```
.REM    &
                IDENTIFICATION

PRODUCT CODE:   AC-T400D-MC

PRODUCT NAME:   CXRQADO RQDX1 DEC/X MODULE

PRODUCT DATE:   9 APRIL 1984

MAINTAINER:     DIAGNOSTIC ENGINEERING

AUTHOR:         RICHARD DIETZ
```

```
                        .SBTTL  MODULE PRIVATE DATA
                        ;*****************************************************************************
                        ;*      MACRO DEFINITIONS
                        ;*****************************************************************************
                                .SBTTL  MACROS
                        ;
                        ; IN CASE NO SOB INSTRUCTION
                        ;
000001                  NOSOB   =       1
                        .MACRO  SOB     REG,TO
                                DEC     REG
                                BNE     TO
                        .ENDM

                        ;*****************************************************************************

                        .MACRO  PUSH    A,B,C,D,E
                                MOV     A, -(R6)
                        .IF     NB      B
                                MOV     B, -(R6)
                        .ENDC
                        .IF     NB      C
                                MOV     C, -(R6)
                        .ENDC
                        .IF     NB      D
                                MOV     D, -(R6)
                        .ENDC
                        .IF     NB      E
                                MOV     E, -(R6)
                        .ENDC
                        .ENDM

                        ;*****************************************************************************

                        .MACRO  POP     A,B,C,D,E
                                MOV     (R6)+, A
                        .IF     NB      B
                                MOV     (R6)+, B
                        .ENDC
                        .IF     NB      C
                                MOV     (R6)+, C
                        .ENDC
                        .IF     NB      D
                                MOV     (R6)+, D
                        .ENDC
                        .IF     NB      E
                                MOV     (R6)+, E
                        .ENDC
                        .ENDM

                        ;*****************************************************************************
```

```
                        .SBTTL  EQUATES

000001          BIT00 = 1
000002          BIT01 = 2
000004          BIT02 = 4
000010          BIT03 = 10
000020          BIT04 = 20
000040          BIT05 = 40
000100          BIT06 = 100
000200          BIT07 = 200
000400          BIT08 = 400
001000          BIT09 = 1000
002000          BIT10 = 2000
004000          BIT11 = 4000
010000          BIT12 = 10000
020000          BIT13 = 20000
040000          BIT14 = 40000
100000          BIT15 = 100000

000002          UPPER = 2

                ;
                ;       STATUS BITS
                ;
000001          SR.PAK = BIT00  ; PRINT OUT PACKET, 1 = YES
000002          SR.RON = BIT01  ; 0 MEANS READ ONLY
                                ; 1 MEANS WRITE, READ, CHECK
000004          SR.REP = BIT02  ; REPORT ERROR AS THEY OCCUR    0 = REPORT,
                                ;                               1 = DON'T REPORT
000010          SR.SUM = BIT03  ; REPORT ERRORS ON END OF PASS 0 = REPORT,
                                ;                              1 = DON'T REPORT
002000          SR.SEQ = BIT10  ; DISK ADDRESSING 0 = SEQUENTIAL,
                                ;                 1 = RANDOM
004000          SR.CMP = BIT11  ; DATA COMPARE    0 = DO DATA COMPARE,
                                ;                 1 = DON'T DO DATA COMPARE
                ;
                ;       ERROR BITS
                ;
000000          ERR.0 = 0       ; NOT DEFINED
000001          ERR.1 = 1       ; DATA ERROR
000003          ERR.3 = 3       ; CONTROLLER NOT READY
000004          ERR.4 = 4       ; BLOCK NOT FOUND
000006          ERR.6 = 6       ; DRIVE NOT READY, OFF LINE OR NON EXISTENT
000032          ERR.32 = 32     ; NPR ERROR
000034          ERR.34 = 34     ; UNABLE TO INITIALIZE ERROR
000036          ERR.36 = 36     ; UNABLE TO READ  ERROR
000037          ERR.37 = 37     ; UNABLE TO WRITE ERROR
000043          ERR.43 = 43     ; CRC  ERROR
000051          ERR.51 = 51     ; SEEK ERROR
```

RQDX BIT DEFINITIONS

```
                        .SBTTL RQDX BIT DEFINITIONS

                        ;RQDXSA REGISTER UNIVERSAL READ BITS

004000                          SA.S1= 004000           ;STEP 1 STATUS BIT
010000                          SA.S2= 010000           ;STEP 2 STATUS BIT
020000                          SA.S3= 020000           ;STEP 3 STATUS BIT
040000                          SA.S4= 040000           ;STEP 4 STATUS BIT
100000                          SA.ERR= 100000          ;ERROR INDICATOR

                        ;RQDXSA REGISTER ERROR STATUS BITS

003777                          SA.ERC= 003777          ;ERROR CODE

                        ;RQDXSA REGISTER STEP ONE READ BITS

003400                          SA.CTP= 003400          ;CONTROLLER TYPE
000400                          SA.DIA= 000400          ;DIAG BIT IN RQDXSA
001000                          SA.QB = 001000          ;SAME - DENOTES QBUS SUPPORT

                        ;RQDXSA REGISTER STEP ONE WRITE BITS

000177                          SA.VEC= 000177          ;INTERRUPT VECTOR (DIVIDED BY 4)
000200                          SA.INT= 000200          ;INTERRUPT ENABLE DURING INITIALIZATION
007400                          SA.RSP= 007400          ;RESPONCE RING LENGTH
170000                          SA.CMD= 170000          ;COMMAND RING LENGTH

                        ;RQDXSA REGISTER STEP TWO READ BITS

000177                          SA.VCE= 000177          ;INTERRUPT VECTOR ECHO
000200                          SA.INE= 000200          ;INTERRUPT ENABLE ECHO

                        ;RQDXSA REGISTER STEP TWO WRITE BITS

                                                        ;LOW ORDER MESSAGE RING BYTE ADDRESS
000001                          SA.PRG= 000001          ;ENABLE VAX UNIBUS ADAPTER PURGE INTERRUPT

                        ;RQDXSA REGISTER STEP THREE READ BITS

000017                          SA.RSE= 000017          ;RESPONCE RING LENGTH ECHO
000360                          SA.CME= 000360          ;COMMAND RING LENGTH ECHO

                        ;RQDXSA REGISTER STEP THREE WRITE BITS

                                                        ;HIGH ORDER MESSAGE RING BYTE ADDRESS
040000                          SA.LFC= 040000          ;LAST FAILURE CODE REQUEST

                        ;RQDXSA REGISTER STEP FOUR READ BITS

                        ;RQDXSA REGISTER STEP FOUR WRITE BITS

000001                          SA.GO= BIT0             ;GO BIT TO START RQDX FIRMWARE
```

COMMAND MESSAGE DESCRIPTOR BIT DEFINITIONS

```
                        .SBTTL COMMAND/MESSAGE DESCRIPTOR BIT DEFINITIONS

100000                          RG.OWN= BIT15           ;SET WHEN RQDX OWNS RING
040000                          RG.FLG= BIT14           ;FLAG BIT

                        ;OFFSETS INTO HOST COMMUNICATIONS AREA WITH ONE DESCRIPTOR TO EACH RING

000010                          HC.SIZ= 8.              ;SIZE OF HOST COMM AREA IN BYTES
000060                          PKTSIZ= 48.             ;SIZE OF PACKETS IN BYTES

000000                          HC.RES= 0.              ;RESPONCE RING START
000002                          HC.RCT= 2.              ;RESPONCE RING CONTROL WORD
000004                          HC.CMD= 4.              ;COMMAND RING START
000006                          HC.CCT= 6.              ;CONTROL RING CONTROL WORD
000276                          HC.RPK= RSPACK          ;START OF RESPONCE PACKET BUFFER
000356'                         HC.CPK= HC.RPK+PKTSIZ   ;START OF COMMAND PACKET BUFFER

                        .SBTTL COMMAND PACKET OPCODES

000001                          OP.ABO= 01              ;ABORT COMMAND
000020                          OP.ACC= 20              ;ACCESS COMMAND
000010                          OP.AVL= 10              ;AVAILABLE COMMAND
000021                          OP.CCD= 21              ;COMPARE CONTROLLER DATA COMMAND
000040                          OP.CMP= 40              ;COMPARE HOST DATA COMMAND
000013                          OP.DAP= 13              ;DETERMINE ACCESS PATHS COMMAND
000022                          OP.ERS= 22              ;ERASE COMMAND
000023                          OP.FLU= 23              ;FLUSH COMMAND
000002                          OP.GCS= 02              ;GET COMMAND STATUS COMMAND
000003                          OP.GUS= 03              ;GET UNIT STATUS COMMAND
000011                          OP.ONL= 11              ;ONLINE COMMAND
000041                          OP.RD=  41              ;READ COMMAND
000024                          OP.RPL= 24              ;REPLACE COMMAND
000004                          OP.SCC= 04              ;SET CONTROLLER CHARACTERISTICS COMMAND
000012                          OP.SUC= 12              ;SET UNIT CHARACTERISTICS COMMAND
000042                          OP.WR=  42              ;WRITE COMMAND
000030                          OP.MRD= 30              ;MAINTENANCE READ COMMAND
000031                          OP.MWR= 31              ;MAINTENANCE WRITE COMMAND
000200                          OP.END= 200             ;END PACKET FLAG
000100                          OP.AVA= 100             ;AVAILABLE ATTENTION MESSAGE
000101                          OP.ERL= 101             ;ERROR LOG ATTENTION MESSAGE
000103                          OP.SHC= 103             ;SHADOW COPY COMPLETE ATTENTION MESSAGE
000102                          OP.ACP= 102             ;ACCESS PATH ATTENTION MESSAGE


                        ; NOTE: END PACKET OPCODES (ALSO CALLED ENDCODES) ARE FORMED BY ADDING THE END
                        ; PACKET FLAG TO THE COMMAND OPCODE. THE UNKNOWN COMMAND END PACKET CONTAINS
                        ; JUST THE END PACKET FLAG IN ITS OPCODE FIELD.
```

```
;208
;209
;210
;211
;212
;213
;214
;215
;216
;217
;218
;219
;220
;221
;222
;223
;224
;225
;226
;227
;228
;229
;230
;231
;232
;233
;234
;235
;236
;237
;238
;239
;240
;241
;242
;243
;244
;245
;246
;247
;248
;249
;250
;251
;252
;253
;254
;255
;256
;257
;258
```

9.0 OPERATION OPTIONS

ONE OR MORE SOFTWARE SWITCH REGISTERS CAN BE USED BY THE MODULE PROGRAM GENERAL PURPOSE SWITCHES. THESE WORDS ARE USED TO DEFINE OR SPECIFY A UNIQUE DEVICE OPTION OR TO POINT TO A SPECIFIC ROUTINE IN THE MODULE. ANY OPTION MUST BE SPECIFIED BY THE OPERATOR BEFORE THE MODULE IS RUN. SWITCH REGISTER 1 HAS THE FOLLOWING CHARACTERISTICS (ALL BITS DEFAULT TO 0):

```
SR1
        BIT 0  SET (1):     PRINT OUT PACKETS AS THEY ARE SENT
                                AND RECIEVED (FOR MODULE DIAGNOSTIC
                                PURPOSES)
               RESET (0):   SUPRESS PRINTING OF PACKETS

        BIT 1  SET (1):     PERFORM FULL TEST OF WRITE, READ, AND DATA
                                CHECK ON ALL DEVICES WHOSE WRITE BITS
                                ARE SET IN SW3 (EXCEPT THE LOAD UNIT)
               RESET (0):   ONLY READ FROM THE DESIRED UNITS USING
                                THE MSCP READ FUNCTION.

        BIT 2  SET (1):     DO NOT REPORT ERRORS AS THEY OCCUR
               RESET (0):   REPORT ERRORS AS THEY OCCUR.

        BIT 3  SET (1):     DO NOT PRINT ERROR SUMMARY AT END OF PASS.
               RESET (0):   PRINT ERROR SUMMARY AT END OF PASS.

        BIT 4  SET (1):     PERFORM TEST OF WRITE, AND READ DBN'S ON
                                RD DEVICE IF SELECTED IN DVIGT.
               RESET (0):   DO NOT ACCESS THE DBN'S OF THE RD.

        BIT 10 SET (1):     SELECT RANDOM BLOCK ADDRESSING.
               RESET (0):   SELECT SEQUENTIAL BLOCK ADDRESSING.

        BIT 11 SET (1):     BYPASS DATA COMPARE.
               RESET (0):   DO DATA COMPARE.
```

SR2 CONTAINS A NUMBER USED AS A RATIO BETWEEN RD AND RX ACCESSES. FOR EVERY 1 ACCESS OF AN RX THERE WILL BE SR2 TO EACH RD. IF ITS CONTENTS ARE 0 (THE DEFAULT) THE VALUE 80 WILL BE USED.

```
                        MORE MODULE MESSAGES

011170      040     111     116  MSG10:  .ASCIZ  ' IN STEP '
                                         .EVEN
011204      045     052     052  MSG11:  .ASCIZ  '%** SOFT ERROR COUNT #'
                                         .EVEN
011234      045     052     052  MSG12:  .ASCIZ  '%** HARD ERROR COUNT #'
                                         .EVEN
011264      045     052     052  MSG13:  .ASCIZ  '%** CHECK DATA ERROR COUNT #'
                                         .EVEN
011322      045     101     104  MSG14:  .ASCIZ  '%ADDR = '
                                         .EVEN
011334      040     040     127  MSG16:  .ASCIZ  '  WAS FOUND OFFLINE.'
                                         .EVEN
011362      045     122     121  MSG17:  .ASCIZ  '%RQDXSA IS NOT ZERO, = '
                                         .EVEN
011412      045     122     121  MSG18:  .ASCIZ  '%RQDX IS GOING THROUGH INITILIZATION'
                                         .EVEN
011460      045     122     111  MSG19:  .ASCIZ  '%RING AREA NOT CLEARED DURING INITIALIZE%'
                                         .EVEN
011532      045     104     105  MSG20:  .ASCIZ  '%DEVICE ID BIT =  '
                                         .EVEN
011556      123     124     101  MSG21:  .ASCIZ  'STATUS ENDCOD UNIT# '
                                         .EVEN
011604      040     102     131  MSG22:  .ASCIZ  ' BYTECO HI LBN LO LBN EXTADR PHYADR%'
                                         .EVEN
011652      040     000          MSG23:  .ASCIZ  ' '
                                         .EVEN
011654      040     104     102  MSG24:  .ASCIZ  ' DBN%'
                                         .EVEN
011662      040     041     040  MSG36:  .ASCIZ  ' ! UNITS WILL ONLY BE READ TESTED !%'
                                         .EVEN
011730      007     007     040  MSG37:  .ASCII  <07><07>' ! CUSTOMER DATA WILL BE OVERWRITTEN !%'
                                         .EVEN
012002      040     055     055          .ASCIZ  ' ---------------------------------%'<07><07>
                                         .EVEN
012054      040     111     106  MSG40:  .ASCII  ' IF CUSTOMER DATA IS TO BE DESTROYED, SET BIT1 (NOT BIT0)%'
                                         .EVEN
012146      040     111     116          .ASCIZ  ' IN SWITCH REGISTER 1(SR1) EQUAL TO 1.%'
                                         .EVEN
012216      045     115     117  MSG99:  .ASCIZ  '%MORE UNITS REQUESTED THAN EXIST'
                                         .EVEN
012260      045     105     122  MSGD1:  .ASCIZ  '%ERRORS CAUSED DRIVE TO BE DROPPED%'
                                         .EVEN
012324      045     104     122  MSGD2:  .ASCIZ  '%DRIVE WAS NOT FOUND OR NOT AVAILABLE%'
                                         .EVEN
012374      045     104     126  MSGD3:  .ASCIZ  '%DVID1 BIT SET HIGHER THAN ACTUAL # OF DRIVES FOUND%'
                                         .EVEN
012462                           RBUF:   .BLKW   260.            ;THE READ BUFFER
        000001                           .END
```

```
;11
;12
;13
;14
;15
;16
;17
;18
;19
;20
;21
;22
;23
;24
;25
;26
;27
;28
;29
;30
;31
;32
;33
;34
;35
;36
;37
;38
;39
;40
;41
;42
;43
;44
;45
;46
;47
;48
;49
;50
;51
;52
;53
;54
;55
;56
;57
```

10.0 PRINTOUTS

A. MOST PRINTOUTS HAVE THE STANDARD FORMATS DESCRIBED IN
   THE DEC/X11 DOCUMENT.

B. NON STANDARD PRINTOUTS INCLUDE ERROR MESSAGES WHICH DUMP
   THE FOLLOWING:

   FOR MSCP:

```
        1)      SUMMARY STATUS
        2)      FLAGS AND ENDCODE
        3)      UNIT NUMBER
        4)      BYTE COUNT
        5)      HI 16-BIT LBN VALUE
        6)      LO 16-BIT LBN VALUE
        7)      EXTENDED ADDRESS
        8)      PHYSICAL ADDRESS
```

   ALL VALUES EXCEPT FOR PASS, RUNTIME AND ERRCNT ARE PRINTED
   IN OCTAL. PASS, RUNTIME AND ERRCNT ARE PRINTED IN DECIMAL.
   EXAMPLE:

```
   RQAA0 PA: 00064116 APC: 004630 PASS: 00000 ERRCNT: 00001
   CSRA: 174344 CSRC: 000000 ASTAT: 000006 ERRTYP: 000006
   RUNTIME: 000:00:22

   RQAA0 PA: 00064052 APC: 004564 PASS: 00000

   STATUS ENDCOD UNIT # BYTECO HI LBN LO LBN EXTADR PHYADR
   000006 000242 000005 000000 000003 116321 000001 062100
```

