2    ZRQDM3
NEX.TR  033624-R  # ZRQDM2    ZRQDM3
NULL    005162-R  # ZRQDM1    ZRQDM2
OFF     000002    # ZRQDM1  # ZRQDM2  # ZRQDM3
OF.RC   105060-R  # ZRQDM1    ZRQDM2    ZRQDM3
ON      000001    # ZRQDM1  # ZRQDM2  # ZRQDM3
OUT.IO  035222-R  # ZRQDM2    ZRQDM3
OVF.CH  067336-R  # ZRQDM3
PARITY  033634-R  # ZRQDM2    ZRQDM3
PKT.US  102436-R  # ZRQDM1    ZRQDM2    ZRQDM3
PNT     001000    # ZRQDM1  # ZRQDM2  # ZRQDM3
POLL.C  071640-R  # ZRQDM3
POLL.R  071740-R  # ZRQDM3
PRI     002000    # ZRQDM1  # ZRQDM2  # ZRQDM3
PRI00   000000    # ZRQDM1  # ZRQDM2  # ZRQDM3
PRI01   000040    # ZRQDM1  # ZRQDM2  # ZRQDM3
PRI02   000100    # ZRQDM1  # ZRQDM2  # ZRQDM3
PRI03   000140    # ZRQDM1  # ZRQDM2  # ZRQDM3
PRI04   000200    # ZRQDM1  # ZRQDM2  # ZRQDM3
PRI05   000240    # ZRQDM1  # ZRQDM2  # ZRQDM3
```

E8

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...		
PRI06	000300	# ZRQDM1	# ZRQDM2	# ZRQDM3
PRI07	000340	# ZRQDM1	# ZRQDM2	# ZRQDM3
PROC.R	063474-R	# ZRQDM3		
PTCH1	002136-R	# ZRQDM1		
PTCH2	002210-R	# ZRQDM1		
PTCH3	002262-R	# ZRQDM1		
PTCH4	002334-R	# ZRQDM1		
PTCH5	002406-R	# ZRQDM1		
PUTA.B	035162-R	# ZRQDM2	ZRQDM3	
PUT.IO	035116-R	# ZRQDM2	ZRQDM3	
PUT.PK	034560-R	# ZRQDM2	ZRQDM3	
PUT.RE	035026-R	# ZRQDM2	ZRQDM3	
P.INDE	105166-R	# ZRQDM1	ZRQDM2	ZRQDM3
QIO	105044-R	# ZRQDM1	ZRQDM2	ZRQDM3
QIO.FU	056264-R	# ZRQDM3		
QIO.GE	053672-R	# ZRQDM3		
QIO.LB	062230-R	# ZRQDM3		
QIO.OK	053412-R	# ZRQDM3		
QIO.OU	053620-R	# ZRQDM3		
QIO.SI	063032-R	# ZRQDM3		
QIO.UN	054660-R	# ZRQDM3		
RANDOM	100662-R	# ZRQDM1	ZRQDM2	ZRQDM3
RANDY	054402-R	# ZRQDM3		
RDM.CN	100660-R	# ZRQDM1	ZRQDM2	ZRQDM3
RDRX.A	077274-R	# ZRQDM1	ZRQDM2	ZRQDM3
RDRX.E	025454-R	# ZRQDM1	ZRQDM2	
RD.COU	105170-R	# ZRQDM1	ZRQDM2	ZRQDM3
REG.EX	045062-R	# ZRQDM3		
RETPKT	102452-R	# ZRQDM1	ZRQDM2	ZRQDM3
ROUND.	067412-R	# ZRQDM3		
RPS.RE	070374-R	# ZRQDM3		
RPT1	010454-R	# ZRQDM1	ZRQDM2	
RPT10	011164-R	# ZRQDM1	ZRQDM2	
RPT11	011252-R	# ZRQDM1	ZRQDM2	
RPT12	011320-R	# ZRQDM1	ZRQDM2	
RPT13	011366-R	# ZRQDM1	ZRQDM2	
RPT14	011466-R	# ZRQDM1	ZRQDM2	
RPT15	011564-R	# ZRQDM1	ZRQDM2	
RPT16	011664-R	# ZRQDM1	ZRQDM2	
RPT2	010540-R	# ZRQDM1	ZRQDM2	
RPT3	010604-R	# ZRQDM1	ZRQDM2	
RPT4	010670-R	# ZRQDM1	ZRQDM2	
RPT5	010734-R	# ZRQDM1	ZRQDM2	
RPT6	011022-R	# ZRQDM1	ZRQDM2	
RPT7	011066-R	# ZRQDM1	ZRQDM2	
RPT8	011104-R	# ZRQDM1	ZRQDM2	
RPT9	011132-R	# ZRQDM1	ZRQDM2	
RP.ADD	103224-R	# ZRQDM1	ZRQDM2	ZRQDM3
RP.INO	103222-R	# ZRQDM1	ZRQDM2	ZRQDM3
RP.USE	103212-R	# ZRQDM1	ZRQDM2	ZRQDM3
SA.REG	105062-R	# ZRQDM1	ZRQDM2	ZRQDM3
SB.COD	105054-R	# ZRQDM1	ZRQDM2	ZRQDM3

F8

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...	