FOR DUP:

```
        1)      SUMMARY STATUS          (FROM REPONSE PACKET)
        2)      FLAGS AND ENDCODE       (FROM REPONSE PACKET)
        3)      UNIT NUMBER
        4)      DIAGNOSTIC BLOCK NUMBER

   RQAA0 PA: 00064116 APC: 004630 PASS: 00000 ERRCNT: 00001
   CSRA: 174344 CSRC: 000000 ASTAT: 000006 ERRTYP: 000006
   RUNTIME: 000:00:22

   STATUS ENDCOD UNIT # DBN
   000006 000243 000005 000003    INCORRECT END CODE
```

```
                        .SBTTL ERROR LOG FLAGS

000200                  EF.FRS= 000200          ;FIRST PACKET
000100                  EF.LST= 000100          ;LAST PACKET
000001                  EF.MIS= 000001          ;MESSAGE MISSING
                        ;
                        ;               ERROR LOG MESSAGE OFFSETS
                        ;
000000                  L.EVNT=  0.             ;EVENT CODE
000002                  L.SLOT=  2.             ;SLOT NUMBER
000004                  L.CNTI=  4.             ;CONTROLLER IDENTIFIER
000014                  L.CNTI= 12.             ;CONTROLLER SOFTWARE REVISION
000015                  L.CHVR= 13.             ;CONTROLLER HARDWARE REVISION
000016                  L.UNTI= 14.             ;UNIT IDENTIFIER
000026                  L.USVR= 22.             ;UNIT SOFTWARE REVISION
000027                  L.UHVR= 23.             ;UNIT HARDWARE REVISION
000030                  L.ERLC= 24.             ;ERROR LOCATION
000034                  L.CYL=  28.             ;CYLINDER
000040                  L.GRP=  32.             ;GROUP
000041                  L.TRCK= 33.             ;TRACK
000042                  L.SCTR= 34.             ;SECTOR
000044                  L.VSER= 36.             ;VOLUME SERIAL NUMBER
000050                  L.DATA= 40.             ;EVENT DEPENDENT DATA
                        ;
                        ; STATUS AND EVENT CODE DEFINITIONS
                        ;
000037                  ST.MSK= 37              ;STATUS / EVENT CODE MASK
000040                  ST.SUB= 40              ;SUB CODE MULTIPLIER
000000                  ST.SUC=  0              ;SUCCESS
000001                  ST.CMD=  1              ;INVALID COMMAND
000002                  ST.ABO=  2              ;COMMAND ABORTED
000003                  ST.OFL=  3              ;UNIT OFFLINE
000004                  ST.AVL=  4              ;UNIT AVAILABLE
000005                  ST.MFE=  5              ;MEDIA ERROR
000006                  ST.WPR=  6              ;WRITE PROTECTED
000007                  ST.CMP=  7              ;COMPARE ERROR
000010                  ST.DAT= 10              ;DATA ERROR
000011                  ST.HST= 11              ;HOST BUFFER ACCESS ERROR
000012                  ST.CNT= 12              ;CONTROLLER ERROR
000013                  ST.DRV= 13              ;DRIVE ERROR
000037                  ST.DIA= 37              ;MESSAGE FROM AN INTERNAL DIAGNOSTIC
                        ;
                        ; SUBCODES FOR ST.OFL
                        ;
000040                  SC.NVL=  40             ;NO VOLUME MOUNTED OR DRIVE DISABLED VIA RUN - STOP SWITCH
000100                  SC.IOP= 100             ;UNIT INOPERATIVE
000400                  SC.DIS= 400             ;UNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTIC
000200                  SC.DUP= 200             ;DUPLICATE UNIT NUMBER
                        ;
                        ; SUBCODES FOR ST.DRV
                        ;
000040                  SC.SDI=  40             ;SDI RESPONCE TIME OUT
000100                  SC.INV= 100             ;INVALID SDI RESPONCE
```

```
                                        12.0  BIBLIOGRAPHY

                                        CXQUAA0 'DEC/X11 USER'S MANUAL' SEPT 1978

                                        CXQAFD0 'DEC/X11 PROGRAMMERS'S GUIDE' SEPT 1978

                                .
                                .SBTTL  MODULE HEADER BLOCK
405 000000                              IOMODX <RQAD  >,172150,154,4,0,0,40,104,RBUF,256.,256.
    000000                              MODULE  150000,RQAD  ,172150,154,4,0,0,40,104,RBUF 256.,256.
                                        .TITLE  RQAD DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6.4    28-JAN-82
                                        .LIST   BIN
                                ;*******************************************************************************
000000                          BEGIN:
000000      122     121     101 MODNAM: .ASCII  /RQAD / ;MODULE NAME.
000003      104     040
000005      000                 XFLAG:  .BYTE   OPEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006  172150                  ADDR:   172150+0                ;1ST DEVICE ADDR.
000010  000154                  VECTOR: 154+0                   ;1ST DEVICE VECTOR.
000012      200                 BR1:    .BYTE   PRI+4+0         ;1ST BR LEVEL.
000013      000                 BR2:    .BYTE   PRI+0+0         ;2ND BR LEVEL.
000014  000001                  DVID1:  0+1                     ;DEVICE INDICATOR 1.
000016  000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020  000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022  000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024  000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*******************************************************************************
000026  150000                  STAT:   150000                  ;STATUS WORD.
000030  001002                  INIT:   START                   ;MODULE START ADDR.
000032  000252                  SPOINT: MODSP                   ;MODULE STACK POINTER.
000034  000000                  PASCNT: 0                       ;PASS COUNTER.
000036  000040                  ICONT:  40                      ;# OF ITERATIONS PER PASS=40
000040  000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042  000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044  000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046  000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050  000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052  000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054  000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056                          CONFIG:                         ;RESERVED FOR MONITOR USE
000056  000000                  RES1:   0                       ;RESERVED FOR MONITOR USE.
000060  000000                  RES2:   0                       ;RESERVED FOR MONITOR USE.
000062  000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064  000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066  000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070  000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072  000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074  000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076  000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
001000  000000                  CURAD:  OPEN                    ;ADDR OF CURRENT ERR.
000102                          SBADR:                          ;ADDR OF GOOD DATA, OR
```

```
                        ;*****************************************************************************
                        .SBTTL  MODULE STORAGE AREA
                        ;       VERSION 2.0     FOR RELEASE
                        ;                       XRQAD0 - THIS MODULE IS ADAPTED FREELY FROM XDUBC.
                        ;                       THE GENERAL STRUCTURE IS THE SAME, BUT SOME DEVICE
                        ;                       SPECIFIC CODE WAS ADDED.
                        ;
                        ;RQDXIP:.WORD   0               ;RQDX POLLING REG
000252  000000          RQDXSA: .WORD   0               ;RQDX STATUS REG
                        ;-----------------------------------------------------------------------------
                        ; THESE STORAGE AREAS ARE USED FOR COMMUNICATIONS WITH THE
                        ; CONTROLLER.
                        ;-----------------------------------------------------------------------------
000254  000000          CINTR:  .WORD   0               ; COMMAND INTERRUPT INDICATOR
000256  000000          RINTR:  .WORD   0               ; RESPONCE INTERRUPT INDICATOR
000260                  RSPONC: .BLKW   2.              ; MESSAGE RING
000264                  COMMND: .BLKW   2.              ; COMMAND RING
000270  000000                  .WORD   0               ;
                        ;-----------------------------------------------------------------------------
000272  000000          RSPLEN: .WORD   0               ; RESPONCE PACKET LENGTH
000274  000000          RSPVIR: .WORD   0               ; RESPONCE PACKET VIRTUAL CIRCUIT
000276                  RSPACK: .BLKW   30.             ; RESPONCE PACKET
                        ;-----------------------------------------------------------------------------
000372  000000          CMPLEN: .WORD   0               ; COMMAND PACKET LENGTH
000374  000000          CMPVIR: .WORD   0               ; COMMAND PACKET VIRTUAL CIRCUIT
000376                  CMPACK: .BLKW   30.             ; COMMAND PACKET
                        ;-----------------------------------------------------------------------------
                        ; THESE STORAGE AREAS ARE TO KEEP TRACK OF THE ABOVE COMMNICATION
                        ; AREAS.
                        ;-----------------------------------------------------------------------------
000472  000000          CMDREF: .WORD   0               ; COMMAND REFERENCE NUMBER
000474  000000          BYTCNT: .WORD   0               ; BYTE COUNT
000476  000000          DATLOC: .WORD   0               ; DATA LOCATION
000500  000000          ENDCOD: .WORD   0               ; END CODE EXPECTED
                        ;-----------------------------------------------------------------------------
                        ; ADDRESS MAPPING AREAS.
                        ;-----------------------------------------------------------------------------
000502  000264'         VA:     .WORD   COMMND          ;GENERIC VIRTUAL ADDRESS FOR GETPA
000504  000000          PA:     .WORD   OPEN            ;GENERIC PHYSICAL ADDRESS
000506  000000          EA:      WORD   OPEN            ;GENERIC EXTENDED ADDRESS
000510  000000          LO22:   .WORD   0               ;22 BIT LOW ORDER ADDRESS
000512  000000          HI22:   .WORD   0               ;22 BIT HIGH ORDER ADDRESS

000514  000000          NUM:    .WORD   0               ;ADDRESS USED IN OTOA
000516  000000          OLDPA:  .WORD   0               ;THE OLD PHYSICAL ADDRESS
000520  000000          OLDEA:  .WORD   0               ;THE OLD EXTENDED ADDRESS TO CHECK IF
                                                        ;       RQDX WILL BE REINITED
000522  000000          SAVPA:  .WORD   0               ; SAVED PA NEEDED FOR DUP
000524  000000          SAVEA:  .WORD   0               ; SAVED EA
                        ;-----------------------------------------------------------------------------
```

```
458                                     ;
459             000017                  PRTNUM = 15.                    ;PRINT MESSAGE EVERY 15TH TIME
460 000526      000017                  PRNMSG: .WORD   PRTNUM           ;PRINT WORD SAVES THE VALUE TO CHECK FOR WHEN
461                                                                      ; AN END OF PASS MESSAGE IS WRITTEN
462             027340                  TIMER = 12000.                  ;TIMER TO WAIT 2 3 SECONDS AFTER DAP COMMAND
463
464 000530                              TEMP:                           ;LABEL FOR CONVENIANCE
466                                     ADR'X:  .BLKB   6
467                                             .BYTE   0
468                                     .ENDM
    000530                              ADR1:   .BLKB   6
    000536      000                             .BYTE   0
    000537                              ADR2:   .BLKB   6
    000545      000                             .BYTE   0
    000546                              ADR3:   .BLKB   6
    000554      000                             .BYTE   0
    000555                              ADR4:   .BLKB   6
    000563      000                             .BYTE   0
    000564                              ADR5:   .BLKB   6
    000572      000                             .BYTE   0
    000573                              ADR6:   .BLKB   6
    000601      000                             .BYTE   0
    000602                              ADR7:   .BLKB   6
    000610      000                             .BYTE   0
    000611                              ADR8:   .BLKB   6
    000617      000                             .BYTE   0
469                                             .EVEN
```

```
                        .SBTTL  MORE MODULE STORAGE

000620  000000  FLAGS:  .WORD   0               ; MODULE FLAGS
000622  000000  ERRFLG: .WORD   0               ; ERROR FLAGS FROM DUP PART OF MODULE
000624  000000  DBN:    .WORD   0               ; DBN FOR DUP
000626  000000  SFTFLG: .WORD   0               ; SOFT ERROR FLAGS

000630  000000  XSECL:  .WORD   0               ;CURRENT SECTOR LO ORDER ADDRESS FOR DX
000632  000000  XSECH:  .WORD   0               ;CURRENT SECTOR HI ORDER ADDRESS FOR DX

000634  000000  RSECL:  .WORD   0               ;CURRENT SECTOR LO ORDER ADDRESS FOR DR
000636  000000  RSECH:  .WORD   0               ;CURRENT SECTOR HI ORDER ADDRESS FOR DR

000640  000200  UNSZL:  .WORD   0               ;UNIT SIZE LO ORDER LIMIT FROM ONLINE CMND
000642  000000  UNSZH:  .WORD   0               ;UNIT SIZE HI ORDER LIMIT

000644  000001  DVICE:  .WORD   1               ;DEVICE TO TEST
000646  000000  TRY:    .WORD   0               ;NUMBER OF TRIES
000650  000000  UNITFL: .WORD   0               ;SAVE UNIT FLAGS
000652  000000  WORK:   .WORD   0               ;TEMPORARY WORK AREA

        0056*0  TIMOUT = 3000.                  ;TIME OUT GADGE
        000004  RLIM = 4                        ;RETRY LIMIT

000654  000000  ISRX:   .WORD   0               ;BITMAP, SET IF DEVICE IS RX, CLEAR IF RD

        000001  $$ =1                           ;SET UP BIT MASK
000656  000020  TABLEW: .REPT   16.
                .NLIST
                .WORD   0,$$
                $$ =$$*2
                .LIST
                .ENDR
000756          TEND:                           ;END MARKER

000756  000000  RVA:    .WORD   0               ;READ BUFFER STRUCTURE FOR 22 BIT ADDRESS
000760  000000  R18PA:  .WORD   0
000762  000000  R18EA:  .WORD   0
000764  000000  R22PA:  .WORD   0
000766  000000  R22EA:  .WORD   0

000770  000000  WVA:    .WORD   0               ;WRITE BUFFER STRUCTURE FOR 22 BIT ADDRESS
000772  000000  W18PA:  .WORD   0
000774  000000  W18EA:  .WORD   0
000776  000000  W22PA:  .WORD   0
001000  000000  W22EA:  .WORD   0
```

```
                        .SBTTL  MODULE PRIVATE DATA
                        ;******************************************************************************
                        ;*      MACRO DEFINITIONS
                        ;******************************************************************************
                                .SBTTL  MACROS
                        ;
                        ; IN CASE NO SOB INSTRUCTION
                        ;
000001                  NOSOB   =       1
                        .MACRO  SOB     REG,TO
                                DEC     REG
                                BNE     TO
                        .ENDM

                        ;******************************************************************************

                        .MACRO  PUSH    A,B,C,D,E
                                MOV     A,-(R6)
                        .IF     NB      B
                                MOV     B,-(R6)
                        .ENDC
                        .IF     NB      C
                                MOV     C,-(R6)
                        .ENDC
                        .IF     NB      D
                                MOV     D,-(R6)
                        .ENDC
                        .IF     NB      E
                                MOV     E,-(R6)
                        .ENDC
                        .ENDM

                        ;******************************************************************************

                        .MACRO  POP     A,B,C,D,E
                                MOV     (R6)+,A
                        .IF     NB      B
                                MOV     (R6)+,B
                        .ENDC
                        .IF     NB      C
                                MOV     (R6)+,C
                        .ENDC
                        .IF     NB      D
                                MOV     (R6)+,D
                        .ENDC
                        .IF     NB      E
                                MOV     (R6)+,E
                        .ENDC
                        .ENDM

                        ;******************************************************************************
```

```
                        .SBTTL  EQUATES

000001                  BIT00 = 1
000002                  BIT01 = 2
000004                  BIT02 = 4
000010                  BIT03 = 10
000020                  BIT04 = 20
000040                  BIT05 = 40
000100                  BIT06 = 100
000200                  BIT07 = 200
000400                  BIT08 = 400
001000                  BIT09 = 1000
002000                  BIT10 = 2000
004000                  BIT11 = 4000
010000                  BIT12 = 10000
020000                  BIT13 = 20000
040000                  BIT14 = 40000
100000                  BIT15 = 100000

000002                  UPPER = 2

                        ;
                        ;       STATUS BITS
                        ;
000001                  SR.PAK = BIT00  ; PRINT OUT PACKET, 1 = YES
000002                  SR.RON = BIT01  ; 0 MEANS READ ONLY
                                        ; 1 MEANS WRITE, READ, CHECK
000004                  SR.REP = BIT02  ; REPORT ERROR AS THEY OCCUR    0 = REPORT,
                                        ;                               1 = DON'T REPORT
000010                  SR.SUM = BIT03  ; REPORT ERRORS ON END OF PASS 0 = REPORT,
                                        ;                               1 = DON'T REPORT
002000                  SR.SEQ = BIT10  ; DISK ADDRESSING 0 = SEQUENTIAL,
                                        ;                 1 = RANDOM
004000                  SR.CMP = BIT11  ; DATA COMPARE    0 = DO DATA COMPARE,
                                        ;                 1 = DON'T DO DATA COMPARE
                        ;
                        ;       ERROR BITS
                        ;
000000                  ERR.0 = 0       ; NOT DEFINED
000001                  ERR.1 = 1       ; DATA ERROR
000003                  ERR.3 = 3       ; CONTROLLER NOT READY
000004                  ERR.4 = 4       ; BLOCK NOT FOUND
000006                  ERR.6 = 6       ; DRIVE NOT READY, OFF LINE OR NON EXESTENT
000032                  ERR.32 = 32     ; NPR ERROR
000034                  ERR.34 = 34     ; UNABLE TO INITALIZE ERROR
000036                  ERR.36 = 36     ; UNABLE TO READ  ERROR
000037                  ERR.37 = 37     ; UNABLE TO WRITE ERROR
000045                  ERR.45 = 45     ; CRC  ERROR
000051                  ERR.51 = 51     ; SEEK ERROR
```

RQDX BIT DEFINITIONS

```
                        .SBTTL  RQDX BIT DEFINITIONS

                        ;RQDXSA REGISTER UNIVERSAL READ BITS

000040                          SA.S1=  004000          ;STEP 1 STATUS BIT
000020                          SA.S2=  010000          ;STEP 2 STATUS BIT
000010                          SA.S3=  020000          ;STEP 3 STATUS BIT
000004                          SA.S4=  040000          ;STEP 4 STATUS BIT
000002                          SA.ERR= 100000          ;ERROR INDICATOR

                        ;RQDXSA REGISTER ERROR STATUS BITS

003777                          SA.ERC= 003777          ;ERROR CODE

                        ;RQDXSA REGISTER STEP ONE READ BITS

003400                          SA.CTP= 003400          ;CONTROLLER TYPE
000400                          SA.DIA= 000400          ;DIAG BIT IN RQDXSA
001000                          SA.QB = 001000          ;SAME - DENOTES QBUS SUPPORT

                        ;RQDXSA REGISTER STEP ONE WRITE BITS

000177                          SA.VEC= 000177          ;INTERRUPT VECTOR (DIVIDED BY 4)
000200                          SA.INT= 000200          ;INTERRUPT ENABLE DURING INITIALIZATION
007400                          SA.RSP= 007400          ;RESPONCE RING LENGTH
170000                          SA.CMD= 170000          ;COMMAND RING LENGTH

                        ;RQDXSA REGISTER STEP TWO READ BITS

000177                          SA.VCE= 000177          ;INTERRUPT VECTOR ECHO
000200                          SA.INE= 000200          ;INTERRUPT ENABLE ECHO

                        ;RQDXSA REGISTER STEP TWO WRITE BITS

                                                        ;LOW ORDER MESSAGE RING BYTE ADDRESS
000001                          SA.PRG= 000001          ;ENABLE VAX UNIBUS ADAPTER PURGE INTERRUPT

                        ;RQDXSA REGISTER STEP THREE READ BITS

000017                          SA.RSE= 000017          ;RESPONCE RING LENGTH ECHO
000360                          SA.CME= 000360          ;COMMAND RING LENGTH ECHO

                        ;RQDXSA REGISTER STEP THREE WRITE BITS

                                                        ;HIGH ORDER MESSAGE RING BYTE ADDRESS
040000                          SA.LFC  040000          ;LAST FAILURE CODE REQUEST

                        ;RQDXSA REGISTER STEP FOUR READ BITS

                        ;RQDXSA REGISTER STEP FOUR WRITE BITS

000001                          SA.GO=  BIT0            ;GO BIT TO START RQDX FIRMWARE
```

COMMAND MESSAGE DESCRIPTOR BIT DEFINITIONS

```
                        .SBTTL  COMMAND/MESSAGE DESCRIPTOR BIT DEFINITIONS

100000                          RG.OWN= BIT15           ;SET WHEN RQDX OWNS RING
040000                          RG.FLG= BIT14           ;FLAG BIT

                        ;OFFSETS INTO HOST COMMUNICATIONS AREA WITH ONE DESCRIPTOR TO EACH RING

000010                          HC.SIZ= 8.              ;SIZE OF HOST COMM AREA IN BYTES
000060                          PKTSIZ= 48.             ;SIZE OF PACKETS IN BYTES

000000                          HC.RES= 0.              ;RESPONCE RING START
000002                          HC.RCT= 2.              ;RESPONCE RING CONTROL WORD
000004                          HC.CMD= 4.              ;COMMAND RING START
000006                          HC.CCT= 6.              ;CONTROL RING CONTROL WORD
000026                          HC.RPK= RSPACK          ;START OF RESPONCE PACKET BUFFER
000356'                         HC.CPK= HC.RPK+PKTSIZ   ;START OF COMMAND PACKET BUFFER

                        .SBTTL  COMMAND PACKET OPCODES

000001                          OP.ABO= 01              ;ABORT COMMAND
000020                          OP.ACC= 20              ;ACCESS COMMAND
000010                          OP.AVL= 10              ;AVAILABLE COMMAND
000021                          OP.CCD= 21              ;COMPARE CONTROLLER DATA COMMAND
000040                          OP.CMP= 40              ;COMPARE HOST DATA COMMAND
000013                          OP.DAP= 13              ;DETERMINE ACCESS PATHS COMMAND
000022                          OP.ERS= 22              ;ERASE COMMAND
000023                          OP.FLU= 23              ;FLUSH COMMAND
000002                          OP.GCS= 02              ;GET COMMAND STATUS COMMAND
000003                          OP.GUS= 03              ;GET UNIT STATUS COMMAND
000011                          OP.ONL= 11              ;ONLINE COMMAND
000041                          OP.RD=  41              ;READ COMMAND
000024                          OP.RPL= 24              ;REPLACE COMMAND
000004                          OP.SCC= 04              ;SET CONTROLLER CHARACTERISTICS COMMAND
000012                          OP.SUC= 12              ;SET UNIT CHARACTERISTICS COMMAND
000042                          OP.WR=  42              ;WRITE COMMAND
000030                          OP.MRD= 30              ;MAINTENANCE READ COMMAND
000031                          OP.MWR= 31              ;MAINTENANCE WRITE COMMAND
000200                          OP.END= 200             ;END PACKET FLAG
000100                          OP.AVA= 100             ;AVAILABLE ATTENTION MESSAGE
000101                          OP.ERL= 101             ;ERROR LOG ATTENTION MESSAGE
000102                          OP.SHC= 102             ;SHADOW COPY COMPLETE ATTENTION MESSAGE
                                OP.ACP= 103             ;ACCESS PATH ATTENTION MESSAGE

                        ; NOTE: END PACKET OPCODES (ALSO CALLED ENDCODES) ARE FORMED BY ADDING THE END
                        ; PACKET FLAG TO THE COMMAND OPCODE.  THE UNKNOWN COMMAND END PACKET CONTAINS
                        ; JUST THE END PACKET FLAG IN ITS OPCODE FIELD.
```

```
                                .SBTTL COMMAND MODIFIERS

040000                                  MD.CMP= 040000          ;COMPARE
100000                                  MD.EXP= 100000          ;EXPRESS REQUEST
010000                                  MD.ERR= 010000          ;FORCE ERROR
004000                                  MD.SCH= 004000          ;SUPPRESS CACHING (HIGH SPEED)
002000                                  MD.SCL= 002000          ;SUPPRESS CACHING (LOW SPEED)
001000                                  MD.SEC= 001000          ;SUPPRESS ERROR CORRECTION
000400                                  MD.SER= 000400          ;SUPPRESS ERROR RECOVERY
000200                                  MD.SSH= 000200          ;SUPPRESS SHADOWING
000100                                  MD.WBN= 000100          ;WRITE-BACK (NON VOLATILE)
000040                                  MD.WBV= 000040          ;WRITE BACK (VOLATILE)
000001                                  MD.SPD= 000001          ;SPIN-DOWN
000001                                  MD.FEU= 000001          ;FLUSH ENTIRE UNIT
000002                                  MD.VOL= 000002          ;VOLATILE ONLY
000001                                  MD.NXU= 000001          ;NEXT UNIT

                                .SBTTL END PACKET FLAGS

000200                                  EF.BBR= 000200          ;BAD BLOCK REPORTED
000100                                  EF.BBU= 000100          ;BAD BLOCK UNREPORTED
000040                                  EF.LOG= 000040          ;ERROR LOG GENERATED
000020                                  EF.SEX= 000020          ;SERIOUS EXCEPTION


                                .SBTTL UNIT FLAGS

000001                                  UF.CMR= 000001          ;COMPARE READS
000002                                  UF.CMW= 000002          ;COMPARE WRITES
010000                                  UF.RPL= 010000          ;HOST INITIATED BAD BLOCK REPLACEMENT
040000                                  UF.INA= 040000          ;INACTIVE SHADOW SET UNIT
000200                                  UF.RMV= 000200          ;REMOVEABLE MEDIA
004000                                  UF.SCH= 004000          ;SUPPRESS CACHING (HIGH SPEED)
002000                                  UF.SCL= 002000          ;SUPPRESS CACHING (LOW SPEED)
000040                                  UF.WBN= 000040          ;WRITE BACK (NON VOLATILE)
020000                                  UF.WPH= 020000          ;WRITE PROTECT(HARDWARE)
010000                                  UF.WPS= 010000          ;WRITE PROTECT(SOFTWARE OR VOLUME)
000004                                  UF.576= 000004          ;576 BYTE SECTORS
```

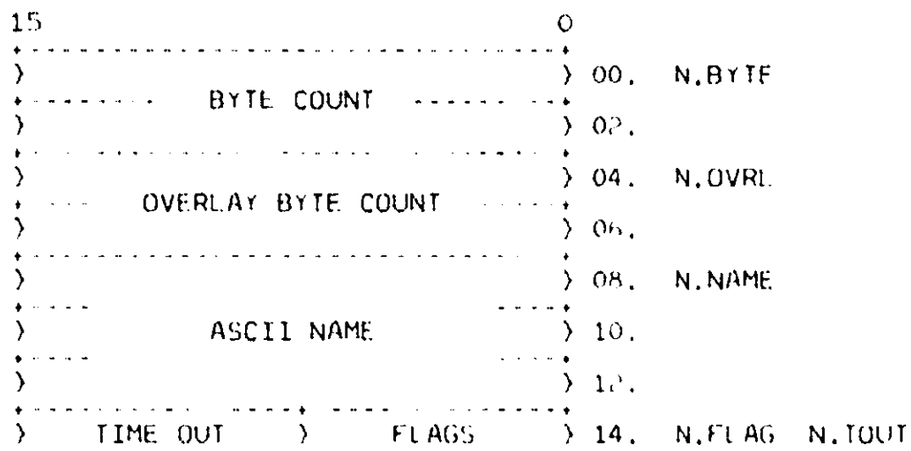
```
                        .SBTTL  CONTROLLER FLAGS

000200                          CF.AVL= 000200          ;ENABLE AVAILABLE ATTENTION MESSAGES
000100                          CF.MSC= 000100          ;ENABLE MISCELLANEOUS ERROR LOG MESSAGES
000040                          CF.OTH= 000040          ;ENABLE OTHER HOST'S ERROR LOG MESSAGES
000020                          CF.THS= 000020          ;ENABLE THIS HOST'S ERROR LOG MESSAGES
000002                          CF.SHD= 000002          ;SHADOWING
000001                          CF.576= 000001          ;576 BYTE SECTORS

                        .SBTTL  COMMAND PACKET OFFSETS

                        ; GENERIC COMMAND PACKET OFFSETS:
000000                          P.CREF=  0.             ;COMMAND REFERENCE NUMBER
000004                          P.UNIT=  4.             ;UNIT NUMBER
000010                          P.OPCD=  8.             ;OPCODE
000012                          P.MOD=  10.             ;MODIFIERS
000014                          P.BCNT= 12.             ;BYTE COUNT
000020                          P.BUFF= 16.             ;BUFFER DESCRIPTOR
000020                          P.ADPA= 16.             ;BUFFER'S PHYSICAL ADDRESS (P.BUFF)
000022                          P.ADEA= 18.             ;BUFFER'S EXTENDED ADDRESS (P.BUFF+2)
000034                          P.LBN=  28.             ;LOGICAL BLOCK NUMBER
000040                          P.SFTW= 32.             ;SOFTWARE WORDS

                        ; ABORT AND GET COMMAND STATUS COMMAND PACKET OFFSETS:
000014                          P.OTRF= 12.             ;OUTSTANDING REFERENCE NUMBER

                        ; ONLINE AND SET UNIT CHARACTERISTICS COMMAND PACKET OFFSETS:
000016                          P.UNFL= 14.             ;UNIT FLAGS
000020                          P.HSTI= 16.             ;HOST IDENTIFIER
000034                          P.UNTI= 28.             ;UNIT IDENTIFIER
000034                          P.ELGF= 28.             ;ERROR LOG FLAGS
000040                          P.SHUN= 32.             ;SHADOW UNIT
000042                          P.CPSP= 34.             ;COPY SPEED

                        ; REPLACE COMMAND PACKET OFFSETS:
000014                          P.RBN=  12.             ;REPLACEMENT BLOCK NUMBER

                        ; SET CONTROLLER CHARACTERISTICS COMMAND PACKET OFFSETS:
000014                          P.VRSN= 12.             ;MSCP VERSION
000016                          P.CNTF= 14.             ;CONTROLLER FLAGS
000020                          P.HTMO= 16.             ;HOST TIMEOUT
000022                          P.USEF= 18.             ;USE FRACTION
000024                          P.TIME= 20.             ;QUAD WORD TIME AND DATE

                        ; MAINTENANCE READ AND MAINTENANCE WRITE COMMAND PACKET OFFSETS:
000034                          P.RGID= 28.             ;REGION ID
000040                          P.RGOF= 32.             ;REGION OFFSET
```

```
                        .SBTTL  END PACKET OFFSETS

                        ; GENERIC END PACKET OFFSETS:
000000                          P.CRF=   0.             ;COMMAND REFERENCE NUMBER
000004                          P.UNIT=  4.             ;UNIT NUMBER
000010                          P.OPCD=  8.             ;OPCODE (ALSO CALLED ENDCODE)
000011                          P.FLGS=  9.             ;END PACKET FLAGS
000012                          P.STS=  10.             ;MODIFIERS
000014                          P.BCNT= 12.             ;BYTE COUNT
000034                          P.FBBK= 28.             ;FIRST BAD BLOCK
000040                          P.SFTW= 32.             ;SOFTWARE WORDS

                        ; GET COMMAND STATUS END PACKET OFFSETS:
000014                          P.OTRF= 12.             ;OUTSTANDING REFERENCE NUMBER
000020                          P.CMST= 16.             ;COMMAND STATUS

                        ; GET UNIT STATUS END PACKET OFFSETS:
000014                          P.MUUN= 12.             ;MULTI-UNIT CODE
000016                          P.UNFL= 14.             ;UNIT FLAGS
000020                          P.HSTI= 16.             ;HOST IDENTIFIER
000024                          P.UNTI= 20.             ;UNIT IDENTIFIER
000034                          P.DEV=  28.             ;MEDIA TYPE IDENTIFIER
000040                          P.SHUN= 32.             ;SHADOW UNIT
000042                          P.SHST= 34.             ;SHADOW STATUS
000044                          P.TRCK= 36.             ;TRACK SIZE
000046                          P.GRP=  38.             ;GROUP SIZE
000050                          P.CYL=  40.             ;CYLINDER SIZE
000054                          P.RCTS= 44.             ;RCT TABLE SIZE
000056                          P.RBNS= 46.             ;RBN / TRACK
000057                          P.RCTC= 47.             ;RCT COPIES

                        ; ONLINE AND SET UNIT CHARACTERISTICS
000014                          P.MUUN= 12.             ;MULTI-UNIT CODE
000016                          P.UNFL= 14.             ;UNIT FLAGS
000020                          P.HSTI= 16.             ;HOST IDENTIFIER
000024                          P.UNTI= 20.             ;UNIT IDENTIFIER
000040                          P.SHUN= 32.             ;SHADOW UNIT
000044                          P.UNSZ= 36.             ;UNIT SIZE
000050                          P.VSER= 40.             ;VOLUME SERIAL NUMBER

                        ; SET CONTROLLER CHARACTERISTICS END PACKET OFFSETS:
000014                          P.VRSN= 12.             ;MSCP VERSION
000016                          P.CNTF= 14.             ;CONTROLLER FLAGS
000020                          P.CTMO= 16.             ;CONTROLLER TIMEOUT
000022                          P.CNCL= 18.             ;CONTROLLER COMMAND LIMIT
000024                          P.CNTI= 20.             ;CONTROLLER ID
000034                          P.MEDI= 28.             ;MEDIA TYPE
000040                          P.SHST= 34.             ;SHADOW STATUS

                        ; ERROR LOG ATTENTION MESSAGE PACKET OFFSETS:
000000                          P.CRF=   0.             ;COMMAND REFERENCE NUMBER
000004                          P.UNIT=  4.             ;UNIT NUMBER
000006                          P.CNT=   6.             ;COUNT
000010                          P.OPCD=  8.             ;OPCODE
000011                          P.FLGS=  9.             ;ERROR LOG FLAGS
000012                          P.EVOF= 10.             ;EVENT OR OFFSET
000014                          P.LDAT= 12.             ;START OF ERROR LOG DATA
```