SC.CLK	022672-R	# ZRQDM1	ZRQDM2
SC.CON	017570-R	# ZRQDM1	ZRQDM2
SC.CTO	022134-R	# ZRQDM1	ZRQDM2
SC.DIS	020166-R	# ZRQDM1	ZRQDM2
SC.DST	020576-R	# ZRQDM1	ZRQDM2
SC.DS2	020652-R	# ZRQDM1	ZRQDM2
SC.DUP	017612-R	# ZRQDM1	ZRQDM2
SC.ECC	020724-R	# ZRQDM1	ZRQDM2
SC.ECD	021006-R	# ZRQDM1	ZRQDM2
SC.EC1	021256-R	# ZRQDM1	ZRQDM2
SC.EC2	021306-R	# ZRQDM1	ZRQDM2
SC.EC3	021336-R	# ZRQDM1	ZRQDM2
SC.EC4	021370-R	# ZRQDM1	ZRQDM2
SC.EC5	021420-R	# ZRQDM1	ZRQDM2
SC.EC6	021450-R	# ZRQDM1	ZRQDM2
SC.EC7	021500-R	# ZRQDM1	ZRQDM2
SC.EC8	021532-R	# ZRQDM1	ZRQDM2
SC.EC9	021564-R	# ZRQDM1	ZRQDM2
SC.EDC	022264-R	# ZRQDM1	ZRQDM2
SC.FCT	021210-R	# ZRQDM1	ZRQDM2
SC.FER	020260-R	# ZRQDM1	ZRQDM2
SC.FE2	020346-R	# ZRQDM1	ZRQDM2
SC.FUL	021060-R	# ZRQDM1	ZRQDM2
SC.HWP	021666-R	# ZRQDM1	ZRQDM2
SC.IDS	022324-R	# ZRQDM1	ZRQDM2
SC.INR	017704-R	# ZRQDM1	ZRQDM2
SC.INV	017736-R	# ZRQDM1	ZRQDM2
SC.IOP	020114-R	# ZRQDM1	ZRQDM2
SC.ISH	020436-R	# ZRQDM1	ZRQDM2
SC.IS2	020516-R	# ZRQDM1	ZRQDM2
SC.NBB	023142-R	# ZRQDM1	ZRQDM2
SC.NXM	022044-R	# ZRQDM1	ZRQDM2
SC.ODA	021772-R	# ZRQDM1	ZRQDM2
SC.ODB	022022-R	# ZRQDM1	ZRQDM2
SC.ONL	017642-R	# ZRQDM1	ZRQDM2
SC.PAR	022100-R	# ZRQDM1	ZRQDM2
SC.POE	022554-R	# ZRQDM1	ZRQDM2
SC.PSP	023016-R	# ZRQDM1	ZRQDM2
SC.RCT	021040-R	# ZRQDM1	ZRQDM2
SC.RDY	022610-R	# ZRQDM1	ZRQDM2
SC.REF	023212-R	# ZRQDM1	ZRQDM2
SC.REP	023100-R	# ZRQDM1	ZRQDM2
SC.RSP	022720-R	# ZRQDM1	ZRQDM2
SC.SAF	021726-R	# ZRQDM1	ZRQDM2
SC.SDI	017544-R	# ZRQDM1	ZRQDM2
SC.SDS	022206-R	# ZRQDM1	ZRQDM2
SC.SON	017664-R	# ZRQDM1	ZRQDM2
SC.SRI	022466-R	# ZRQDM1	ZRQDM2
SC.SRT	022374-R	# ZRQDM1	ZRQDM2
SC.SUR	022766-R	# ZRQDM1	ZRQDM2
SC.SWP	021626-R	# ZRQDM1	ZRQDM2
SC.UNK	017754-R	# ZRQDM1	ZRQDM2

G8

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE		REFERENCES...		