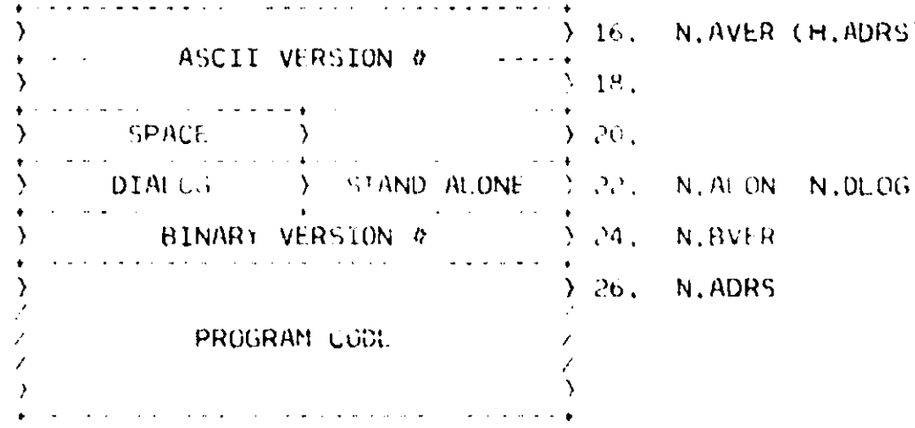
ERROR LOG FLAGS

```
                        .SBTTL ERROR LOG FLAGS

000200                          EF.FRS= 000200          ;FIRST PACKET
000100                          EF.LST= 000100          ;LAST PACKET
000001                          EF.MIS= 000001          ;MESSAGE MISSING
                        ;
                        ;               ERROR LOG MESSAGE OFFSETS
                        ;
000000                          L.EVNT=  0.             ;EVENT CODE
000002                          L.SLOT=  2.             ;SLOT NUMBER
000004                          L.CNTI=  4.             ;CONTROLLER IDENTIFIER
000014                          L.CNTI= 12.             ;CONTROLLER SOFTWARE REVISION
000015                          L.CHVR= 13.             ;CONTROLLER HARDWARE REVISION
000016                          L.UNTI= 14.             ;UNIT IDENTIFIER
000026                          L.USVR= 22.             ;UNIT SOFTWARE REVISION
000027                          L.UHVR= 23.             ;UNIT HARDWARE REVISION
000030                          L.ERLC= 24.             ;ERROR LOCATION
000034                          L.CYL=  28.             ;CYLINDER
000040                          L.GRP=  32.             ;GROUP
000041                          L.TRCK= 33.             ;TRACK
000042                          L.SCTR= 34.             ;SECTOR
000044                          L.VSER= 36.             ;VOLUME SERIAL NUMBER
000050                          L.DATA= 40.             ;EVENT DEPENDENT DATA
                        ;
                        ; STATUS AND EVENT CODE DEFINITIONS
                        ;
000037                          ST.MSK= 37              ;STATUS / EVENT CODE MASK
000040                          ST.SUB= 40              ;SUB-CODE MULTIPLIER
000000                          ST.SUC=  0              ;SUCCESS
000001                          ST.CMD=  1              ;INVALID COMMAND
000002                          ST.ABO=  2              ;COMMAND ABORTED
000003                          ST.OFL=  3              ;UNIT-OFFLINE
000004                          ST.AVL=  4              ;UNIT-AVAILABLE
000005                          ST.MFE=  5              ;MEDIA ERROR
000006                          ST.WPR=  6              ;WRITE PROTECTED
000007                          ST.CMP=  7              ;COMPARE ERROR
000010                          ST.DAT= 10              ;DATA ERROR
000011                          ST.HST= 11              ;HOST BUFFER ACCESS ERROR
000012                          ST.CNT= 12              ;CONTROLLER ERROR
000013                          ST.DRV= 13              ;DRIVE ERROR
000037                          ST.DIA= 37              ;MESSAGE FROM AN INTERNAL DIAGNOSTIC
                        ;
                        ; SUBCODES FOR ST.OFL
                        ;
000040                          SC.NVL=  40             ;NO VOLUME MOUNTED OR DRIVE DISABLED VIA RUN-STOP SWITCH
000100                          SC.IOP= 100             ;UNIT INOPERATIVE
000400                          SC.DIS= 400             ;UNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTIC
000200                          SC.DUP= 200             ;DUPLICATE UNIT NUMBER
                        ;
                        ; SUBCODES FOR ST.DRV
                        ;
000040                          SC.STO=  40             ;SDI RESPONCE TIME OUT
000100                          SC.INV= 100             ;INVALID SDI RESPONCE
```

```
                        .SBTTL DUP DEFINITIONS

                                                                        .REM    &
                         ALL REMOTE PROGRAMS (ROM BASED OR MEDIA BASED) START WITH A HEADER,
                        WHICH LOOKS LIKE THIS:
```




```
                        LOCAL PROGRAMS HAVE AN EXTENDED HEADER, WHICH INCLUDES:
```




```
                        PROGRAM CODE FOR LOCAL PROGRAMS STARTS AT N.ADRS, WHEREAS CODE FOR
                        HOST SUPPLIED PROGRAMS STARTS AT H.ADRS.
                                                                                &

                        ;OTHER USEFUL CONSTANTS.

 177400                         HIBYTE = 177400         ;MASK FOR THE HIGH ORDER BYTE.
 000017                         LONIBL = 017            ;MASK FOR THE LOW ORDER NIBBLE.
 000020                         HINIBL = 20             ;MULTIPLIER TO SHIFT DATA TO HIGH NIBBLE.
```

```
                        .SBTTL  OFFSET AND FLAG DEFINITIONS

                        ; OFFSET DEFINITIONS.
                        ; ------ -----------
                        ;
                        ; GENERIC COMMAND/RESPONSE PACKET.
                        ;
000011                          P.RSVD= 9.              ;RESERVED FIELD.
                        ;
                        ; GET DUST STATUS RESPONSE.
                        ;
000014                          S.EXT = 12.             ;PROGRAM EXTENSION.
000017                          S.FLAG= 15.             ;FLAGS.
000020                          S.PRGS= 16.             ;PROGRESS INDICATOR.
000024                          S.TOUT= 20.             ;TIME-OUT VALUE FOR LOCAL PROGRAMS.
                        ;
                        ; EXECUTE SUPPLIED PROGRAM COMMAND.
                        ;
000034                          P.OVRL= 28.             ;OVERLAY BUFFER DESCRIPTOR.
                        ;
                        ; EXECUTE LOCAL PROGRAM COMMAND.
                        ;
000014                          L.NAME= 12.             ;PROGRAM NAME.
                        ;
                        ; EXECUTE LOCAL PROGRAM RESPONSE.
                        ;
000014                          L.VERS= 12.             ;PROGRAM VERSION.
000016                          L.TOUT= 14.             ;TIME OUT IN SECS.
000017                          L.FLAG= 15.             ;FLAGS.
                        ;
                        ; LOCAL PROGRAM'S HEADER TABLE.
                        ;
000000                          N.BYTE= 0               ;BYTE COUNT.
000004                          N.OVRL= 4               ;OVERLAY BYTE COUNT.
000010                          N.NAME= 8.              ;ASCII NAME.
000016                          N.FLAG= 14.             ;FLAGS.
000017                          N.TOUT= 15.             ;TIME OUT VALUE.
000020                          N.AVER= 16.             ;ASCII VERSION NUMBER.
000026                          N.ALON= 22.             ;STAND-ALONE FLAG.
000027                          N.DLOG= 23.             ;STANDARD DIALOG FLAG.
000030                          N.BVER= 24.             ;BINARY VERSION NUMBER.
000032                          N.ADRS= 26.             ;START OF PROGRAM PROPER.
                        ;
                        ; HOST SUPPLIED PROGRAM'S HEADER TABLE.
                        ;
000020                          H.ADRS  N.AVER          ;START OF PROGRAM PROPER.
                        ;
                        ; SEND DATA TEXT.
                        ;
000000                          S.TEXT= 0               ;ASCII TEXT.
                        ;
                        ; RECEIVE DATA TEXT.
                        ;
000000                          R.MESG= 0               ;MESSAGE NUMBER.
000001                          R.TYPE= 1               ;TYPE.
000002                          R.TEXT= 2               ;ASCII TEXT.
```

```
                                ; FLAGS AND MODIFIERS.
                                ;---------------------
                                ;
                                ; GET DUST STATUS RESPONSE.
                                ;
000000                                  SE.IDL= 0               ;SERVER IDLE.
000001                                  SE.DIS= BIT0            ;PROGRAM EXECUTION DISBLES ALL OTHER SERVERS.
000002                                  SE.LOC= BIT1            ;LOCAL LOAD MEDIA AVAILABLE.
000004                                  SE.SUP= BIT2            ;WILL NOT ACCEPT EXECUTE SUPPLIED PROGRAM.
000010                                  SE.ACT= BIT3            ;SERVER ACTIVE.
                                ;
                                ; EXECUTE LOCAL PROGRAM OPCODE MODIFIER.
                                ;
000001                                  LM.ALO= BIT0            ;ALLOW STAND-ALONE OPERATION OF LOCAL PROGRAM.
                                ;
                                ; REMOTE PROGRAM HEADER FLAGS.
                                ;
000001                                  PH.ALO= BIT0            ;STANDALONE.
000002                                  PH.OVR= BIT1            ;NEEDS OVERLAYS FROM THE HOST.
000004                                  PH.RW = BIT2            ;OVERLAY IS R/W.
000010                                  PH.DLG= BIT3            ;PROGRAM USES STANDARD DUP DIALOG.
                                ;
                                ; CODES, TYPES, ETC.
                                ;-------------------
                                ;
                                ; OP CODES
                                ;
000001                                  OP.GDS = 1               GET DUST STATUS
000002                                  OP.ESP = 2              ;EXECUTE SUPPLIED PROGRAM
000003                                  OP.ELP = 3              ;EXECUTE LOCAL PROGRAM
000004                                  OP.SND = 4              ;SEND DATA
000005                                  OP.RCD = 5              ;RECEIVE DATA
000006                                  OP.ABP = 6              ;ABORT PROGRAM
                                ;
                                ; STATUS CODES.
                                ;
000000                                  ST.SUC = 0              ;SUCCESS
000001                                  ST.CMD = 1              ;ILLEGAL COMMAND
000002                                  ST.NRA= 2               ;NO REGION AVAILABLE.
000003                                  ST.NRS= 3               ;NO REGION SUITABLE.
000004                                  ST.NON= 4               ;PROGRAM NOT KNOWN.
000005                                  ST.FAL= 5               ;LOAD FAILURE.
000006                                  ST.ALO= 6               ;STANDALONE.
                                ;
                                ; RECEIVE DATA MESSAGE TYPES.
                                ;
000020                                  .T.QUE= 1*HINIBL        ;QUESTION.
000040                                  .T.DEF= 2*HINIBL        ;DEFAULT QUESTION.
000060                                  .T.INF= 3*HINIBL        ;INFORMATION.
000100                                  .T.TER= 4*HINIBL        ;TERMINATION.
000120                                  .T.FAT= 5*HINIBL        ;FATAL ERROR.
000140                                  .T.SPL= 6*HINIBL        ;SPECIAL.
```

```
                                ;
                                ; COMMAND LENGTHS (IN BYTES.)
                                ;
000014                                  .C.ABO= 12.             ;ABORT PROGRAM.
000014                                  .C.DUS= 12.             ;GET DUST STATUS.
000050                                  .C.SUP= 40.             ;EXECUTE SUPPLIED PROGRAM.
000022                                  .C.LOC= 18.             ;EXECUTE LOCAL PROGRAM.
000034                                  .C.SND= 28.             ;SEND DATA.
000034                                  .C.RCD= 28.             ;RECEIVE DATA.
                                ;
                                ; RESPONSE LENGTHS (IN BYTES.)
                                ;
000014                                  .R.ABO= 12.             ;ABORT PROGRAM.
000026                                  .R.DUS= 22.             ;GET DUST STATUS.
000014                                  .R.SUP= 12.             ;EXECUTE SUPPLIED PROGRAM.
000020                                  .R.LOC= 16.             ;EXECUTE LOCAL PROGRAM.
000020                                  .R.SND= 16.             ;SEND DATA.
000020                                  .R.RCD= 16.             ;RECEIVE DATA.
```

```
                        .SBTTL  MODULE CODE
                        ;******************************************************************************
                        ;                                                                             ;
                        ;       INIT VALUES                                                           ;
                        ;       INIT RQDX                                                             ;
                        ;       XFER TO DISK?                                                         ;
                        ;               F FOR J = 1,CYCLE LIMIT                                       ;
                        ;                 MAITENANCE WRITE                                            ;
                        ;                 MAITENANCE READ                                             ;
                        ;                 CHECK DATA?                                                 ;
                        ;                       T CHECK                                               ;
                        ;               NEXT J                                                        ;
                        ;               T FOR J = 1,CYCLE LIMIT                                       ;
                        ;                 GET UNIT STATUS                                             ;
                        ;                       IF DRIVE IS NOT AVAILABLE, WAIT UNTIL IT IS           ;
                        ;                 DRIVE THERE?                                                ;
                        ;                 F  DROP                                                     ;
                        ;                    ALL  DRIVES DROPPED?                                     ;
                        ;                       T DROP MODULE                                         ;
                        ;                       F                                                     ;
                        ;                 T ONLINE                                                    ;
                        ;                 ONLINE?                                                     ;
                        ;                       T PICK BLOCK    IF RANDOM, GET RAND # MOD #           ;
                        ;                               ELSE INCREMENT                                ;
                        ;                                       IF LBN > LIMIT THEN LBN = 0           ;
                        ;                         WRITE                                               ;
                        ;                         READ                                                ;
                        ;                         CHECK DATA ?                                        ;
                        ;                               T CHECK                                       ;
                        ;                         AVAILABLE DRIVE(I)                                  ;
                        ;                               F TRY TO BRING ONLINE AGAIN                   ;
                        ;               NEXT J                                                        ;
                        ;                                                                             ;
                        ;******************************************************************************
```

```
1150                                    ;*****************************************************************
1151                                    ;                                                               *
1152                                    ;  START SEQUENCE                                              *
1153                                    ;                                                               *
1154                                    ;       IF THE CODE IS RESTARTED, CLEAR THE OLD ADDRESSES SO THE *
1155                                    ;       THE RQDX WILL GET REINITED.                             *
1156                                    ;                                                               *
1157                                    ;*****************************************************************
1158                                    ;
1159 001002                            START:
1160 001002  042767  100000  177634            BIC     #BIT15,DVICE    ; MUST BE CLEAR FOR LOOPING
1161 001010  005067  177130                    CLR     CDERCT          ; CLEAR DATA CHECK ERROR COUNT
1162 001014  012767  000017  177504            MOV     #PRINUM,PRNMSG  ; INITIALIZE PRINT WORD
1163 001022  016767  176766  177614            MOV     DVID1,DVICE     ; DVICE HAS DESIRED BITS SET
1164 001030  012767  177777  177620            MOV     #-1,TABLEW      ; FORCE TABLE SET UP FIRST TIME THROUGH
1165 001036  005067  177430                    CLR     CMDREF          ; COMMAND REF # = 0
1166 001042  104417  000000                    RAND$,BEGIN
1167 001046  016767  177002  177554            MOV     RANNUM,XSECI    ;
1168 001054  005067  177552                    CLR     XSECH           ; FOR RESTARTING (INITIAL SECTOR ADDR.
1169 001060  016767  176770  177546            MOV     RANNUM,RSECI    ;
1170 001066  005067  177544                    CLR     RSECH           ; FOR RESTARTING (INITIAL SECTOR ADDR)
1171 001072  016767  176710  177152            MOV     ADDR,RQDXSA     ; STORE IN SA REG
1172 001100  062767  000002  177144            ADD     #2,RQDXSA       ; RQDXSA HAS PROPER ADDRESS
1173 001106  005067  177404                    CLR     OLDPA           ; OLD PHYSICAL ADDRESS CLEARED
1174 001112  005067  177402                    CLR     OLDEA           ; OLD EXTENDED ADDRESS CLEARED
1175                                                                    ; THIS WILL FORCE A RQDX REINIT TO TAKE PLACE
1176 001116  052767  000001  177474            BIS     #BIT0,FLAGS     ; FLAGS = START
```

```
                                        ;*****************************************************************************
                                        ;                                                                            *
                                        ;  RESTART SEQUENCE                                                         *
                                        ;                                                                            *
                                        ;       CHECK THE ADDRESS OF THE RINGS TO SEE IF THEY WERE RELOCATED         *
                                        ;       IF THEY WERE, REINIT THE RQDX.                                       *
                                        ;                                                                            *
                                        ;       GET THE NEW ADDRESSES.  IF THE DISKLESS OPERATION IS DESIRED         *
                                        ;       THEN DO THE MAITENENCE WRITE AND READ.  ELSE DO THE WRITE            *
                                        ;       AND READ WITH A DRIVE.                                               *
                                        ;                                                                            *
                                        ;*****************************************************************************
001124                                  RESTRT:
001124  016767  000260'  177350          MOV     @RSPONC,VA          ; DID THE RINGS RELOCATE?
001132  104415  000000'  000502'          GETPA$,BEGIN, VA                    ;GET PHYSICAL ADDRESS FROM 16 BIT VA
001140  032767  001000  176710          BIT     @ADDR22,CONFIG      ; CHECK FOR 22BIT ADDRESSING
001146  001404                          BEQ     1$                  ; IF 18BIT ADDRESSING THEN BRANCH
001150  104416  000000'  000504'          MAP22$, BEGIN,PA                    ; GET 22-BIT ADDR FROM 18 BIT ADDR
001156  000416                          BR      2$
001160  016767  177320  177322  1$:     MOV     PA,LO22             ; MOV 18BIT ADDRESS INTO 22BIT SPACE
001166  006267  177314                  ASR     EA                  ; ADJUST EXTENDED BITS SO THEY LINE
001172  006267  177310                  ASR     EA                  ; UP AT BITS 0 & 1
001176  006267  177304                  ASR     EA
001202  006267  177300                  ASR     EA
001206  016767  177274  177276          MOV     EA,HI22
001214  026767  177270  177274  2$:     CMP     LO22,OLDPA          ; IF OLD PHYS ADDR NOT = NEW ONE?
001222  001004                          BNE     RESTR2              ; OR
001224  026767  177262  177266          CMP     HI22,OLDEA          ; IF OLD EXTN ADDR NOT = NEW ONE?
001232  001412                          BEQ     RESTR1              ; ; THEN
001234  016767  177250  177254  RESTR2: MOV     LO22,OLDPA          ; ;  SET THE OLD RING ADDR
001242  016767  177244  177250          MOV     HI22,OLDEA          ; ;  AND THE OLD EXTENED ADDR
001250  004767  000550                  CALL    INIPRT              ; ;  AND INIT THE RQDX
001254  005067  177366                  CLR     TRY                 ; ;  CLEAR RETRY COUNT
001260                                  RESTR1:                     ; ENDIF
001260  032767  000001  177532          BIT     @BIT0,FLAGS         ; IF FLAGS = START (ONLY DO DUP WHEN LOCATED HERE)
001266  001411                          BEQ     1$                  ; ; THEN
001270  042767  000001  177522          BIC     @BIT0,FLAGS         ; ;  FLAGS = START CLEARED
001276  016767  177214  177216          MOV     OLDPA,SAVPA         ; ;  SAVE PA
001304  016767  177210  177212          MOV     OLDEA,SAVEA         ; ;  SAVE EA
001312                                  1$:                         ; ENDIF
001312  032767  000010  176476          BIT     @SR,SUM,SR1         ; IF WE WANT THE REPORT?
001320  001006                          BNE     2$                  ; ; THEN
001322  026767  177200  176504          CMP     PRNMSG,PASCNT       ; ;  IF SO WE PRINT?
001330  001002                          BNE     2$                  ; ;  ; THEN (ONLY IF PASS COUNT IS RIGHT)
001332  004767  000404                  CALL    PRTSTS              ; ;  ;   CALL PRINT EOP STATUS
001336                                  2$:                         ; ;  ENDIF
                                        ;                           ; ENDIF
001336  012777  004770'  176444          MOV     @INTRPT,@VECTOR     ; GET VECTOR ADDRESS  SET POINTER
001344  012767  012462'  177130          MOV     @RBUF,VA            ; MAP THIS TO 22 BITS
001352  104415  000000'  000502'          GETPA$,BEGIN, VA                    ;GET PHYSICAL ADDRESS FROM 16 BIT VA
001360  032767  001000  176470          BIT     @ADDR22,CONFIG      ; CHECK FOR 22BIT ADDRESSING
001366  001420                          BEQ     22$                 ; BRANCH IF 18 BIT ADDRESSING
001370  104416  000000'  000504'          MAP22$, BEGIN,PA                    ; GET 22 BIT ADDR FROM 18 BIT ADDR
001376  016767  177210'  176522          MOV     PA,RBUFPA           ; FILL IN BUFFER ADDRESS FOR CMDATA
001404  016767  177206'  176516          MOV     EA,RBUFEA           ;                                  !!!!
001412  016767  177072  177344          MOV     LO22,R22PA          ; FILL IN BUFFER ADDRESS FOR READ CMD
```

```
1235 001420  016767  177066  177340          MOV     HI22,R22EA      ;
1236 001426  000424                          BR      33$
1237 001430  016767  177050  176470  22$:    MOV     PA,RBUFPA       ; FILL IN BUFFER ADDRESS FOR CRDATA
1238 001436  016767  177044  176464          MOV     EA,RBUFEA       ;     ""
1239 001444  016767  177034  177312          MOV     PA,R22PA        ; FILL IN 18BIT ADDRESS FOR READ CMD
1240 001452  006267  177030                  ASR     EA              ; ADJUST EXTENDED BITS
1241 001456  006267  177024                  ASR     EA
1242 001462  006267  177020                  ASR     EA
1243 001466  006267  177014                  ASR     EA
1244 001472  016767  177010  177266          MOV     EA,R22EA        ; FILL IN BUFFER ADDRESS FOR READ CMD
1245 001500  004767  001326          33$:    CALL    SETUP           ; FIND DRIVES/SET UP TABLE
1246 001504  005767  177134                  TST     DVICE           ; IF NO DRIVES TO DO
1247 001510  001002                          BNE     3$              ; : THEN
1248 001512  104410  000000'                END$,BEGIN              ;
1249 001516                          3$:                             ; ENDIF
1250 001516  026767  176774  176776          CMP     OLDPA,SAVPA     ; IF OLD_PHYS_ADDR = SAVE_PHYS_ADDR
1251 001524  001026                          BNE     4$              ; AND
1252 001526  026767  176766  176770          CMP     OLDEA,SAVEA     ; IF OLD_EXTN_ADDR = SAVE_EXTN_ADDR
1253 001534  001022                          BNE     4$              ; : THEN
1254 001536  012704  000656'                MOV     #TABLEW,R4      ; :  R4 = ADRS OF TABLE OF UNITNO AND PORTID
1255 001542  036467  000002  177074          BIT     2(R4),DVICE     ; :  IF DRIVE NOT DROPPED
1256 001550  001414                          BEQ     4$              ; :  AND
1257 001552  032767  000001  177074          BIT     #1,ISRX         ; :  IF RD = AVAILABLE
1258 001560  001010                          BNE     4$              ; :  AND
1259 001562  032767  000020  176226          BIT     #BIT4,SR1       ; :  IF DUP OPERATIONS = SELECTED
1260 001570  001404                          BEQ     4$              ; :  : THEN
1261 001572  004767  004074                  CALL    DODUP           ; :  :  CALL DO_DUP
1262 001576  004767  000222                  CALL    INIPRT          ; :  :  CALL INIT THE RQDX (MUST DO AFTER DUP)
1263                                                                 ; :  ENDIF
1264 001602                          4$:                             ; ENDIF
1265 001602                          LOOP1:                          ; BGNDO
1266 001602  012702  000001                  MOV     #1,R2           ; : SCALING FACTOR FOR RD
1267 001606  026767  177032  177040          CMP     DVICE,ISRX      ; : IF RD
1268 001614  001405                          BEQ     1$              ; : : THEN
1269 001616  016702  176176                  MOV     SR2,R2          ; : :  LOAD SCALING FACTOR
1270 001622  001002                          BNE     1$              ; : :
1271 001624  012702  000120                  MOV     #80.,R2         ; : :  DEFAULT  RD SCALING FACTOR TO 80.
1272 001630                          1$:                             ; : ENDIF
1273                                                                 ; : BGNDO
1274 001630  012704  000656'                MOV     #TABLEW,R4      ; : : R4 = ADRS OF TABLE OF UNITNO AND PORTID
1275 001634                          2$:                             ; : : BGNDO
1276 001634  036467  000002  177002          BIT     2(R4),DVICE     ; : : : IF DRIVE NOT DROPPED
1277 001642  001424                          BEQ     10$             ; : : : : THEN
1278 001644  036467  000002  177002          BIT     2(R4),ISRX      ; : : : :  IF DRIVE IS AN RX (ACCESS ONLY ONCE PER LOOP)
1279 001652  001403                          BEQ     4$              ; : : : :  AND
1280 001654  022702  000001                  CMP     #1,R2           ; : : : :  IF LAST TIME THROUGH
1281 001660  001015                          BNE     10$             ; : : : :  : THEN
1282 001662  010246                  4$:     MOV     R2,-(SP)        ; : : : :  :  R2 IS KILLED IN CYCLED AND DROP1
1283 001664  016746  176134                  MOV     SR4,-(SP)       ; : : : :  :  CONSECUTIVE ACCESSES TO DO TO A DEVICE
1284 001670                          5$:                             ; : : : :  :  BGNDO
1285 001670  004767  001504                  CALL    CYCLED          ; : : : :  :  :  DO A DISK CYCLE OPERATION
1286 001674  103402                          BCS     9$              ; : : : :  :  :  EXIT ON FAILURE
1287 001676  005316                          DEC     (SP)            ; : : : :  :  :  DECREMENT ACCESS CTR
1288 001700  003373                          BGT     5$              ; : : : :  :  DOUNTIL (IF 0 ONLY DO ONCE)
1289 001702  005726                  9$:     TST     (SP)+           ; : : : :  :  DON'T TOUCH C BIT
1290 001704  012602                          MOV     (SP)+,R2        ; : : : :  :
1291                                                                 ; : : : :  :  IF CYCLE NOT SUCCESSFUL
```

```
1292 001706  103002                        BCC     10$             ; : : : : :  : : THEN
1293 001710  004767  002310                JSR     PC,DROP1        ; : : : : :  : :  CALL DROP DRIVE
1294                                                                ; : : : : :  :  ENDIF
1295                                                                ; : : : : :  ENDIF
1296 001714                          10$:                           ; : : : : ENDIF
1297 001714  062704  000004                ADD     #4,R4           ; : : : : POINT TO NEXT ENTRY OF THE TABLE
1298 001720  026467  000002  176716        CMP     2(R4),DVICE     ; : : : DOUNTIL TABLE_ENTRY = DEVICE
1299 001726  101742                        BLOS    2$              ; : : :
1300 001730  005302                        DEC     R2              ; : : : DECREMENT SCALE_FACTOR
1301 001732  001336                        BNE     1$              ; : : DOUNTIL SCALE_FACTOR = 0
1302 001734  104413  000000'               ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                                    ;MONITOR SHALL TEST END OF PASS
1303 001740  000720                        BR      LOOP1           ; ENDDO
1304
1305
```

```
                                ;*****************************************************************************
                                ;                                                                            *
                                ;    PRINT EOP STATUS                                                        *
                                ;                                                                            *
                                ;                                                                            *
                                ;*****************************************************************************
                                ;
001742                          PRTSTS:
001742  062767  000017  176556         ADD     #PRINUM,PRNMSG  ; PRINT WORD IS INCREMENT
                                       ;*************************************************************
                                                               ;CONVERT SOFCNT TO ASCII AND
                                                               ;STORE AT ADR2
001750  104421  000000' 000042'       BTOD$,BEGIN,SOFCNT,ADR2
001756  000537'
                                       ;*************************************************************
001760  105067  176560                 CLRB    ADR2+5          ;
                                       ;*************************************************************
                                                               ;CONVERT HRDCNT TO ASCII AND
                                                               ;STORE AT ADR3
001764  104421  000000' 000044'       BTOD$,BEGIN,HRDCNT,ADR3
001772  000546'
                                       ;*************************************************************
001774  105067  176553                 CLRB    ADR3+5          ;
                                       ;*************************************************************
                                                               ;CONVERT CDERCT TO ASCII AND
                                                               ;STORE AT ADR1
002000  104421  000000' 000144'       BTOD$,BEGIN,CDERCT,ADR1
002006  000530'
                                       ;*************************************************************
002010  105067  176521                 CLRB    ADR1+5          ;
002014  104403  000000' 007764'       MSGN$,BEGIN,ERRPAS      ;ASCII MESSAGE CALL WITH COMMON HEADER
002022  000207                         RETURN                  ;
```

```
                        ;*****************************************************************************
                        ;                                                                            *
                        ;   INITIALIZE THE RQDX                                                      *
                        ;                                                                            *
                        ;               DO THE 4 STEPS FOR INITIALIZING THE RQDX.                    *
                        ;                                                                            *
                        ;       STEP 1 - CHECK FOR ERROR, STEP 1                                     *
                        ;                SEND VECTOR/4, INTERRUPT ENABLE, RING LEN'S = 0             *
                        ;                                                                            *
                        ;       STEP 2 - CHECK VECTOR ECHO, INTERRUPT ECHO,                          *
                        ;                        ERROR, STEP 2                                       *
                        ;                SEND PHYSICAL ADDRESS & PURGE = 0                           *
                        ;                                                                            *
                        ;       STEP 3 - CHECK RING LEN = 0, ERROR, STEP 3                           *
                        ;                SEND EXTENDED ADDRESS BITS                                  *
                        ;                                                                            *
                        ;       STEP 4 - CHECK STEP 4                                                *
                        ;                SEND CFAIL = 0 , GO AND BURST                               *
                        ;                                                                            *
                        ;*****************************************************************************
                        ;
002024                  INIPRT:                                 ; INITIALIZE
002024  005077  175756          CLR     @ADDR                   ; WRITE TO RQDXIP TO INIT RQDX
002030  012767  000260' 176444  MOV     #RSPONC,VA              ; VA -> RSPONC
002036  104415  000000' 000502' GETPA$,BEGIN, VA                ;GET PHYSICAL ADDRESS FROM 16 BIT VA
002044  032767  001000  176004  BIT     #ADDR22,CONFIG          ; CHECK FOR 22BIT ADDRESSING
002052  001404                  BEQ     11$                     ; IF 18BIT ADDRESSING THEN BRANCH
002054  104416  000000' 000504' MAP22$, BEGIN,PA               ; GET 22 BIT ADDR FROM 18-BIT ADDR
002062  000416                  BR      22$
002064  016767  176414  176416  11$:    MOV     PA,LO22         ; MOV 18BIT ADDRESS INTO 22BIT SPACE
002072  006267  176410          ASR     EA                      ; ADJUSTED EXTENDED BITS
002076  006267  176404          ASR     EA
002102  006267  176400          ASR     EA
002106  006267  176374          ASR     EA
002112  016767  176370  176372  MOV     EA,HI22
002120  005004                  22$:    CLR     R4              ; R4 IS USED IF AN ERROR IS DETECTED
002122  012702  000001          MOV     #1,R2                   ; R2 = STEP INDICATOR REG FOR MSG'S
002126  012701  027340          MOV     #TIMER,R1               ; SET TIME OUT LIMIT
002132  017700  176114          1$:     MOV     @RQDX$A,R0      ; R0 HAS RQDX$A DATA
002136  032700  100000          BIT     #<$A,ERR>,R0            ; CHECK FOR ERROR
002142  001003                  BNE     2$                      ; IF FOUND, GET OUT OF LOOP
002144  005501                  DEC     R1                      ;     TIME OUT?
002146  001371                  BNE     1$                      ; IF NOT, LOOP
002150  000404                  BR      4$                      ; IF DONE, CONTINUE
002152  012703  004000          2$:     MOV     #$A,S1,R3       ;   R3 = STEP 1 BIT
002156  000167  000474          JMP     ERROR1                  ; IF HERE, ERROR
                                ;4$:    CMP     #<$A,S1+$A,QB+$A,DIA>,R0; DID DATA COMPARE PROPERLY?
                                ;       BEQ     5$              ; IF SO, CONTINUE
002162  032700  004000          4$:     BIT     #$A,S1,R0       ; STEP 1 BIT SET?                       ;R3 002
002166  001002                  BNE     5$                      ; IF NE, YES                            ;R0 002
002170  000167  000456          JMP     ERROR3                  ;   REPORT ERROR
                                ;***************        STEP 1
002174  016705  175610          5$:     MOV     VECTOR,R5       ; VECTOR GIVEN
002200  006205                  ASR     R5                      ; SET TO APPROPRIATE VALUE
002202  006205                  ASR     R5                      ;       VECTOR/4
002204  052705  100200          BIS     #<$A.INI+BIT15>,R5      ; ENABLE INTERRUPTS, SET RING LEN'S ARE 0
002210  010500                  MOV     R5,R0                   ; STORE R5 IN R0 FOR SUBROUTINE
```

```
1387 002212  012703  004000                    MOV     #SA,S1,R3               ; R3 HAS STEP BIT FOR SUBROUTINE
1388 002216  004767  000332                    JSR     PC,SNDSTP               ; SEND STEP DATA
1389 002222  042705  100000                    BIC     #BIT15,R5               ; CLEAR MSB FOR COMPARE DATA
1390 002226  042700  000200                    BIC     #BIT07,R0               ; IF BIT07 ONLY SET
1391 002232  001404                            BEQ     6$                      ; : THEN
1392 002234  052700  010200                    BIS     #<SA,S2+BIT07>,R0       ; :  SET R0 TO REPORT THE ERROR
1393 002240  000167  000406                    JMP     ERROR3                  ; :  EXIT BLOCK TO REPORT ERROR
1394                                                                           ; ENDIF
1395                                                                           ;>>>>>>>>>>>>>>>>>     STEP 2
1396 002244  016700  176240          6$:       MOV     LO22,R0                 ; R0 GETS PHYSICAL ADDRESS
1397 002250  004767  000300                    JSR     PC,SNDSTP               ; SEND STEP DATA
1398 002254  042705  177400                    BIC     #177400,R5              ; HIGH BYTE CLEARED
1399 002260  020500                            CMP     R5,R0                   ; CHECK ECHO DATA
1400 002262  001402                            BEQ     7$                      ; IF OK, SKIP
1401 002264  000167  000362                    JMP     ERROR3                  ; IF NOT, REPORT ERROR
1402                                                                           ;>>>>>>>>>>>>>>>>>     STEP 3
1403 002270  016700  176216          7$:       MOV     HI22,R0                 ; ADJUST THE EXTENDED ADDRESS BITS
1404 002274  004767  000254                    JSR     PC,SNDSTP               ; SEND STEP DATA
1405 002300  012700  000254'                   MOV     #RSPONC+4,R0            ; R0 -> RING ENVELOP
1406                                                                           ;>>>>>>>>>>>>>>>>>     STEP 4
1407 002304                          8$:                                       ; BGNDO
1408 002304  005720                            TST     (R0)+                   ; : IF RING.ENTRY NOT = 0?
1409 002306  001154                            BNE     ERROR5                  ; : : THEN -> ERROR
1410 002310  022700  000266'                   CMP     #RSPONC+6,R0            ; DOUNTIL R0 POINT PAST THE RINGS?
1411 002314  001373                            BNE     8$                      ;
1412 002316  012777  000001  175726            MOV     #SA,GO,@RQDXSA          ; SEND DATA TO RQDX/DONE WITH THE INIT
1413 002324  012767  000376'  176150            MOV     #CMPACK,VA              ; GET COMMAND PACKET PA AND EA
1414 002332  104415  000000'  000502'          GETPA$,BEGIN, VA                ;GET PHYSICAL ADDRESS FROM 16-BIT VA
1415 002340  032767  001000  175510            BIT     #ADDR22,CONFIG          ; CHECK FOR 22BIT ADDRESSING
1416 002346  001404                            BEQ     88$                     ; IF 18BIT ADDRESSING THEN BRANCH
1417 002350  104416  000000'  000504'          MAP22$, BEGIN,PA                ; GET 22-BIT ADDR FROM 18-BIT ADDR
1418 002356  000416                            BR      99$
1419 002360  016767  176120  176122  88$:      MOV     PA,LO22                 ; MOV 18BIT ADDRESS INTO 22BIT SPACE
1420 002366  006267  176114                    ASR     EA                      ; ADJUST EXTENDED BITS
1421 002372  006267  176110                    ASR     EA
1422 002376  006267  176104                    ASR     EA
1423 002402  006267  176100                    ASR     EA
1424 002406  016767  176074  176076            MOV     EA,HI22
1425 002414  016767  176070  175642  99$:      MOV     LO22,COMMND             ; STORE ADDRESS IN THE RING
1426 002422  016767  176064  175636            MOV     HI22,COMMND+2           ; MOVE ADJUSTED EA INTO RING
1427 002430  012767  000276'  176044            MOV     #RSPACK,VA              ; GET RESPONCE PACKET PA AND EA
1428 002436  104415  000000'  000502'          GETPA$,BEGIN, VA                ;GET PHYSICAL ADDRESS FROM 16-BIT VA
1429 002444  032767  001000  175404            BIT     #ADDR22,CONFIG          ; CHECK FOR 22BIT ADDRESSING
1430 002452  001404                            BEQ     66$                     ; IF 18BIT ADDRESSING THEN BRANCH
1431 002454  104416  000000'  000504'          MAP22$, BEGIN,PA                ; GET 22 BIT ADDR FROM 18-BIT ADDR
1432 002462  000416                            BR      77$
1433 002464  016767  176014  176016  66$:      MOV     PA,LO22                 ; MOV 18BIT ADDRESS INTO 22BIT SPACE
1434 002472  006267  176010                    ASR     EA
1435 002476  006267  176004                    ASR     EA
1436 002502  006267  176000                    ASR     EA
1437 002506  006267  175774                    ASR     EA
1438 002512  016767  175770  175772            MOV     EA,HI22
1439 002520  016767  175764  175552  77$:      MOV     LO22,RSPONC             ; STORE ADDRESS IN THE RING
1440 002526  016767  175760  175526            MOV     HI22,RSPONC+2           ; MOVE ADJUSTED EA INTO RING
1441 002534  012777  004770'  175246            MOV     #NTRUPT,@VECTOR         ; STORE INTERRUPT ADDRESS IN VECTOR
1442 002542  005067  176100                    CLR     TR1                     ; CLEAR TR1 SO DRIVE WILL GO BACK ONLINE IF NECESSARY
;
1443 002546  004767  002042                    CALL    SCC                     ; SET CONTROLLER CHARACTERISTICS EVERY TIME WE RE IN
IT RQDX1
```

```
1444 002552  000207                    RTS     PC                  ;EXIT INITIALIZE
```

```
1446
1447                                    ;*****************************************************************************
1448                                    ;
1449                                    ;   SEND STEP DATA
1450                                    ;
1451                                    ;       INPUT:  R0 HAS DATA TO BE SENT TO RQDX FOR STEP
1452                                    ;               R3 HAS PREVIOUS STEP FLAG SET
1453                                    ;
1454                                    ;       OUTPUT: R0 HAS DATA SENT FROM RQDX TO HOST FOR ECHO AND NEXT STEP
1455                                    ;               R3 HAS CURRENT STEP FLAG SET
1456                                    ;
1457                                    ;*****************************************************************************
1458                                    ;
1459 002554  016701  175230            SNDSTP: MOV     VECTOR,R1       ;
1460 002560  012721  002600'                    MOV     #INTA,(R1)+     ;SET UP INTERRUPT HANDLER ADDRESS
1461 002564  116711  175223                    MOVB    BR1,(R1)        ;SET PRIORITY LEVEL
1462 002570  010077  175456                    MOV     R0,@RQDXSA      ;SEND STEP1 WRITE FORMMATED DATA
1463
1464 002574  104400  000000'                    EXIT$,BEGIN                     ;EXIT TO MONITOR, MODULE WAIT FOR INTERRUPT.
1465
1466 002600                             INTA:
1467                                            ;-------------------------------------------------------------------
     002600  000004  000000' 002606'            PIRQ$,BEGIN,3$          ; QUEUE UP TO CONTINUE AT 3$ AND RTI
                                                ;-------------------------------------------------------------------
1468 002606                             3$:
1469 002606  017700  175440                    MOV     @RQDXSA,R0      ; GET STEP N FORMATTED DATA
1470 002612  032700  100000                    BIT     #SA.ERR,R0      ; IF NO HARDWARE ERROR
1471 002616  001017                            BNE     ERROR1          ; : THEN
1472 002620  005202                            INC     R2              ; SET STEP REGISTER
1473 002622  006303                            ASL     R3              ; R3 HAS STEP BIT PROPERLY SET
1474 002624  030300                            BIT     R3,R0           ; IF STEP N NOT SET?
1475 002626  001002                            BNE     4$              ; : THEN
1476 002630  000167  000020                    JMP     ERROR2          ; :   NOT CORRECT STEP, ERROR
1477 002634                             4$:                             ; ENDIF
1478 002634  040300                            BIC     R3,R0           ; CLEAR THE STEP BIT, FOR COMPARE
1479 002636  000207                            RTS     PC              ; RETURN
```

```
                                ;*****************************************************************************
                                ;                                                                            ;
                                ;  ERROR 1                                                                   ;
                                ;       PRINT AN ERROR REPORTED BY THE RQDX DIAGNOSTICS                       ;
                                ;                                                                            ;
                                ;  ERROR 2                                                                   ;
                                ;       PRINT VALUE OF THE RQDXSA WHEN THE STEP BIT WAS NOT SET               ;
                                ;                                                                            ;
                                ;  ERROR 3                                                                   ;
                                ;       PRINT VALUE OF THE RQDXSA WHEN THE ECHO WAS NOT SET CORRECTLY         ;
                                ;                                                                            ;
                                ;       INPUT:  R0 -> RQDXSA                                                 ;
                                ;               R2 = STEP COUNT                                              ;
                                ;                                                                            ;
                                ;       OUTPUT: THE RETRY COUNT IS INCREMENTED                               ;
                                ;               IF THE RETRY COUNT > RETRY LIMIT, END MODULE                 ;
                                ;                                                                            ;
                                ;  ERROR 5                                                                   ;
                                ;       RING WASN'T ALL ZERO -> FATAL ERROR -> DROP MODULE                    ;
                                ;                                                                            ;
                                ;*****************************************************************************
002640                          ERROR5:
002640  104403  000000' 010044'        MSGN$,BEGIN,ZERO        ;ASCII MESSAGE CALL WITH COMMON HEADER
002646  104410  000000'                END$,BEGIN              ;
                                ERROR3: INC     R4              ;R4 = 3 FOR ERROR3
                                ERROR2: INC     R4              ;R4 = 2 FOR ERROR2
                                ERROR1: INC     R4              ;R4 = 1 FOR ERROR1
002660  010267  175630                MOV     R2,NUM          ;STORE STEP REG IN A NUMBER FOR CONVRT
                                        ;*************************************************************
                                                                ;CONVERT NUM TO ASCII AND
002664  104420  000000' 000514'        OTOA$,BEGIN,NUM,ADR2    ;STORE AT ADR2
002672  000537'
                                        ;*************************************************************
002674  017767  175352  175612        MOV     @RQDXSA,NUM     ;STORE VALUE IN A NUMBER
                                        ;*************************************************************
                                                                ;CONVERT NUM TO ASCII AND
                                                                ;STORE AT ADR1
002702  104420  000000' 000514'        OTOA$,BEGIN,NUM,ADR1
002710  000530'
                                        ;*************************************************************
002712  105067  175620                CLRB    ADR2+1          ;
                                        ;
002716  005304                        DEC     R4              ;ERROR 1?
002720  001003                        BNE     1$              ;IF NOT, CHECK IF IT IS THE NEXT ERROR
002722  104403  000000' 007656'        MSGN$,BEGIN,INITE1      ;ASCII MESSAGE CALL WITH COMMON HEADER
002730                          1$:
002730  005304                        DEC     R4              ;ERROR 2?
002732  001003                        BNE     2$              ;IF NOT, CHECK IF IT IS THE NEXT ERROR
002734  104403  000000' 007702'        MSGN$,BEGIN,INITE2      ;ASCII MESSAGE CALL WITH COMMON HEADER
002742                          2$:
002742  005304                        DEC     R4              ;ERROR 3?
002744  001003                        BNE     3$              ;IF NOT, CHECK IF IT IS THE NEXT ERROR
002746  104403  000000' 007710'        MSGN$,BEGIN,INITE3      ;ASCII MESSAGE CALL WITH COMMON HEADER
002754                          3$:
                                ;*****************************************************************************
```

```
                                                                ;CONVERT ADDR TO ASCII AND
                                                                ;STORE AT ADR3
     002754  104420  000000' 000006'          OTOA$,BEGIN,ADDR,ADR3
     002762  000546'
                                                ;*****************************************************************
1527                                            ;*****************************************************************
     002764  104405  000000' 000000          HRDER$,BEGIN,NULL       ;
                                                ;*****************************************************************
1528 002772  104403  000000' 007664'          MSGN$,BEGIN,INITER      ;ASCII MESSAGE CALL WITH COMMON HEADER
1529 003000  005267  175642                   INC     TRY             ;INCREMENT RETRY COUNT
1530 003004  022767  000004  175634          CMP     #RLIM,TRY       ;IS THE RETRY COUNT EXCEEDED?
1531 003012  001402                           BEQ     6$              ;IF SO, END IT
1532 003014  000167  175762                   JMP     START           ;IF NOT, TRY AGAIN
1533 003020                           6$:
     003020  104403  000000' 010040'          MSGN$,BEGIN,ABORT       ;ASCII MESSAGE CALL WITH COMMON HEADER
1534 003026  104410  000000'                  END$,BEGIN              ;
1535
```

```
1537                                    ;*******************************************************************************
1538                                    ;
1539                                    ;   SET UP
1540                                    ;
1541                                    ;       GO FIND OUT WHAT DRIVES ARE OUT THERE.
1542                                    ;       A TABLE IS FILLED WITH UNIT NUMBERS(MAX IS 16)
1543                                    ;
1544                                    ;       THIS SHOULD ONLY BE DONE AT THE VERY BEGINNING OF RUNNING
1545                                    ;       THIS DECX MODULE; THEN NOT RUN AGAIN.
1546                                    ;
1547                                    ;       INPUT:  DVICE HAS APPROPRIATE BITS SET.  THE # OF BITS =
1548                                    ;                       # OF DRIVES WANTED TO TEST.
1549                                    ;               POSITION OF BITS = WHICH DRIVE IN THE SYSTEM IS DESIRED.
1550                                    ;
1551                                    ;*******************************************************************************
1552                                    ;
1553 003032                            SETUP:
1554 003032  005767  175620                    TST     TABLEW                  ; ONLY DO SETUP THE FIRST TIME THROUGH
1555 003036  002133                            BGE     100$                    ; (USE DRIVES FOUND AT BEGINNING)
1556                                    ;       CALL    SCC                     ; CALL SET_CONTROLLER_CHARACTERISTICS
1557 003040  012703  000001                    MOV     #1,R3                   ; INITIAL PORTID VALUE
1558 003044  012704  000656'                   MOV     #TABLEW,R4              ; R4 -> TABLEW
1559 003050  005014                            CLR     (R4)                    ;
1560 003052                            1$:                                     ;
1561 003052  012746  000024                    MOV     #24, -(SP)              ; WORK = RETRY LIMIT
1562 003056  004767  001474            3$:     JSR     PC,GTSTAT               ; GET STATUS, GET NEXT UNIT NUMBER
1563 003062  103011                            BCC     6$                      ; OK, CONTINUE
1564 003064  005316                            DEC     (SP)                    ;
1565 003066  001373                            BNE     3$                      ;
1566 003070  005726                            TST     (SP)+                   ;
1567 003072  056467  000002  175544            BIS     2(R4),DVICE             ; ASSURE REPORT WILL BE PRINTED
1568 003100  004767  001130                    JSR     PC,DROP2                ; DROP THE DRIVE
1569 003104  000476                            BR      17$                     ; TRY NEXT UNIT
1570 003106  005726                    6$:     TST     (SP)+                   ; POP RETRY COUNT OFF STACK
1571 003110  016714  175166            7$:     MOV     P.UNIT+RSPACK,(R4)      ; UNIT NUMBER FROM RESPONCE PACKET
1572 003114  001015                            BNE     10$                     ; UNIT NUMBER RETURNED, CONTINUE
1573 003116  022703  000001                    CMP     #1,R3                   ; FIRST TIME THROUGH MAY BE UNIT 0
1574 003122  001412                            BEQ     10$
1575                                    ;
1576                                    ; NO UNITS ARE LEFT, BUT BITS ARE STILL SET IN DVICE
1577                                    ;
1578 003124  030367  175514            8$:     BIT     R3,DVICE                ; IS THIS ONE SET?
1579 003130  001402                            BEQ     9$                      ; NO, CONTINUE
1580 003132  004767  001106                    JSR     PC,DROP3                ; DROP UNIT AND WARN
1581 003136  062704  000004            9$:     ADD     #4,R4                   ; MAINTAIN TABLE POINTER FOR DROP3
1582 003142  006303                            ASL     R3                      ; TRY NEXT BIT
1583 003144  001367                            BNE     8$                      ;IF ANY ARE LEFT, TRY AGAIN
1584 003146  000467                            BR      100$                    ; OTHERWISE, EXIT
1585
1586 003150  126727  175154  000006    10$:    CMPB    P.UNIT+6+RSPACK,#6      ; MEDIA ID IS 6 FOR RD, 7 FOR RX
1587 003156  001416                            BEQ     17$                     ; NO - IT IS AN RX
1588 003160  050367  175470                    BIS     R3,ISRX                 ; YES - SET RD BIT
1589 003164  122737  000025  000041            CMPB    #25,@#41                ; IS LOAD MEDIUM DU2
1590 003172  001010                            BNE     17$                     ; NO - SKIP CHECKING
1591 003174  121437  000040                    CMPB    (R4),@#40               ;
1592 003200  001005                            BNE     17$                     ;
1593 003202  040367  174614                    BIC     R3,SR3                  ; CLEAR WRITE BIT FOR LOAD DEVICE
```

```
RQAD DEC X11 SYSTEM EXERCISER M MACRO M1.00  09 APR 84 16:51  PAGE 35 1                                                SEQ 40
MODULE CODE

1594 003206  104405  000000' 010074'          MSGN$,BEGIN,BUNIT       ;ASCII MESSAGE CALL WITH COMMON HEADER
1595 003214  056467  000000  175422  177$:    BIT     2(R4),DVICE             ; ONLY PRINT MESSAGE IF IT IS VALID
1596 003222  001422                           BEQ     177$                    ; UNIT NOT TO BE TESTED
1597 003224  011467  175264                   MOV     (R4),NUM                ; WE HAVE A GOOD UNIT   PRINT MESSAGE
1598                                           ;*************************************************************
                                               ;CONVERT NUM TO ASCII AND
                                               ;STORE AT ADR1
     003230  104420  000000' 000514'          OTOA$,BEGIN,NUM,ADR1
     003236  000530'
                                               ;*************************************************************
1599 003240  105067  175272                   CLRB    ADR1+1                  ;
1600 003244  032767  000002  174544  173$:    BIT     #BIT1,SR1               ; NO READ ACCESS - READ ONLY MESSAGE
1601 003252  001410                           BEQ     175$                    ;
1602 003254  056667  000002  174540           BIT     2(R4),SR3               ; UNIT HAS WRITE TURNED OFF
1603 003262  001404                           BEQ     175$                    ;
1604 003264  104405  000000' 010050           MSGN$,BEGIN,RDWR1       ;ASCII MESSAGE CALL WITH COMMON HEADER
1605 003272  000403                           BR      177$                    ;
1606 003274                           175$:
     003274  104405  000000' 010062'          MSGN$,BEGIN,RDONLY      ;ASCII MESSAGE CALL WITH COMMON HEADER
1607 003302  062704  000004  177774  177$:    ADD     #4,R4                   ; BUMP R4 TO NEXT TABLE POINTER
1608 003306  016414  177774                   MOV     -4(R4),(R4)             ; SET NEXT UNIT NUMBER = CURRENT ONE
1609 003312  005214                           INC     (R4)                    ;
1610 003314  006503                           ASL     R3                      ; NEXT BIT IN DVICE
1611 003316  020367  175322                   CMP     R3,DVICE                ;IF =, WE'RE DONE
1612 003322  101653                           BLOS    1$                      ; (BIT 15 IN DVICE IS EXPLICITLY 0)
1613 003324  005014                           CLR     (R4)                    ; TAKE EXTRA NUMBER OUT OF TABLE
1614 003326  000207                   100$:   RTS     PC                      ; ...AND BACK.
```

```
                        ;*****************************************************************************
                        ;
                        ;   TSTOFL
                        ;       TEST TO SEE WHAT KIND OF AN OFFLINE CONDITION HAS OCCURED.
                        ;
                        ;*****************************************************************************
003350  022700  000005  TSTOFL: CMP     @ST.OFL,R0          ; WAS THE DRIVE FOUND OFF LINE?
003354  001403                  BEQ     10$                 ; CHECK WHAT KIND OF OFFLINE
003356  022700  000015          CMP     @ST.DRV,R0          ; WAS IT A DRIVE ERROR?  > SDI?
003362  001012                  BNE     13$                 ;IF IT WAS NOT, ERROR (DROP DRIVE)
                        ;
003364                  10$:    ; IF ANY OF FOLLOWING BITS SET
                        ;        NO VOLUME MOUNTED, UNIT DISABLED BY FIELD SERVICE
                        ;        OR DUPLICATE UNIT NUMBER OR UNIT INOPERATIVE
                        ;
003364  032767  000740  174736          BIT     @#<SC.NVL+SC.DIS+SC.DUP+SC.IOP>,@P.STS+RSPACK
003372  001004                  BNE     12$                 ;IF SO, EXIT
                                                            ; ANY OTHER DATA?
                        ;
003374  032767  177000  174726          BIT     @#C<SC.NVL+SC.DIS+SC.DUP+SC.IOP+ST.MSK>,@P.STS+RSPACK
003402  001003                  BNE     13$                 ;IF SO, DROP
                        ;
003404  000241          12$:    CLC                         ;CLEAR CARRY   NO ERROR
003406  000403                  BR      14$
                        ;
003410  000261          13$:    SEC                         ;SET CARRY, DRIVE WAS FOUND TO BE OFFLINE
                                                            ; OR ANOTHER ERROR
003412  004767  002040          JSR     PC,ERRORH           ;REPORT ERROR
003416  000207          14$:    RTS     PC                  ;RETURN
                        ;
```

```
                                                ;*******************************************************************
                                                ;                                                                  ;
                                                ;   CYCLE DISK                                                     ;
                                                ;                                                                  ;
                                                ;               DO GET STATUS COMMANDS TO ASSURE THAT THE DRIVE    ;
                                                ;                       IS AVAILABLE (FOR DUAL PORTING)            ;
                                                ;               CHECK DRIVE TO BE ONLINE                           ;
                                                ;               IF TRUE                                            ;
                                                ;                       PICK THE BLOCK                             ;
                                                ;                       WRITE                                      ;
                                                ;                       READ                                       ;
                                                ;                       DATA CHECK                                 ;
                                                ;                       MAKE THE DRIVE AVAILABLE                   ;
                                                ;               ELSE DROP DRIVE                                    ;
                                                ;                                                                  ;
                                                ;*******************************************************************
                                                ;
     003400                                     CYCLED:
                                                                                        ;
                                                                                        ; CHECK IF WE DO ONLINE FOR THE FIRST TIME.
                                                                                        ;
     003400  005767  175242                             TST     TRY                     ; IF TRY HAS NOT SET MSB
     003404  100420                                     BMI     16$                     ; : THEN
     003406  004767  001240                     10$:    JSR     PC,ONLINE               ; :  DO AND ONLINE COMMAND
     003412  103571                                     BCS     24$                     ; :  IF CARRY WAS SET, TRY AGAIN
     003414  016767  174724  175220                     MOV     P.UNSZ+2+RSPACK,UNSZH   ; :   IS THE UNIT SIZE HI ADDRESS
     003422  016767  174714  175210                     MOV     P.UNSZ+RSPACK,UNSZL     ; :   GET UNIT SIZE/IS IT < 0?
     003430  001006                                     BNE     16$                     ; :   IF NOT ZERO, CONTINUE WITH ITERATION
     003432  005767  175204                             TST     UNSZH                   ; :   IS UNSZH ALSO 0?
     003436  001760                                     BEQ     CYCLED                  ; :   IF 0, TRY TO BRING ONLINE AGAIN
     003440  012767  100000  175200                     MOV     #100000,TRY             ; :   TRY = MSB SET (INITIAL ONLINE IS DONE)
     003446                                     16$:                                    ; ENDIF
     003446  016700  174470                             MOV     WBUFSZ,R0       ; WBUFSZ IN R0 AS A LIMIT
     003452  005001                                     CLR     R1              ; R1 = # OF BLOCKS
     003454                                     18$:                            ; BGNDO
     003454  005201                                     INC     R1              ; : INCREMENT THE # OF BLOCKS
     003456  162700  000400                             SUB     #400,R0         ; : DECREMENT A BLOCK
     003462  100374                                     BPL     18$             ; DOUNTIL # OF BLOCKS < 0
     003464  160167  175150                             SUB     R1,UNSZL        ; ADJUST THE UNIT SIZE
                                                                                ;  ( NOW PICK WHICH BLOCK TO WRITE TO )
     003470  012702  000630                             MOV     #XSECL,R2       ; RX LOW SECTOR NUMBER
     003474  036467  000002  175152                     BIT     2(R4),ISRX      ;
     003502  001001                                     BNE     185$            ;
     003504  022222                                     CMP     (R2)+,(R2)+     ; BUMP TO DSLCL
     003506  000240                             185$:   NOP
     003510  004767  000264                             CALL    PICKBK          ; ELSE SELECT A BLOCK TO TEST
     003514  004767  001???                             CALL    GTSTAT          ; DID WE NOT GET THE DRIVE ONLINE?
     003520  103732                                     BCS     10$             ;IF NOT, GO BACK TO TOP AND TRY AGAIN
     003522  022700  000006                             CMP     #ST.AVL,R0      ;IS IT AVAILABLE?
     003526  001727                                     BEQ     10$             ;IF SO, GO BACK TO TOP AND TRY AGAIN
     003530  000240                                     NOP
     003532  032767  000002  174256                     BIT     #SR.RON,SR1     ; IF WRITE ACCESS
     003540  001506                                     BEQ     20$             ; AND
     003542  036467  000002  174252                     BIT     2(R4),SR3       ; IF UNIT WRITE IS DESIRED
     003550  001502                                     BEQ     20$             ; : THEN
     003552  104414  000000                             GWBUF$, BEGIN           ;GET WRITE BUFFER INFORMATION
     003556  016767  174352  175206                     MOV     WBUFPA,W18PA    ; :   COPY 18 BIT ADDRESS
     003564  016767  174346  175202                     MOV     WBUFPA+2,W18EA  ; :
```

```
1704 003572  032767  001000  174256          BIT     #ADDR22,CONFIG  ; :  CHECK FOR 22BIT ADDRESSING
1705 003600  001404                          BEQ     111$            ; :  IF 18BIT ADDRESSING THEN BRANCH
1706 003602  104416  000000' 000772'         MAP22$, BEGIN,W18PA             ; GET 22 BIT ADDR FROM 18 BIT ADDR
1707 003610  000416                          BR      222$
1708 003612  016767  175154  175156  111$:   MOV     W18PA,W22PA     ; :  MOV 18BIT ADDRESS INTO 22BIT SPACE
1709 003620  006267  175150                  ASR     W18EA           ; :  ADJUST EXTENDED BITS
1710 003624  006267  175144                  ASR     W18EA
1711 003630  006267  175140                  ASR     W18EA
1712 003634  006267  175134                  ASR     W18EA
1713 003640  016767  175130  175132          MOV     W18EA,W22EA
1714
1715 003646  004767  000544          222$:   JSR     PC,WRITE        ; :  WRITE THE DATA
1716 003652  103003                          BCC     19$             ; :  IF NOT OK
1717                                                                 ; :  : THEN
1718 003654  004767  001556                  JSR     PC,ERRORH       ; :  :  HARD ERROR (BCS 22$ )
1719 003660  000445                          BR      22$             ; :  :  EXIT
1720 003662                          19$:                            ; :  ENDIF
1721 003662  004767  000600                  CALL    READ            ; :  READ A BLOCK
1722 003666  103003                          BCC     195$            ; :  IF NOT OK
1723                                                                 ; :  : THEN
1724 003670  004767  001542                  JSR     PC,ERRORH       ; :  :  HARD ERROR
1725 003674  000437                          BR      22$             ; :  :  EXIT
1726 003676                          195$:                           ; :  ENDIF
1727 003676  032767  004000  174112          BIT     #SR.CMP,SR1     ; :  IF DO A DATA COMPARE
1728 003704  001053                          BNE     22$             ; :  : THEN
1729 003706  104412  000000' 000126'         CDATA$,BEGIN,RBUFPA     ; REQUEST FOR MONITOR TO CHECK DATA
     003714  003716'                         .+2                     ; IF ERROR, CONTINUE
1730 003716  000240                          NOP
1731 003720  000240                          NOP
1732 003722  000240                          NOP
1733
1734 003724  000423                          BR      22$             ; :  :
1735 003726  011367  174562          197$:   MOV     (R4),NUM        ; :  :
1736                                         ;*************************************************************************
                                             ;CONVERT NUM TO ASCII AND
                                             ;STORE AT ADR1
     003732  104420  000000' 000514'         OTOA$,BEGIN,NUM,ADR1
     003740  000530'
                                             ;*************************************************************************
1737 003742  105067  174570                  CLRB    ADR2+1          ; :  :
1738 003746  104403  000000' 010116'         MSGN$,BEGIN,CHKRPT      ;ASCII MESSAGE CALL WITH COMMON HEADER
1739 003754  000407                          BR      22$             ; :  ENDIF
1740 003756  000240                  20$:    NOP                     ; : ELSE
1741 003760  004767  000502                  CALL    READ            ; :  READ ONLY - NO DATA CHECK
1742 003764  103003                          BCC     22$             ; :  IF ERROR
1743                                                                 ; :  : THEN
1744 003766  004767  001444                  JSR     PC,ERRORH       ; :  :  HARD ERROR
1745 003772  000400                          BR      22$             ; :  :  AND EXIT; BCS 22$
1746 003774                          22$:                            ; :  ENDIF
1747                                                                 ; ENDIF
1748 003774  000241                          CLC                     ; EVERY THING WAS OK
1749 003776  000207                  24$:    RTS     PC              ; RETURN
1750
1751
```

```
                                        ;*****************************************************************************
                                        ;                                                                            *
                                        ;   PICK A BLOCK TO WRITE TO.                                               *
                                        ;                                                                            *
                                        ;       PICK THE NEXT SEQUNTIAL BLOCK (DEFAULT) OR TAKE ONE AT RANDOM.       *
                                        ;                                                                            *
                                        ;       OUTPUT: FILL SECH & SECL (CURRENT SECTOR ADDR)                       *
                                        ;                                                                            *
                                        ;*****************************************************************************
004000                                  PICKBK:
004000  032767  002000  174010                  BIT     #SR.SEQ,SR1     ;CHECK SR1 FOR RANDOM ACCESS MODE
004006  001020                                  BNE     RANACC          ;BR IF RANDOM  ACCESS
                                        ;
                                        ;  GENERATE DISK ADDRESS BY SEQUENTIAL ADDRESSING
                                        ;
004010                                  SEQACC:
004010  005212                                  INC     (R2)            ;INCREMENT THE SECTOR ADDRESS
004012  001408                                  BEQ     16$             ;BR IF ZERO
004014  021267  174620                          CMP     (R2),UNSZL      ;OVER LIMIT?
004020  103412                                  BLO     18$             ;BR IF LOWER
004022  000402                                  BR      17$             ;SKIP THE INCREMENT
004024                                  16$:                            ;
004024  005262  000002                          INC     2(R2)           ;INCREMENT SECTOR HIGH ADDRESS
004030                                  17$:                            ;
004030  026267  000000  174604                  CMP     2(R2),UNSZH     ;OVER LIMIT?
004036  103403                                  BLO     18$             ;BR IF LOWER
004040  005012                                  CLR     (R2)            ;RESET THE STARTING SECTOR ADDRESS
004042  005062  000002                          CLR     2(R2)           ;
004046                                  18$:                            ;
004046  000207                                  RTS     PC              ;
```

```
                                ;*****************************************************************************
                                ;
                                ;  GENERATE DISK ADDRESS BY RANDOM ADDRESSING
                                ;
                                ;*****************************************************************************
                                ;
                                RANACC:
004050                                  RAND$,BEGIN
004050  104417  000000'                  MOV     RANNUM,-(SP)    ;GENERATE THE SECTOR ADDRESS
004054  016746  173774
004060  104417  000000'                  RAND$,BEGIN
004064  016746  173764                  MOV     RANNUM,(SP)     ;GENERATE THE SECTOR ADDRESS
                                ;
                                ;       ADJUST HI ADDRESS FIRST
                                ;
004070  000241                          CLC                     ; CLEAR CARRY FOR ROTATE
004072  042716  100000                  BIC     #100000,(SP)    ; CLEAR UPPER BIT MAKES SURE VALUE'S +
004076  012662  000002                  MOV     (SP)+,2(R2)     ; STORE IN SECTOR HI ADDRESS
004102  005767  174534                  TST     UNSZH           ; IS THE MAX SIZE 0?
004106  001430                          BEQ     3$              ;IF 0, GET LOW SECTOR ADDRESS
                                ;
                                ;  ***  UNSZH > 0 IF CODE FALLS THROUGH HERE
                                ;
004110  016700  174526                  MOV     UNSZH,R0        ; R0 = MAX VALUE
004114  005100                          COM     R0              ; R0 COMPLEMENT, NOW FIND MS ZERO
004116  012701  100000                  MOV     #100000,R1      ; R1 IS INDEX INTO MAX VALUE
004122  030100                  1$:     BIT     R1,R0           ; HAVE 0 YET?
004124  001403                          BEQ     2$              ;IF 1ST 0 REACHED, CLEAR REST OF THE BITS
004126  000241                          CLC                     ; CLEAR CARRY FOR ROR
004130  006001                          ROR     R1              ; POINT TO NEXT BIT
004132  000773                          BR      1$              ; BRANCH TO TEST AGAIN
004134  040100                  2$:     BIC     R1,R0           ; CLEAR REST OF THE BITS
004136  000241                          CLC                     ; CLEAR CARRY FOR ROR
004140  006001                          ROR     R1              ;IF R1 ROTATES INTO CARRY, R1 = 0
004142  001374                          BNE     2$              ;IF R1 NOT 0, MORE BITS TO CLEAR
004144  040062  000002                  BIC     R0,2(R2)        ;  CLEAR UPPER BITS OF HIGH SECTOR VALUE
004150  026267  000002  174464          CMP     2(R2),UNSZH     ;IF THE HIGH SECTOR VALUE > MAX VALUE?
004156  002420                          BLT     7$              ;IF <, EXIT
004160  001405                          BEQ     4$              ;IF =, TEST LOW ORDER VALUE
004162  006262  000002                  ASR     2(R2)           ;  SECH = SECH/2 - CAN'T BE > MAX NOW
004166  000414                          BR      7$              ;EXIT
                                ;
                                ;       GET LOW SECTOR ADDRESS
                                ;
004170  005062  000002          3$:     CLR     2(R2)           ;CLEAR HI SECTOR SIZE
004174  005767  174440          4$:     TST     UNSZL           ;IS THE HIGHEST POSSIBLE = 0?
004200  001406                          BEQ     6$              ;IF TRUE, DON'T DO LOOP
004202  166716  174436          5$:     SUB     UNSZL,(SP)      ;  ELSE, SECL = SECL - UNSZL (ADJUST)
004206  103375                          BCC     5$              ;  IF UNSZL > SECL, LOOP
004210  066716  174424                  ADD     UNSZL,(SP)      ;    ELSE SUBTRACTED ONCE TOO OFTEN
004214  000401                          BR      7$              ;    AND EXIT
004216  005016                  6$:     CLR     (SP)            ;HIGHEST POSSIBLE = 0
004220  011612                  7$:     MOV     (SP)+,(R2)      ;SAVE LO SECTOR ADDRESS
004222  000207                          RTS     PC              ; RETURN
```

```
1840                                    ;*****************************************************************************
1841                                    ;                                                                            ;
1842                                    ;   DROP A DRIVE                                                           ;
1843                                    ;                                                                            ;
1844                                    ;       A DRIVE WOULDN'T RESPOND, DROP IT.  SET THIS UP IN DVICE.            ;
1845                                    ;                                                                            ;
1846                                    ;       INPUT:  UNITNO = UNIT NUMBER OF DRIVE TO DROP                        ;
1847                                    ;               PORTID = BIT SET TO DROP DRIVE                               ;
1848                                    ;                                                                            ;
1849                                    ;       OUTPUT: DVICE HAS A BIT CLEARED.  THE BIT POSITION = THE DRIVE       ;