SC.VOL	020034-R	#	ZRQDM1	ZRQDM2	
SC.576	021134-R	#	ZRQDM1	ZRQDM2	
SEND	035532-R	#	ZRQDM2	ZRQDM3	
SEQUEN	072316-R	#	ZRQDM3		
SET.CP	034000-R	#	ZRQDM2	ZRQDM3	
SET.CT	047064-R	#	ZRQDM3		
SET.UP	034074-R	#	ZRQDM2	ZRQDM3	
SFPTBL	026076-R	#	ZRQDM1		
SFT.MS	025666-R	#	ZRQDM1	ZRQDM3	
SPACE4	026020-R	#	ZRQDM1	ZRQDM2	
STEP	105056-R	#	ZRQDM1	ZRQDM2	ZRQDM3
ST.COD	105052-R	#	ZRQDM1	ZRQDM2	ZRQDM3
SWEEP	070242-R	#	ZRQDM3		
SWM1	004770-R	#	ZRQDM1	ZRQDM2	
SWP.DP	026104-R	#	ZRQDM1	ZRQDM3	
SWP.ER	026076-R	#	ZRQDM1	ZRQDM3	
SWP.FL	026102-R	#	ZRQDM1	ZRQDM2	ZRQDM3
SWP.RA	026106-R	#	ZRQDM1	ZRQDM3	
SWP.TI	026110-R	#	ZRQDM1	ZRQDM3	
SWP.UC	026112-R	#	ZRQDM1	ZRQDM3	
SWP.UD	026114-R	#	ZRQDM1	ZRQDM3	
SWP.XF	026100-R	#	ZRQDM1	ZRQDM3	
SWQ1	003342-R	#	ZRQDM1	ZRQDM2	
SWQ10	003614-R	#	ZRQDM1	ZRQDM2	
SWQ11	003660-R	#	ZRQDM1	ZRQDM2	
SWQ12	003712-R	#	ZRQDM1	ZRQDM2	
SWQ13	004010-R	#	ZRQDM1	ZRQDM2	
SWQ14	004066-R	#	ZRQDM1	ZRQDM2	
SWQ15	004140-R	#	ZRQDM1	ZRQDM2	
SWQ17	004210-R	#	ZRQDM1	ZRQDM2	
SWQ19	004306-R	#	ZRQDM1	ZRQDM2	
SWQ2	003364-R	#	ZRQDM1	ZRQDM2	
SWQ20	004376-R	#	ZRQDM1	ZRQDM2	
SWQ21	004464-R	#	ZRQDM1	ZRQDM2	
SWQ22	004550-R	#	ZRQDM1	ZRQDM2	
SWQ23	004610-R	#	ZRQDM1	ZRQDM2	
SWQ24	004662-R	#	ZRQDM1	ZRQDM2	
SWQ26	004726-R	#	ZRQDM1	ZRQDM2	
SWQ27	005066-R	#	ZRQDM1	ZRQDM2	
SWQ28	005112-R	#	ZRQDM1	ZRQDM2	
SWQ4	003444-R	#	ZRQDM1	ZRQDM2	
SWQ7	003466-R	#	ZRQDM1	ZRQDM2	
SWQ9	003540-R	#	ZRQDM1	ZRQDM2	
S.DUPP	105164-R	#	ZRQDM1	ZRQDM2	ZRQDM3
S.PATT	105162-R	#	ZRQDM1	ZRQDM2	ZRQDM3
TALLY	077320-R	#	ZRQDM1	ZRQDM2	ZRQDM3
TEMP1	105074-R	#	ZRQDM1	ZRQDM2	ZRQDM3
TEMP2	105076-R	#	ZRQDM1	ZRQDM2	ZRQDM3
TIME	033716-R	#	ZRQDM2	ZRQDM3	
TRK.SG	100654-R	#	ZRQDM1	ZRQDM2	ZRQDM3
TST.PA	026160-R	#	ZRQDM1	ZRQDM3	
TYPER	105116-R	#	ZRQDM1	ZRQDM2	ZRQDM3

GLOBAL CROSS REFERENCE CREF 04.00

SYMBOL	VALUE	REFERENCES...				
TYPEW	105126-R	# ZRQDM1	ZRQDM2	ZRQDM3		
T$FREE	107326-R	# ZRQDM4				
T$PTHV	000004	ZRQDM1	# ZRQDM4			
T.ADDR	077650-R	# ZRQDM1	ZRQDM2	ZRQDM3		
T1	043730-R	ZRQDM1	# ZRQDM3			
UAM	000200	# ZRQDM1	# ZRQDM2	# ZRQDM3		
UPD.IO	066470-R	# ZRQDM3				
VEC.BR	045274-R	# ZRQDM3				
WAIT	036170-R	# ZRQDM2	ZRQDM3			
XX13	017212-R	# ZRQDM1	ZRQDM2			
XX23	017234-R	# ZRQDM1	ZRQDM2			
XX32	017270-R	# ZRQDM1	ZRQDM2			
XX33	017316-R	# ZRQDM1	ZRQDM2			
XX34	017354-R	# ZRQDM1	ZRQDM2			
$END.L	107330-R	# ZRQDM4				
$SAVE2	077012-R	B16MUL	# B16SAV	ZRQDM2	ZRQDM3	
$SAVE3	077026-R	# B16SAV	ZRQDM2	ZRQDM3		
$SAVE4	077044-R	# B16SAV	ZRQDM2	ZRQDM3		
$SAVE5	077064-R	B16MUL	# B16SAV	ZRQDM2	ZRQDM3	