1850                                    ;                                                                            ;
1851                                    ;*****************************************************************************
1852                                    ;
1853 004224                            DROP1:
1854 004224  012767  000001  174262            MOV     #1,NUM          ;
1855 004232  000407                            BR      DROP4           ;
1856 004234                            DROP2:
1857 004234  012767  000002  174252            MOV     #2,NUM          ;
1858 004242  000403                            BR      DROP4           ;
1859 004244                            DROP3:
1860 004244  012767  000003  174242            MOV     #3,NUM          ;
1861 004252  010046                    DROP4:  MOV     R0,-(SP)        ;
1862 004254  016700  174234                    MOV     NUM,R0          ;
1863 004260  0364 '  000002  174256            BIT     2(R4),DVICE     ;DON'T DROP DEVICE TWICE
1864 004266  001401                            BEQ     10$             ;
1865 004270  046467  000002  174346            BIC     2(R4),DVICE     ;DROP THE DRIVE
1866 004276  046467  000002  174350            BIC     2(R4),ISRX      ;MAINTAIN ACCURACY
1867 004304  011467  174204                    MOV     (R4),NUM        ;MOVE IT INTO A LOCATION
1868                                            ;*************************************************************
                                                ;CONVERT NUM TO ASCII AND
                                                ;STORE AT ADR2
       004310  104421  000000' 000514'         BTOD$,BEGIN,NUM,ADR2
       004316  000537'
                                                ;*************************************************************
1869 004320  105067  174220                    CLRB    ADR2+5          ;
1870 004324  016467  000002  174162            MOV     2(R4),NUM       ;
1871 004332  042767  100000  174154            BIC     #BIT15,NUM      ;IF BOOT DEVICE
1872                                            ;*************************************************************
                                                ;CONVERT NUM TO ASCII AND
                                                ;STORE AT ADR1
       004340  104420  000000' 000514'         OTOA$,BEGIN,NUM,ADR1
       004346  000530'
                                                ;*************************************************************
1873 004350  105067  174161                    CLRB    ADR1+5          ;
1874 004354  005300                            DEC     R0              ;DROPPED FOR WHICH ERROR?
1875 004356  001004                            BNE     1$              ;IF NOT FOR ERROR5, CONTINUE
1876 004360  104403  000000' 007716'           MSGN$,BEGIN,DRP1        ;ASCII MESSAGE CALL WITH COMMON HEADER
1877 004366  000411                            BR      10$             ;
1878 004370  005300                    1$:     DEC     R0              ;WAS UNIT NOT FOUND?(NON EXISTENT UNIT)
1879 004372  001004                            BNE     2$              ;IF NOT, CONTINUE
1880 004374  104403  000000' 007734'           MSGN$,BEGIN,DRP2        ;ASCII MESSAGE CALL WITH COMMON HEADER
1881 004402  000403                            BR      10$             ;
1882 004404                            2$:
       004404  104403  000000' 007752'         MSGN$,BEGIN,DRP3        ;ASCII MESSAGE CALL WITH COMMON HEADER
1883                                            ; ACTUAL UNITS FOUND
1884 004412  012600                    10$:    MOV     (SP)+,R0        ;
1885 004414  000207                            RTS     PC              ;
```

```
1887                            ;*****************************************************************************
1888                            ;                                                                            ;
1889                            ;   WRITE                                                                    ;
1890                            ;                                                                            ;
1891                            ;       SET UP OP CODE, MODIFIERS,BUFFER SIZE (BYTE COUNT),                  ;
1892                            ;       BUFFER DESCRIPTOR (PYSICAL AND EXTENDED ADDRESS)                     ;
1893                            ;               LET READ SET SIMULAR DATA IN THE PACKET:                     ;
1894                            ;       DISK ADDRESS AND CYLINDER ID (LOGICAL BLOCK NUMBER),                 ;
1895                            ;       THEN SEND THE PACKET.                                                ;
1896                            ;                                                                            ;
1897                            ;*****************************************************************************
1898                            ;
1899 004416  004767  000714    WRITE:  CALL    CLR                     ;CLEAR PACKETS
1900 004422  012767  000042  173756    MOV     #OP.WR,P.OPCD+CMPACK    ;SET THE OPCODE
1901 004430  016700  173506    WRITEA: MOV     WBUFSZ,R0               ;STORE THE BUFFER SIZE IN WORDS
1902 004434  016767  174334  177774    MOV     W22PA,P.ADPA+CMPACK     ;SET THE BUFFER DESCRIPTOR(PA)
1903 004442  016767  174332  173750    MOV     W22EA,P.ADEA+CMPACK     ;SET THE BUFFER DESCRIPTOR(EA)
1904 004450  000423                    BR      READA                   ;
1905
1906
1907                            ;*****************************************************************************
1908                            ;                                                                            ;
1909                            ;   READ, ACCESS                                                             ;
1910                            ;                                                                            ;
1911                            ;       SET UP OP CODE, MODIFIERS,BUFFER SIZE (BYTE COUNT),                  ;
1912                            ;       BUFFER DESCRIPTOR (PYSICAL AND EXTENDED ADDRESS),                    ;
1913                            ;       DISK ADDRESS AND CYLINDER ID (LOGICAL BLOCK NUMBER),                 ;
1914                            ;       THEN SEND THE PACKET.                                                ;
1915                            ;                                                                            ;
1916                            ;*****************************************************************************
1917                            ;
1918 004452  004767  000660    ACCESS: JSR     PC,CLR                  ;CLEAR PACKETS
1919 004456  012767  000020  173722            MOV     #OP.ACC,P.OPCD+CMPACK   ;SET THE OPCODE
1920 004464  000405                            BR      ACREAD          ;COMMON ACCESS/READ CODE
1921
1922 004466  004767  000644    READ:   JSR     PC,CLR                  ;CLEAR PACKETS
1923 004472  012767  000041  173706            MOV     #OP.RD,P.OPCD+CMPACK    ;SET THE OPCODE
1924 004500  016700  174526    ACREAD: MOV     RBUFSZ,R0               ;STORE THE BUFFER SIZE IN WORDS
1925 004504  016767  174256  173706            MOV     R22EA,P.ADEA+CMPACK     ;SET THE BUFFER DESCRIPTOR
1926 004512  016767  174246  173676            MOV     R22PA,P.ADPA+CMPACK     ;
1927 004520  012767  000040  173541    READA:  MOV     #32.,RSPLEN             ;SET RESPONCE PACKET LENGTH
1928 004526  012767  000040  173636            MOV     #32.,CMPLEN             ;SET COMMAND PACKET LENGTH
1929 004534  006300                            ASL     R0                      ;MAKE IT NUMBER OF BYTES
1930 004536  010067  173650                    MOV     R0,P.BCNT+CMPACK        ;SET READ BUFFER SIZE
1931 004542  011067  173664                    MOV     (R0),P.LBN+CMPACK       ;SET LOGICAL BLOCK NUMBER
1932 004546  016067  000002  173660            MOV     2(R0),P.LBN+2+CMPACK    ;
1933 004554  000455                            BR      SEND                    ;SEND THE PACKET
```

```
1955
1956                                    ;*****************************************************************************
1957                                    ;                                                                            *
1958                                    ;   GET UNIT STATUS                                                         *
1959                                    ;                                                                            *
1960                                    ;       SET OPCODE AND MODIFIER (FOR THEN NEXT UNIT                          *
1961                                    ;               THEN SEND THE PACKET                                         *
1962                                    ;                                                                            *
1963                                    ;*****************************************************************************
1964
1965 004556  004767  000554            GTSTA:  CALL    CLR                     ;CLEAR PACKETS
1966 004562  012767  000003  173516      GTSTAA: MOV     #OP.GUS,P.OPCD+CMPACK   ;SET THE OPCODE
1967 004570  012767  000001  173512              MOV     #MD.NXU,P.MOD+CMPACK    ;CLEAR MODIFIERS
1968 004576  012767  000060  173456              MOV     #48.,RSPLEN             ;SET RESPONCE PACKET LENGTH
1969 004604  012767  000014  173560              MOV     #12.,CMPLEN             ;SET COMMAND PACKET LENGTH
1970 004612  000434                    GTSTAA: ER      SEND                    ;SEND THE PACKET
1971
1972
1973                                    ;*****************************************************************************
1974                                    ;                                                                            *
1975                                    ;   SET CONTROLLER CHARACTERISTICS                                           *
1976                                    ;                                                                            *
1977                                    ;       SET OP CODE AND CONTROLLER FLAG (ENABLE ERROR LOGS MSGS)              *
1978                                    ;       CLEAR MSCP VERSION, HOST TIMEOUT, USE FRACTION,                      *
1979                                    ;               AND ALL OF QUAD WORD TIME AND DATE.                          *
1980                                    ;       THEN SEND PACKET                                                     *
1981                                    ;                                                                            *
1982                                    ;*****************************************************************************
1983
1984 004614                            SCC.
1985 004614  004767  000516                    CALL    CLR                     ;GO CLEAR THE COMMAND PACKET
1986 004620  012767  000040  173446              MOV     #32.,CMPLEN             ;SET UP COMMAND PACKET LENGTH
1987 004626  012767  000034  173436              MOV     #28.,RSPLEN             ;SET UP RESPONCE PACKET LENGTH
1988 004634  012767  000004  173544              MOV     #OP.SCC,P.OPCD+CMPACK   ;SET THE OPCODE
1989 004642  012767  000120  173544              MOV     #120,P.CNTF+CMPACK      ;SET CONTROLLER FLAGS = ERROR LOGS
1990 004650  000415                              BR      SEND                    ;SEND THE PACKET
1991
1992                                    ;*****************************************************************************
1993                                    ;                                                                            *
1994                                    ;   ONLINE                                                                   *
1995                                    ;                                                                            *
1996                                    ;       SET OPCODE, MODIFIERS, UNIT ID, HOST ID                              *
1997                                    ;       SHADOW UNIT, ERROR FLAGS                                             *
1998                                    ;       THEN SEND PACKET                                                     *
1999                                    ;                                                                            *
2000                                    ;*****************************************************************************
2001
2002 004652  004767  000460            ONLINE: CALL    CLR                     ;CLEAR PACKETS
2003 004656  012767  000054  173406              MOV     #44.,RSPLEN             ;SET RESPONCE PACKET LENGTH
2004 004664  012767  000044  173500              MOV     #36.,CMPLEN             ;SET COMMAND PACKET LENGTH
2005 004672  012767  000011  173506              MOV     #OP.ONL,P.OPCD+CMPACK   ;SET THE OPCODE
2006 004700  011467  173532                      MOV     (R4),P.SHUN+CMPACK      ;SHADOW UNIT  (UNITNO.
2007                                                                            ;SEND THE PACKET
```

```
                        ;*****************************************************************************
                        ;                                                                            ;
                        ;   SEND - SEND A PACKET                                                     ;
                        ;       INTERP - WAIT FOR AN INTERRUPT                                       ;
                        ;                                                                            ;
                        ;       SET UP THE COMMAND REFERENCE NUMBER AND UNITNO IN THE PACKET         ;
                        ;       SET OWN, CLEAR FLAG IN THE COMMAND RING (FOR RQDX)                   ;
                        ;       SET OWN & FLAG IN MESSAGE RING (FOR INTERRUPTS BY RQDX)              ;
                        ;       AFTER INTERRUPT, MAKE SURE THE PACKET WAS PROCESSED (NO HARD         ;
                        ;       OR SOFT ERRORS) THEN RETURN TO CYCLED.                               ;
                        ;                                                                            ;
                        ;       INPUT:  CMPACK IS FILLED EXCEPT FOR CMDREF & UNITNO                  ;
                        ;               INTERRUPT VECTOR AND BR LEVEL ARE ESTABLISHED                ;
                        ;                                                                            ;
                        ;       OUTPUT: MSPACK IS FILLED                                             ;
                        ;               CLEAR CARRY IF COMMAND PACKET WAS OK                         ;
                        ;                       ELSE GO DO A HARD/SOFT ERROR                         ;
                        ;                                                                            ;
                        ;*****************************************************************************
                        ;
004704                  SEND:
004704  010046                  MOV     R0,-(R6)
004706  005067  173714            CLR     SFTFLG                  ; CLEAR SOFT ERROR FLAG
004712  005267  173554    1$:     INC     CMDREF                  ; NEW COMMAND REFERENCE NUMBER
004716  001775                  BEQ     1$                      ; COMMAND REF # CANNOT - 0
004720  016767  173546  173450        MOV     CMDREF,P.CRF+CMPACK     ; SET COMMAND REF NUMBER
004726  011467  173450            MOV     (R4),P.UNIT+CMPACK      ; SET UNIT NUMBER
004732  042767  040000  173326        BIC     #RG.FLG,COMMND+2        ; CLEAR FLAG
004740  052767  100000  173320        BIS     #RG.OWN,COMMND+2        ; SET OWN  FOR COMMAND RING
004746  052767  140000  173306        BIS     #RG.FLG!RG.OWN,RSPONC+2 ; SET OWN AND FLAG FOR MESSAGE RING
004754  004767  002470            JSR     PC,OUTPKT               ; PRINT PACKET (DIAGNOSTICS)
004760  005777  173022            TST     @ADDR                   ; FORCE POLLING TO PACKET
004764                  WAIT:                                   ; BGNDO
004764  104400  000000'            EXIT$,BEGIN                     ;EXIT TO MONITOR. MODULE WAIT FOR INTERRUPT.
004770                  NTRUPT:                                 ; :
                        ;       ------------------------------------------------------------------
004770  000004  000000' 004776'        PIRQ$,BEGIN,1$          ; QUEUE UP TO CONTINUE AT 1$ AND RT1
                        ;       ------------------------------------------------------------------
004776                  1$:                                     ; :
004776  005067  173254            CLR     RINTR                   ; :  CLEAR INTERRUPT FLAG
005002  004767  002460            JSR     PC,INPKT                ; :  PRINT PACKET (DIAGNOSTICS)
005006                  2$:                                     ; :
005006  016700  173262            MOV     RSPVTR,R0               ; :  GET RESPONSE
005012  042700  177417            BIC     #177417,R0              ; :  CLEAR ANY CREDITS
005016  022700  000020            CMP     #20,R0                  ; :  IF RESPONSE = ERROR LOG
005022  001006                  BNE     10$                     ; :
005024  005267  173576            INC     SFTFLG                  ; :      INCREMNET SOFT ERROR FLAG
005030  052767  140000  173224        BIS     #RG.FLG!RG.OWN,RSPONC+2 ; :      SET OWN AND FLAG FOR MESSAGE RING
005036  000752                  BR      WAIT                    ; DOUNTIL
005040                  10$:                                    ;
005040  004767  000004            CALL    ERRCHK                  ; CALL ERROR CHECK
005044                  99$:
005044  012600                  MOV     (R6)+,R0
005046  000207                  RTS     PC                      ; RETURN
```

```
                                        ;*****************************************************************************
                                        ;                                                                            *
                                        ;    ERROR CHECK                                                             *
                                        ;                                                                            *
                                        ;       CHECK FOR HARD OR SOFT ERRORS THEN RETURN                            *
                                        ;                                                                            *
                                        ;       INPUT:  R0 = STATUS OF CONTROLLER                                    *
                                        ;                                                                            *
                                        ;       OUTPUT: CLEAR CARRY IF COMMAND PACKET WAS OK                         *
                                        ;                       ELSE GO DO A HARD/SOFT ERROR                         *
                                        ;                                                                            *
                                        ;*****************************************************************************
                                        ;
005050                                  ERRCHK:
005050  005067  173032                          CLR     ERRTYP                  ; CLEAR ERROR_TYPE
005054  016700  173230                          MOV     P.STS+RSPACK,R0         ; GET SUCCESS
005060  042700  177740                          BIC     #177740,R0              ; CLEAR UPPER 11 BITS OF SUB-STATUS
005064  001513                                  BEQ     15$                     ; IF SUCCESS NOT = 0
                                                                                ; : THEN
                                                                                ; :   SELECT STATUS
005066  122700  000013                          CMPB    #ST.DRV,R0              ; :   :  CASE DRIVE ERROR:
005072  001004                                  BNE     3$                      ; :   :
005074  012767  000006  173004                  MOV     #ERR.6,ERRTYP           ; :   :    ERROR_TYPE = DRIVE NOT READY, OFF LINE OR
NON EXESTENT
005102  000474                                  BR      12$                     ; :   :
005104                                  3$:                                     ; :   :
005104  122700  000012                          CMPB    #ST.CNT,R0              ; :   :  CASE CONTROLLER ERROR
005110  001004                                  BNE     4$                      ; :   :
005112  012767  000003  172766                  MOV     #ERR.3,ERRTYP           ; :   :    ERROR_TYPE = CONTROLLER NOT READY
005120  000465                                  BR      12$                     ; :   :
005122                                  4$:                                     ; :   :
005122  122700  000011                          CMPB    #ST.HST,R0              ; :   :  CASE HOST BUFFER ACCESS ERROR:
005126  001004                                  BNE     5$                      ; :   :
005130  012767  000032  172750                  MOV     #ERR.32,ERRTYP          ; :   :    ERROR_TYPE = NPR ERROR
005136  000456                                  BR      12$                     ; :   :
005140                                  5$:                                     ; :   :
005140  122700  000010                          CMPB    #ST.DAT,R0              ; :   :  CASE DATA ERROR:
005144  001004                                  BNE     6$                      ; :   :
005146  012767  000001  172732                  MOV     #ERR.1,ERRTYP           ; :   :    ERROR_TYPE = DATA ERROR
005154  000447                                  BR      12$                     ; :   :
005156                                  6$:                                     ; :   :
005156  122700  000006                          CMPB    #ST.WPR,R0              ; :   :  CASE WRITE PROTECTED:
005162  001004                                  BNE     8$                      ; :   :
005164  012767  000006  172714                  MOV     #ERR.6,ERRTYP           ; :   :    ERROR_TYPE = DRIVE NOT READY, OFF LINE OR
NON EXESTENT
005172  000440                                  BR      12$                     ; :   :
005174                                  8$:                                     ; :   :
005174  122700  000004                          CMPB    #ST.AVL,R0              ; :   :  CASE STILL AVAILABLE:
005200  001010                                  BNE     9$                      ; :   :
005202  022767  000003  173176                  CMP     #OP.GUS,P.OPCD+CMPACK   ; :   :    IF COMMAND NOT   GET UNIT STATUS
005210  001437                                  BEQ     14$                     ; :   :
005212  012767  000006  172666                  MOV     #ERR.6,ERRTYP           ; :   :     ERROR TYPE = DRIVE NOT READY, OFF LINE O
R NON EXESTENT
005220  000425                                  BR      12$                     ; :   :
005222                                  9$:                                     ; :   :
005222  122700  000003                          CMPB    #ST.OFL,R0              ; :   :  CASE UNIT OFFLINE:
005226  001025                                  BNE     13$                     ; :   :
005230  022767  000011  173150                  CMP     #OP.ONL,P.OPCD+CMPACK   ; :   :     IF AN ONLINE COMMAND
005236  001410                                  BEQ     10$                     ; :   :     OR
005240  022767  000003  173140                  CMP     #OP.GUS,P.OPCD+CMPACK   ; :   :     IF GET UNIT STATUS COMMAND
```

```
   2098 005246  001404                        BEQ     10$                     ; :   :     OR
   2099 005250  022767  000042  173130        CMP     #OP.WR,P.OPCD+CMPACK    ; :   :     IF A WRITE COMMAND
   2100 005256  001002                        BNE     11$                     ; :   :     : THEN
   2101 005260  000261                10$:    SEC                             ; :   :     :   SET CARRY TO DROP DRIVE/AWAIT AVAILA
BLE DRIVE
   2102 005262  000424                        BR      99$                     ; :   :     :   EXIT
   2103 005264                        11$:                                    ; :   :     : ELSE
   2104 005264  012767  000006  172614        MOV     #ERR.6,ERRTYP           ; :   :     :   ERROR_TYPE =  DRIVE NOT READY, OFF L
INE OR NON EXESTEN
T
   2105 005272  000400                        BR      12$                     ; :   :     ENDIF
   2106                                                                       ; :   ENDSELECT
   2107 005274                        12$:                                    ; :   *** HARD ERROR EXIT ->
   2108 005274  004767  000136                CALL    ERRORH                  ; :    AND HARD ERROR
   2109 005300  000415                        BR      99$                     ; :
   2110                                                                       ; :
   2111 005302                        13$:                                    ; :   *** SOFT ERROR EXIT -> ERROR TYPE = 0
   2112 005302  004767  000170                CALL    ERRORS                  ; :    ERROR WITH ERRTYP = 0 & IS A SOFT ERROR
   2113                                                                       ; :    ST.CMP,ST.MFE,.ST.ABO,ST.CMD
   2114 005306  000412                        BR      99$                     ; :
   2115                                                                       ; :
   2116 005310                        14$:                                    ; :   *** SUCESSFUL EXIT
   2117 005310  000241                        CLC                             ; :    CLEAR CARRY 'CAUSE PACKET IS OK
   2118 005312  000410                        BR      99$                     ; :
   2119 005314                        15$:                                    ; : ELSE
   2120 005314  005767  173306                TST     SFTFLG                  ; :  IF SOFT_ERROR_FLAG NOT = 0
   2121 005320  001405                        BEQ     99$                     ; :  : THEN
   2122 005322  005067  173300                CLR     SFTFLG                  ; :  :  CLEAR SOFT ERROR FLAG
   2123 005326  004767  000144                CALL    ERRORS                  ; :  :  CALL ERROR_SOFT
   2124 005332  000241                        CLC                             ; :  :  CLEAR CARRY - DON'T DROP DRIVE
   2125                                                                       ; :  ENDIF
   2126                                                                       ; ENDIF
   2127 005334  000207                99$:    RETURN                          ;
   2128
   2129
```

```
2131                                    ;*****************************************************************************
2132                                    ;                                                                            ;
2133                                    ;   CLEAR                                                                    ;
2134                                    ;       IF SA = ERROR REPORT & REINIT                                        ;
2135                                    ;       CLEAR PACKETS                                                        ;
2136                                    ;                                                                            ;
2137                                    ;*****************************************************************************
2138                                    ;
2139 005336                            CLR:
2140 005336  010246                            MOV     R2,-(SP)        ;
2141 005340  017767  172706  173146            MOV     @RQDXSA,NUM     ; IS RQDXSA NOT ZERO?STORE IN NUM IF TRUE
2142                                                                    ; IF RQDXSA IS NOT ZERO
2143 005346  001417                            BEQ     5$              ; : THEN
2144                                            ;***************************************************************
                                                ;CONVERT NUM TO ASCII AND
                                                ;STORE AT ADR1
     005350  104420  000000' 000514'            OTOA$,BEGIN,NUM,ADR1
     005356  000530'
                                                ;***************************************************************
2145 005360  104403  000000' 010004'            MSGN$,BEGIN,SANOTO      ;ASCII MESSAGE CALL WITH COMMON HEADER
2146 005366  010346                            MOV     R3,-(SP)        ; : SAVE R3
2147 005370  010446                            MOV     R4,-(SP)        ; : SAVE R4
2148 005372  004767  174426                    JSR     PC,INIPRT       ; : RE INIT RQDXSA
2149 005376  012604                            MOV     (SP)+,R4        ; : RESTORE R3
2150 005400  012603                            MOV     (SP)+,R3        ; : RESTORE R4
2151                                    ;       JSR     PC,SCC          ; : SET CONTROLLER CHARS AGAIN
2152 005402  005267  172436                    INC     HRDCNT          ; : INCREMENT HARD ERROR COUNT
2153                                                                    ; : THIS WILL CAUSE ANOTHER CALL TO CLRPAK
2154 005406                            5$:                             ; ENDIF
2155 005406  004767  000004                    CALL    CLRPAK          ; CALL CLEAR_PACKETS
2156 005412  012602                            MOV     (SP)+,R2        ;POP
2157 005414  000207                            RTS     PC              ;RETURN
2158
2159                                    ;*****************************************************************************
2160                                    ;                                                                            ;
2161                                    ;   CLEAR PACKETS                                                            ;
2162                                    ;                                                                            ;
2163                                    ;       ASSUMPTION:  1)  RESPONCE BUFFER PRECEDES THE COMMAND BUFFER         ;
2164                                    ;                    2)  TWO WORDS BEFORE EACH BUFFER IS FOR LENGTH          ;
2165                                    ;                            OF PACKET AND VIRTUAL CIRCUIT                   ;
2166                                    ;       OUTPUT:                                                              ;
2167                                    ;               R5 = END OF COMMAND PACKET WHEN DONE                         ;
2168                                    ;                                                                            ;
2169                                    ;*****************************************************************************
2170                                    ;
2171 005416                            CLRPAK:
2172 005416  012702  000100                    MOV     #64.,R2         ; R2 = # OF WORDS TO CLEAR
2173 005422  012705  000272'                    MOV     #RSPLEN,R5      ; R5 -> RSPLEN, 1ST WORD TO CLEAR
2174 005426                            6$:                             ; BGNDO
2175 005426  005025                            CLR     (R5)+           ; : CLEAR WORD
2176 005430  005302                            DEC     R2              ; : DECREMENT COUNTER
2177 005432  001375                            BNE     6$              ; DOUNTIL = ZERO
2178 005434  000207                            RETURN
```

```
                                     ;**********************************************************************
                                     ;                                                                     ;
                                     ;   HARD ERROR          CARRY WILL BE SET                             ;
                                     ;                                                                     ;
                                     ;**********************************************************************

005436                               ERRORH:
005436  032767  000004  172352              BIT     #SR.REP,SR1        ; IF REPORT ERROR = NO
005444  001403                              BEQ     7$                 ; : THEN
005446  005267  172372                      INC     HRDCNT             ; :  INCREMENT THE HARD ERROR
005452  000407                              BR      8$                 ; :
005454                               7$:                               ; : ELSE
005454  004767  000056                      JSR     PC,SETTAB          ; :  SET UP TABLE
                                            ;****************************************************************
005460  104405  000000  000000              HRDER$,BEGIN,NULL          ;
                                            ;****************************************************************
005466  004767  000070                      JSR     PC,PRINTE          ; :
005472                               8$:                               ; ENDIF
005472  000261                              SEC                        ;
005474  000207                              RTS     PC                 ; RETURN TO CYCLED

                                     ;**********************************************************************
                                     ;                                                                     ;
                                     ;   SOFT ERROR          CARRY WILL BE SET                             ;
                                     ;                                                                     ;
                                     ;**********************************************************************
                                     ;
005476                               ERRORS:
005476  032767  000004  172312              BIT     #SR.REP,SR1        ; IF REPORT ERROR = NO
005504  001405                              BEQ     9$                 ; : THEN
005506  005267  172330                      INC     SOFCNT             ; :  INCREMENT THE SOFT ERROR
005512  000407                              BR      10$                ; :
005514                               9$:                               ; : ELSE
005514  004767  000016                      CALL    SETTAB             ; :   SET UP TABLE
                                            ;****************************************************************
005520  104406  000000  000000              SOFER$,BEGIN,NULL          ;
                                            ;****************************************************************
005526  004767  000030                      CALL    PRINTE             ; :
005532                               10$:                              ; ENDIF
005532  000261                              SEC                        ; SET CARRY
005534  000207                              RTS     PC                 ; RETURN TO CYCLED

                                     ;**********************************************************************
                                     ;                                                                     ;
                                     ;   SETTAB                                                            ;
                                     ;                                                                     ;
                                     ;        SET UP A TABLE OF VALUES FOR A SOFT OR HARD ERROR            ;
                                     ;                                                                     ;
                                     ;**********************************************************************
                                     ;
005536                               SETTAB:
005536  016767  172510  172334              MOV     RQDXSA,CSRA                ;SET UP CONTROL STATUS REG REPORT
005544  016767  172540  172330              MOV     P.STS+RSPAC+,ASTAT         ;SET UP STATUS
005552  017767  172474  172322              MOV     @RQDXSA,ACSR               ;REPORT WHAT IS STATUS REG
005560  000207                              RTS     PC                         ; RETURN
```

```
                                ;*****************************************************************
                                ;
                                ;   PRINT EXTENDED ERROR   (STATUS, OPCODE, UNIT NUMBER, BYTE COUNT,
                                ;                           LBN AND ADDRESS)
                                ;
                                ;*****************************************************************
005560                          PRINTE:
                                        ;*************************************************************
                                                                ;CONVERT P.STS+RSPACK TO ASCII AND
                                                                ;STORE AT ADR1
005560  1044,20  000000' 000310'        OTOA$,BEGIN,P.STS+RSPACK,ADR1
005570  000530'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.OPCD+RSPACK TO ASCII AND
                                                                ;STORE AT ADR2
005574  1044,20  000000' 000306'        OTOA$,BEGIN,P.OPCD+RSPACK,ADR2
005600  000537'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.UNIT+RSPACK TO ASCII AND
                                                                ;STORE AT ADR3
005602  1044,20  000000' 000302'        OTOA$,BEGIN,P.UNIT+RSPACK,ADR3
005610  000546'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.BCNT+RSPACK TO ASCII AND
                                                                ;STORE AT ADR4
005612  1044,20  000000' 000312'        OTOA$,BEGIN,P.BCNT+RSPACK,ADR4
005620  000555'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.LBN+2+CMPACK TO ASCII AND
                                                                ;STORE AT ADR5
005622  1044,20  000000' 000434'        OTOA$,BEGIN,P.LBN+2+CMPACK,ADR5
005630  000564'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.LBN+CMPACK TO ASCII AND
                                                                ;STORE AT ADR6
005632  1044,20  000000' 000432'        OTOA$,BEGIN,P.LBN+CMPACK,ADR6
005640  000573'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.ADEA+CMPACK TO ASCII AND
                                                                ;STORE AT ADR7
005642  1044,20  000000' 000420'        OTOA$,BEGIN,P.ADEA+CMPACK,ADR7
005650  000602'
                                        ;*************************************************************
                                        ;*************************************************************
                                                                ;CONVERT P.ADPA+CMPACK TO ASCII AND
                                                                ;STORE AT ADR8
005652  1044,20  000000' 000416'        OTOA$,BEGIN,P.ADPA+CMPACK,ADR8
005660  000611
                                        ;*************************************************************
005662  104403  000000' 010226'        MSGN$,BEGIN,BANER1      ;ASCII MESSAGE CALL WITH COMMON HEADER
005670  000207                          RTS     PC                      ;RETURN
```

```
                                ;*****************************************************************************
                                ;
                                ;   DO DUP OPERATIONS
                                ;
                                ;
                                ;*****************************************************************************
                                ;
                                DODUP:
                                1$:                                             ; BGNDO
                                                                                ; :  SETUP
005672  004767  000462          CALL    GTDUST                                  ; :  CALL GET DUST STATUS
                                                                                ; :  ERROR CHECK
005676  132767  000010  172373  BITB    #SE.ACT,RSPONS+S.FLAG                   ; DOUNTIL NOT ACTIVE OR ERROR
005704  001401                  BEQ     2$                                      ;
005706  103371                  BCC     1$                                      ;
005710  103423                  BCS     99$                                     ; IF NO ERROR
                                                                                ; : THEN
005712  104417  000000'          RAND$,BEGIN
005716  016767  172152  172700  MOV     RANNUM,DBN                              ; :
005724  042767  177560  172672  BIC     #177560,DBN                             ; :  SELECT A BLOCK
005732  004767  000240          CALL    REDDBN                                  ; :  CALL READ DBN
005736  10 410                  BCS     99$                                     ; :  IF NO ERROR
                                                                                ; :  : THEN
005740  004767  000016          CALL    WRTDBN                                  ; :  :  CALL WRITE DBN
005744  10 405                  BCS     99$                                     ; :  :  IF NO ERROR
                                                                                ; :  :  : THEN
005746  004767  000224          CALL    REDDBN                                  ; :  :  :  CALL READ DBN
                                                                                ; :  :  ENDIF
005752  000402                  BR      99$                                     ; :  ENDIF
005754                          10$:                                            ; : ELSE
005754  004767  001220          CALL    DUPERR                                  ; :  CALL DUP ERROR
                                                                                ; ENDIF
005760  000207                  99$:    RETURN                                  ; RETURN
```

```
2286                                    ;*****************************************************************************
2287                                    ;
2288                                    ;   DO WRITE DBN
2289                                    ;
2290                                    ;*****************************************************************************
2291                                    ;
2292 005760                            WRTDBN:
2293 005762  012700  006476'                   MOV     #WDBN,R0                ; SETUP = WRITE DBN
2294 005766  004767  000424                    CALL    EXECPG                  ; CALL EXCUTE LOACAL PROGRAM
2295 005772  103500                            BCS     99$                     ; IF ERROR
2296                                                                            ; : THEN
2297                                                                            ; :  EXIT BLOCK
2298                                                                            ; ENDIF
2299 005774  012767  000002  172472           MOV     #2,BYTCNT               ; SET BYTE COUNT = 2
2300 006002  012767  012462'  172466          MOV     #RBUF,DATLOC            ; SET DATA LOCATION
2301 006010  004767  000556                    CALL    RCVDAT                  ; CALL RECIEVE DATA
2302 006014  103467                            BCS     99$                     ; IF ERROR
2303                                                                            ; : THEN -> EXIT BLOCK
2304                                                                            ; ENDIF
2305 006016  022767  010006  004436           CMP     #BIT12+6,RBUF           ; IF RESPONSE NOT = OK
2306 006024  001406                            BEQ     2$                      ; : THEN
2307 006026  012767  000006  172566           MOV     #6,ERRFLG               ; :   SET ERROR
2308 006034  004767  001140                    CALL    DUPERR                  ; :   CALL DUP ERROR
2309 006040  000455                            BR      99$                     ; :   EXIT BLOCK
2310 006042                            2$:                                     ; ENDIF
2311 006042  011467  004414                    MOV     (R4),RBUF               ; SET UNIT NO
2312 006046  016767  172552  004410           MOV     DBN,RBUF+2              ; SET DBN
2313 006054  012767  125125  004404           MOV     #125125,RBUF+4          ; SET PATTERN
2314 006062  012767  000006  172404           MOV     #6,BYTCNT               ; SET BYTE COUNT = 6
2315 006070  012767  012462'  172400          MOV     #RBUF,DATLOC            ; SET DATA LOCATION
2316 006076  004767  000410                    CALL    SNDDAT                  ; CALL SEND DATA
2317 006102  103434                            BCS     99$                     ; IF ERROR
2318                                                                            ; : THEN -> EXIT BLOCK
2319                                                                            ; ENDIF
2320 006104  012767  000004  172362           MOV     #4,BYTCNT               ; SET BYTE COUNT = 4
2321 006112  012767  012462'  172356          MOV     #RBUF,DATLOC            ; SET DATA LOCATION
2322 006120  004767  000446                    CALL    RCVDAT                  ; CALL RECIEVE DATA
2323 006124  103423                            BCS     99$                     ; IF ERROR
2324                                                                            ; : THEN -> EXIT BLOCK
2325                                                                            ; ENDIF
2326 006126  022767  030003  004326           CMP     #BIT13+BIT12+3,RBUF     ; IF RESPONSE NOT = OK
2327 006134  001406                            BEQ     6$                      ; : THEN
2328 006136  012767  000006  172456           MOV     #6,ERRFLG               ; :   SET ERROR
2329 006144  004767  001030                    CALL    DUPERR                  ; :   CALL DUP ERROR
2330 006150  000411                            BR      99$                     ; :   EXIT BLOCK
2331 006152                            6$:                                     ; : ELSE
2332 006152  005767  004306                    TST     RBUF+2                  ; :   IF ERROR SET
2333 006156  001406                            BEQ     7$                      ; :   : THEN
2334 006160  012767  000006  172434           MOV     #6,ERRFLG               ; :   :   SET ERROR
2335 006166  004767  001006                    CALL    DUPERR                  ; :   :   CALL DUP ERROR
2336 006172  000400                            BR      99$                     ; :   :   EXIT BLOCK
2337                                                                            ; :   ENDIF
2338 006174                            7$:                                     ; ENDIF
2339 006174  000207                    99$:    RETURN                          ;
2340
```

```
2342                                    ;*****************************************************************************
2343                                    ;
2344                                    ;   DO READ DBN
2345                                    ;
2346                                    ;
2347                                    ;*****************************************************************************
2348                                    ;
2349 006176                            REDDBN:
2350 006176  012700  006504'                   MOV     #RDBN,R0                ; SETUP = READ_DBN
2351 006202  004767  000210                    CALL    EXLCPG                  ; CALL EXCUTE_LOACAL_PROGRAM
2352 006206  103463                            BCS     99$                     ; IF ERROR
2353                                                                           ; : THEN -> EXIT BLOCK
2354                                                                           ; ENDIF
2355 006210  012767  000002  172256            MOV     #2,BYTCNT               ; SET BYTE_COUNT = 2
2356 006216  012767  012462' 172252            MOV     #RBUF,DATLOC            ; SET DATA_LOCATION
2357 006224  004767  000342                    CALL    RCVDAT                  ; CALL RECIEVE_DATA
2358 006230  103452                            BCS     99$                     ; IF ERROR
2359                                                                           ; : THEN -> EXIT BLOCK
2360                                                                           ; ENDIF
2361 006232  022767  010005  004222            CMP     #BIT12+5,RBUF           ; IF RESPONSE NOT = OK
2362 006240  001406                            BEQ     2$                      ; : THEN
2363 006242  012767  000006  172352            MOV     #6,ERRFLG               ; :   SET ERROR
2364 006250  004767  000724                    CALL    DUPERR                  ; :   CALL DUP_ERROR
2365 006254  000440                            BR      99$                     ; :   EXIT BLOCK
2366 006256                            2$:                                     ; ENDIF
2367 006256  011467  004200                    MOV     (R4),RBUF               ; SET UNIT NO
2368 006262  016767  172336  004174            MOV     DBN,RBUF+2              ; SET DBN
2369 006270  012767  000004  172176            MOV     #4,BYTCNT               ; SET BYTE COUNT = 4
2370 006276  012767  012462' 172172            MOV     #RBUF,DATLOC            ; SET DATA_LOCATION
2371 006304  004767  000202                    CALL    SNDDAT                  ; CALL SEND_DATA
2372 006310  103422                            BCS     99$                     ; IF ERROR
2373                                                                           ; : THEN -> EXIT BLOCK
2374                                                                           ; ENDIF
2375 006312  012767  001002  172154            MOV     #514.,BYTCNT            ; SET BYTE_COUNT = 514.
2376 006320  012767  012462' 172150            MOV     #RBUF,DATLOC            ; SET DATA LOCATION
2377 006326  004767  000240                    CALL    RCVDAT                  ; CALL RECIEVE DATA
2378 006332  103411                            BCS     99$                     ; IF ERROR
2379                                                                           ; : THEN -> EXIT BLOCK
2380                                                                           ; ENDIF
2381 006334  022767  060002  004120            CMP     #BIT14+BIT13+2,RBUF     ; IF R_BUF NOT = VALID
2382 006342  001405                            BEQ     10$                     ; : THEN
2383 006344  012767  000006  172250            MOV     #6,ERRFLG               ; :   SET ERROR
2384 006352  004767  000622                    CALL    DUPERR                  ; :   CALL DUP_ERROR
2385 006356                            10$:                                    ; ENDIF
2386 006356  000207                    99$:    RETURN                          ;
2387
```

```
                                        ;*****************************************************************************
                                        ;
                                        ;   GET DUST STATUS
                                        ;
                                        ;
                                        ;*****************************************************************************
                                        ;
006360                                  GTDUST:
006360  004767  177032                          JSR     PC,CLRPAK               ; GO CLEAR THE COMMAND PACKET
006364  012767  000014  172000                  MOV     #12.,CMPLEN             ; SET UP COMMAND PACKET LENGTH
006372  012767  000026  171672                  MOV     #22.,RSPLEN             ; SET UP RESPONCE PACKET LENGTH
006400  012767  000001  172000                  MOV     #OP.GDS,P.OPCD+CMPACK   ; SET THE OPCODE
006406  012767  000201  172064                  MOV     #OP.GDS!OP.END,ENDCOD   ; SET END CODE FOR CHECK
006414  000516                                  BR      SNDDUP                  ; SEND THE PACKET
                                        ;*****************************************************************************
                                        ;
                                        ;   EXECUTE LOCAL PROGRAM
                                        ;
                                        ;       INPUT:
                                        ;               R0 = POINTER TO ASCII STRING NAME OF PROGRAM TO EXECUTE.
                                        ;
                                        ;*****************************************************************************
                                        ;
006416                                  EXLCPG:
006416  004767  176774                          JSR     PC,CLRPAK               ; GO CLEAR THE COMMAND PACKET
006422  012767  000022  171742                  MOV     #18.,CMPLEN             ; SET UP COMMAND PACKET LENGTH
006430  012767  000020  171634                  MOV     #16.,RSPLEN             ; SET UP RESPONCE PACKET LENGTH
006436  012767  000003  171742                  MOV     #OP.ELP,P.OPCD+CMPACK   ; SET THE OPCODE
006444  012767  000001  171736                  MOV     #1,CMPACK+12            ; SET MODIFIERS
006452  012067  171734                          MOV     (R0)+,CMPACK+14         ; SET ADDRESS OF ASCII NAME PRG TO RUN
006456  012067  171732                          MOV     (R0)+,CMPACK+16         ; SET ADDRESS OF ASCII NAME PRG TO RUN
006462  011067  171730                          MOV     (R0),CMPACK+20          ; SET ADDRESS OF ASCII NAME PRG TO RUN
006466  012767  000203  172004                  MOV     #OP.ELP!OP.END,ENDCOD   ; SET END CODE FOR CHECK
006474  000466                                  BR      SNDDUP                  ; SEND THE PACKET

006476  127  122  124  WDBN:   .ASCII  /WRTDBN/
006501  104  102  116
006504  122  105  104  RDBN:   .ASCII  /REDDBN/
006507  104  102  116
```

```
2431                                    ;*****************************************************************************
2432                                    ;
2433                                    ;   SEND DATA
2434                                    ;
2435                                    ;       INPUT:
2436                                    ;               OPCODE =
2437                                    ;               BYTE_COUNT =
2438                                    ;               LOCATION =
2439                                    ;
2440                                    ;
2441                                    ;*****************************************************************************
2442                                    ;
2443 006512                            SNDDAT:
2444 006512  004767  176700            JSR     PC,CLRPAK               ; GO CLEAR THE COMMAND PACKET
2445 006516  016767  171750  171652    MOV     CMDREF,P.CRF+CMPACK     ; SET COMMAND REFERENCE #
2446 006524  012767  000034  171640    MOV     #.C.SND,CMPLEN          ; SET UP COMMAND PACKET LENGTH
2447 006532  012767  000020  171532    MOV     #.R.SND,RSPLEN          ; SET UP RESPONCE PACKET LENGTH ??????
2448 006540  012767  000004  171640    MOV     #OP.SND,P.OPCD+CMPACK   ; SET THE OPCODE
2449 006546  016767  171722  171636    MOV     BYTCNT,CMPACK+14        ; SET BYTE COUNT
2450 006554  016767  171716  171634    MOV     DATLOC,CMPACK+20        ; SET DATA LOCATION
2451 006562  012767  000204  171710    MOV     #OP.SND!OP.END,ENDCOD   ; SET END CODE FOR CHECK
2452                                    ;
2453 006570  000430                    BR      SNDDUP                  ; SEND THE PACKET
2454
2455                                    ;*****************************************************************************
2456                                    ;
2457                                    ;   RECEIVE DATA
2458                                    ;
2459                                    ;       INPUT:
2460                                    ;               OPCODE =
2461                                    ;               BYTE_COUNT =
2462                                    ;               LOCATION =
2463                                    ;
2464                                    ;
2465                                    ;*****************************************************************************
2466                                    ;
2467 006572                            RCVDAT:
2468 006572  004767  176620            JSR     PC,CLRPAK               ; GO CLEAR THE COMMAND PACKET
2469 006576  016767  171670  171572    MOV     CMDREF,P.CRF+CMPACK     ; SET COMMAND REFERENCE #
2470 006604  012767  000034  171560    MOV     #.C.RCD,CMPLEN          ; SET UP COMMAND PACKET LENGTH
2471 006612  012767  000020  171452    MOV     #.R.RCD,RSPLEN          ; SET UP RESPONCE PACKET LENGTH ????
2472 006620  012767  000005  171560    MOV     #OP.RCD,P.OPCD+CMPACK   ; SET THE OPCODE
2473 006626  016767  171642  171556    MOV     BYTCNT,CMPACK+14        ; SET BYTE COUNT
2474 006634  016767  171636  171554    MOV     DATLOC,CMPACK+20        ; SET DATA LOCATION
2475 006642  012767  000205  171630    MOV     #OP.RCD!OP.END,ENDCOD   ; SET END CODE FOR CHECK
2476                                    ;
2477 006650  000400                    BR      SNDDUP                  ; SEND THE PACKET
2478
2479
2480
```

```
2482                            ;*****************************************************************************
2483                            ;                                                                            ;
2484                            ;   SNDDUP - SEND A DUP PACKET                                               ;
2485                            ;                                                                            ;
2486                            ;       SET UP THE COMMAND REFERENCE NUMBER AND UNITNO IN THE PACKET          ;
2487                            ;       SET OWN, CLEAR FLAG IN THE COMMAND RING (FOR RQDX)                   ;
2488                            ;       SET OWN & FLAG IN MESSAGE RING                                       ;
2489                            ;       AFTER DONE, MAKE SURE THE PACKET WAS PROCESSED (NO HARD               ;
2490                            ;       OR SOFT ERRORS) THEN RETURN.                                         ;
2491                            ;                                                                            ;
2492                            ;       INPUT:  CMPACK IS FILLED EXCEPT FOR CMDREF & UNITNO                  ;
2493                            ;               INTERRUPT VECTOR AND BR LEVEL ARE ESTABLISHED                ;
2494                            ;                                                                            ;
2495                            ;       OUTPUT: MSPACK IS FILLED                                             ;
2496                            ;               CLEAR CARRY IF COMMAND PACKET WAS OK                         ;
2497                            ;                       ELSE GO DO A HARD/SOFT ERROR                         ;
2498                            ;                                                                            ;
2499                            ;*****************************************************************************
2500                            ;
2501 006652  005267  171614    SNDDUP: INC     CMDREF                  ; NEW COMMAND REFERENCE NUMBER
2502 006656  001775                    BEQ     SNDDUP                  ; COMMAND REF # CANNOT = 0
2503 006660  016767  171606  171510        MOV     CMDREF,P.CRF+CMPACK     ; SET COMMAND REF NUMBER
2504 006666  011467  171510            MOV     (R4),P.UNIT+CMPACK      ; SET UNIT NUMBER
2505 006672  042767  040000  171366        BIC     #RG.FLG,COMMND+2        ; CLEAR FLAG
2506 006700  052767  100000  171360        BIS     #RG.OWN,COMMND+2        ; SET OWN FOR COMMAND RING
2507 006706  042767  040000  171346        BIC     #RG.FLG,RSPONC+2        ; CLEAR FLAG
2508 006714  052767  100000  171340        BIS     #RG.OWN,RSPONC+2        ; SET OWN FOR RESPONSE RING
2509 006722  012767  001001  171444        MOV     #1001,CMPVIR            ; SET DUP COMMMAND
2510 006730  004767  000514            JSR     PC,OUTPKT               ;PRINT PACKET (DIAGNOSTICS)
2511 006734  005777  171046            TST     @ADDR                   ; FORCE POLLING TO PACKET
2512                                    ;
2513 006740  004767  000020            CALL    POLL                    ; CALL POLL FOR DONE ( )
2514 006744  004767  000516            JSR     PC,INPKT                ;PRINT PACKET (DIAGNOSTICS)
2515 006750  004767  000076            CALL    RSPCHK                  ; CALL RESPOSE CHECK
2516 006754  103002                    BCC     99$                     ; IF ERROR
2517                                    ; : THEN
2518 006756  004767  000216            CALL    DUPERR                  ; :   CALL DUP ERROR
2519                                    ; ENDIF
2520 006762  000207            99$:    RETURN                          ;
2521
```

```
2533                            ;*****************************************************************************
2534                            ;                                                                            *
2535                            ;   POLL     WAIT FOR DONE                                                   *
2536                            ;                                                                            *
2537                            ;       AFTER DONE MAKE SURE THE PACKET WAS PROCESSED (NO HARD               *
2538                            ;       OR SOFT ERRORS) THEN RETURN.                                         *
2539                            ;                                                                            *
2540                            ;       INPUT:                                                               *
2541                            ;                                                                            *
2542                            ;       OUTPUT:                                                              *
2543                            ;                                                                            *
2544                            ;                                                                            *
2545                            ;*****************************************************************************
2546                            ;
2547 006764                    POLL:
     006764  010446                    MOV     R4,-(R6)
     006766  010546                    MOV     R5,-(R6)
2548 006770  000241                    CLC                             ; CLEAR ERROR FLAG
2549 006772  012704  000100            MOV     #100,R4                 ; SET TIME_OUT = 100
2550 006776                    1$:                                     ; BGNDO
2551 006776  012705  000010            MOV     #10,R5                  ; : SET DOWN_COUNTER = 10
2552 007002                    2$:                                     ; : BGNDO
2553 007002  104407  000000'          BREAK$,BEGIN                    ;TEMPORARY RETURN TO MONITOR...
     007006  104407  000000'          BREAK$,BEGIN                    ;THEN CONTINUE AT NEXT INSTRUCTION.
2554 007012  005305                    DEC     R5                      ; : :  DECREMENT DOWN_COUNTER
2555 007014  001372                    BNE     2$                      ; : DOUNTIL DOWN_COUNTER = 0
2556 007016  005304                    DEC     R4                      ; : DECREMENT TIME_OUT
2557                                                                    ; :
2558 007020  032767  100000  171234    BIT     #RG.OWN,RSPONC+2        ; DOUNTIL RESPONSE = DONE (WE OWN)
2559 007026  001406                    BEQ     99$                     ; OR
2560 007030  032777  100000  171214    BIT     #SA.ERR,@RQDXSA         ; SA = ERROR
2561 007036  001002                    BNE     99$                     ; OR
2562 007040  005704                    TST     R4                      ; TIME_OUT = 0
2563 007042  001355                    BNE     1$                      ;
2564 007044                    99$:                                    ;
2565 007044  012605                    MOV     (R6)+,R5
     007046  012604                    MOV     (R6)+,R4
2566 007050  000207                    RETURN                          ;
2567
2568
```

```
                        ;*****************************************************************************
                        ;
                        ;   CHECK RESPONSE AND COMMAND PACKETS
                        ;
                        ;       INPUT:
                        ;               END_CODE
                        ;
                        ;       OUTPUT:
                        ;               ERROR
                        ;
                        ;*****************************************************************************
                        ;
007052                  RSPCHK:                                         ;
007052  000241                  CLC                                     ; CLEAR ERROR
007054  005067  171026          CLR     ERRTYP                          ; ERROR TYPE = 0
007060  005067  171536          CLR     ERRFLG                          ; ERROR FLAGS = 0
007064  026767  171216  171406  CMP     RSPACK+P.OPCD,ENDCOD            ; IF RESPONSE CODE NOT = END CODE
007072  001405                  BEQ     1$                              ; : THEN
007074  000261                  SEC                                     ; :   SET ERROR
007076  012767  000001  171516  MOV     #1,ERRFLG                       ; :   END CODE DOES NOT MATCH
007104  000434                  BR      99$                             ; ENDIF
007106  026767  171164  171356  1$:     CMP     RSPACK,CMDREF           ; IF CMD REF NUMBER NOT = REF NUMBER
007114  001405                  BEQ     2$                              ; : THEN
007116  000261                  SEC                                     ; :   SET ERROR
007120  012767  000002  171474  MOV     #2,ERRFLG                       ; :   COMMAND REF NO NOT MATCH RESPOSE
007126  000423                  BR      99$                             ; ENDIF
007130  016700  171154          2$:     MOV     P.STS+RSPACK,R0         ; GET STATUS SUCCESS
007134  042700  177740          BIC     #177740,R0                      ; CLEAR UPPER 11 BITS OF SUB-STATUS
007140  001405                  BEQ     3$                              ; IF STATUS NOT = 0
                                                                        ; : THEN
007142  000261                  SEC                                     ; :   SET ERROR
007144  012767  000003  171450  MOV     #3,ERRFLG                       ; :   COMMAND NOT SUCCESSFUL
007152  000411                  BR      99$                             ; ENDIF
007154  032777  100000  171070  3$:     BIT     #SA,ERR,@RQDXSA         ; IF SA = ERROR
007162  001405                  BEQ     99$                             ; : THEN
007164  000261                  SEC                                     ; :   SET ERROR
007166  012767  000004  171426  MOV     #4,ERRFLG                       ; :   SA = ERROR
007174  000400                  BR      99$                             ; ENDIF
007176  000207                  99$:    RETURN                          ;
```

```
                                        ;*****************************************************************************
                                        ;                                                                            ;
                                        ;   DUP HARD ERROR                CARRY WILL BE SET                         ;
                                        ;                                                                            ;
                                        ;*****************************************************************************
                                        ;
007200                                  DUPERR:
007200  032767  000004  170610                  BIT     #SR.REP,SR1     ; IF REPORT ERROR = NO
007206  001403                                 BEQ     7$              ; : THEN
007210  005267  170630                         INC     HRLCNT          ; :  INCREMENT THE HARD ERROR
007214  000465                                 BR      99$             ; :
007216                                  7$:                             ; : ELSE
007216  004767  176314                         CALL    SETTAB          ; :  SET UP TABLE
                                                ;*************************************************************
007222  104405  000000' 000000                 HRDER$,BEGIN,NULL       ;
                                                ;*************************************************************
007230  004767  000140                         CALL    DERRPT          ; :  CALL DUP_ERROR_REPORT
                                                                        ; :  SELECT ERROR_FLAGS
007234  005367  171362                         DEC     ERRFLG          ; :  : CASE END CODE ERROR:
007240  001004                                 BNE     10$             ; :  :
007242  104402  000000' 010126'                 MSGS$,BEGIN,DUPMS1      ; ASCII MESSAGE CALL WITH NO HEADER
007250  000447                                 BR      99$             ; :  :
007252  005367  171344                 10$:    DEC     ERRFLG          ; :  : CASE COMMAND REF ERROR:
007256  001004                                 BNE     12$             ; :  :
007260  104402  000000' 010140'                 MSGS$,BEGIN,DUPMS2      ; ASCII MESSAGE CALL WITH NO HEADER
007266  000440                                 BR      99$             ; :  :
007270  005367  171326                 12$:    DEC     ERRFLG          ; :  : CASE NO SUCCESS:
007274  001004                                 BNE     14$             ; :  :
007276  104402  000000' 010152'                 MSGS$,BEGIN,DUPMS3      ; ASCII MESSAGE CALL WITH NO HEADER
007304  000431                                 BR      99$             ; :  :
007306  005367  171310                 14$:    DEC     ERRFLG          ; :  : CASE SA ERROR:
007312  001004                                 BNE     16$             ; :  :
007314  104402  000000' 010162'                 MSGS$,BEGIN,DUPMS4      ; ASCII MESSAGE CALL WITH NO HEADER
007322  000422                                 BR      99$             ; :  :
007324  005367  171272                 16$:    DEC     ERRFLG          ; :  : CASE SEND ERROR:
007330  001004                                 BNE     18$             ; :  :
007332  104402  000000' 010172'                 MSGS$,BEGIN,DUPMS5      ; ASCII MESSAGE CALL WITH NO HEADER
007340  000413                                 BR      99$             ; :  :
007342  005367  171254                 18$:    DEC     ERRFLG          ; :  : CASE RECIEVE ERROR:
007346  001004                                 BNE     50$             ; :  :
007350  104402  000000' 010204'                 MSGS$,BEGIN,DUPMS6      ; ASCII MESSAGE CALL WITH NO HEADER
007356  000404                                 BR      99$             ; :  :
007360                                  50$:                            ; :  : DEFAULT : DUP ERROR
007360  104402  000000' 010216'                 MSGS$,BEGIN,DUPMSF      ; ASCII MESSAGE CALL WITH NO HEADER
007366  000400                                 BR      99$             ; :  :
                                                                        ; :  ENDSELECT
007370                                  99$:                            ; ENDIF
007370  000261                                 SEC                     ; SET ERROR EXIT
007372  000207                                 RTS     PC              ; RETURN TO CYCLED
```

```
                                ;*******************************************************************************
                                ;                                                                              ;
                                ;   DUP ERROR REPORT                                                           ;
                                ;                                                                              ;
                                ;       INPUT:                                                                 ;
                                ;                                                                              ;
                                ;       OUTPUT: REPORT ERROR                                                   ;
                                ;                                                                              ;
                                ;               PRINT  STATUS, OPCODE, UNIT NUMBER, DBN )                      ;
                                ;                                                                              ;
                                ;*******************************************************************************
                                ;
007374                          DERRPT:                                 ;
                                        ;*******************************************************************
                                                                        ;CONVERT P.STS+RSPACK TO ASCII AND
                                                                        ;STORE AT ADR1
007374  104420  000000' 000310'         OTOA$,BEGIN,P.STS+RSPACK,ADR1
007402  000530'
                                        ;*******************************************************************
                                        ;*******************************************************************
                                                                        ;CONVERT P.OPCD+RSPACK TO ASCII AND
                                                                        ;STORE AT ADR2
007404  104420  000000' 000306'         OTOA$,BEGIN,P.OPCD+RSPACK,ADR2
007412  000537'
                                        ;*******************************************************************
007414  011467  171074                 MOV     (R4),NUM                ;   GET UNIT #
                                        ;*******************************************************************
                                                                        ;CONVERT NUM TO ASCII AND
                                                                        ;STORE AT ADR3
007420  104420  000000' 000514'         OTOA$,BEGIN,NUM,ADR3
007426  000546'
                                        ;*******************************************************************
                                        ;*******************************************************************
                                                                        ;CONVERT DBN TO ASCII AND
                                                                        ;STORE AT ADR4
007430  104420  000000' 000624'         OTOA$,BEGIN,DBN,ADR4
007436  000555'
                                        ;*******************************************************************
007440  104403  000000' 01027 '         MSGN$,BEGIN,BANER2      ;ASCII MESSAGE CALL WITH COMMON HEADER
007446  000207                          RETURN                          ;
```

```
                                        ;*****************************************************************************
                                        ;
                                        ;   INPKT AND OUTPKT - PRINT RESPONSE AND COMMAND PACKETS
                                        ;
                                        ;*****************************************************************************
007450  010246                OUTPKT: MOV     R2,-(SP)
007452  012702  000376'              MOV     #CMPACK,R2
007456  012767  052517  001074      MOV     #"OU,RY1
007464  000406                      BR      COM
007466  010246                INPKT:  MOV     R2,-(SP)
007470  012702  000276'              MOV     #RSPACK,R2
007474  012767  047111  001056      MOV     #"IN,RY1

007502  032767  000001  170306  COM:    BIT     #SR.PAK,SR1             ;PRINT PACKET INFORMATION?
007510  001460                      BEQ     10$
007512  010046                      MOV     R0,-(SP)
007514  012700  000004              MOV     #4,R0

007520  104403  000000' 010102'      MSGN$,BEGIN,PAKHED      ;ASCII MESSAGE CALL WITH COMMON HEADER
007526  012267  170762          1$:     MOV     (R2)+,NUM
                                        ;*****************************************************************************
                                                                ;CONVERT NUM TO ASCII AND
                                                                ;STORE AT ADR3
007532  104420  000000' 000514'      OTOA$,BEGIN,NUM,ADR3
007540  000546
                                        ;*****************************************************************************
007542  012267  170746              MOV     (R2)+,NUM
                                        ;*****************************************************************************
                                                                ;CONVERT NUM TO ASCII AND
                                                                ;STORE AT ADR4
007546  104420  000000' 000514'      OTOA$,BEGIN,NUM,ADR4
007554  000555'
                                        ;*****************************************************************************
007556  012267  170732              MOV     (R2)+,NUM
                                        ;*****************************************************************************
                                                                ;CONVERT NUM TO ASCII AND
                                                                ;STORE AT ADR5
007562  104420  000000' 000514'      OTOA$,BEGIN,NUM,ADR5
007570  000564'
                                        ;*****************************************************************************
007572  012267  170716              MOV     (R2)+,NUM
                                        ;*****************************************************************************
                                                                ;CONVERT NUM TO ASCII AND
                                                                ;STORE AT ADR6
007576  104420  000000' 000514'      OTOA$,BEGIN,NUM,ADR6
007604  000573'
                                        ;*****************************************************************************
007606  012267  170702              MOV     (R2)+,NUM
                                        ;*****************************************************************************
                                                                ;CONVERT NUM TO ASCII AND
                                                                ;STORE AT ADR7
007612  104420  000000' 000514'      OTOA$,BEGIN,NUM,ADR7
007620  000602'
                                        ;*****************************************************************************
007622  012267  170666              MOV     (R2)+,NUM
                                        ;*****************************************************************************
                                                                ;CONVERT NUM TO ASCII AND
```

MODULE CODE

```
                                                        ;STORE AT ADR8
        007626  104400  000000' 000514'          OTOA$,BEGIN,NUM,ADR8
        007634  000611'
                                                ;*******************************************************************
2703    007636  104402  000000' 010240'          MSG$$,BEGIN,PACKET      ; ASCII MESSAGE CALL WITH NO HEADER
2704
2705    007644  005300                          DEC     R0
        007646  001377                          BNE     1$
2706    007650  012600                          MOV     (SP)+,R0
2707    007652  012602                  10$:    MOV     (SP)+,R2
2708    007654  000207                          RTS     PC
2709
```

SYMBOL TABLE

```
OP.AVL= 000010  PRTY5 = 000240  QMON22= 000010  SA.CME= 000360  ST.ABO= 000002
OP.CCD= 000021  PRTY6 = 000300  RANACC  004050R  SA.CTP= 003400  ST.ALO= 000006
OP.CMP= 000040  PRTY7 = 000340  RAND$ = 104417  SA.DIA= 000400  ST.AVL= 000004
OP.DAP= 000013  PS    = 177776  RANNUM  000054R  SA.ERC= 003777  ST.CMD= 000001
OP.ELP= 000003  PSW   = 177776  RBUF    012462R  SA.ERR= 100000  ST.CMP= 000007
OP.END= 000200  PUSH  = 005746  RBUFEA  000130R  SA.GO = 000001  ST.CNT= 000012
OP.ERL= 000101  PUSH2 = 024646  RBUFPA  000126R  SA.INE= 000200  ST.DAT= 000010
OP.ERS= 000022  PWRFLG= 000002  RBUFSZ  000132R  SA.INT= 000200  ST.DIA= 000037
OP.ESP= 000002  P.ADEA= 000022  RBUFVA  000124R  SA.LFC= 040000  ST.DRV= 000013
OP.FLU= 000023  P.ADPA= 000020  RCVDAT  006572R  SA.PRG= 000001  ST.FAL= 000005
OP.GCS= 000002  P.BCNT= 000014  RDBN    006504R  SA.QB = 001000  ST.HST= 000011
OP.GDS= 000001  P.BUFF= 000020  RDONLY  010062R  SA.RSE= 000017  ST.MFE= 000005
OP.GUS= 000003  P.CMST= 000020  RDWRT   010050R  SA.RSP= 007400  ST.MSK= 000037
OP.MRD= 000030  P.CNCL= 000022  READ    004466R  SA.S1 = 004000  ST.NON= 000004
OP.MWR= 000031  P.CNT = 000006  READA   004520R  SA.S2 = 010000  ST.NRA= 000002
OP.ONL  000011  P.CNTF= 000016  REDDBN  006176R  SA.S3 = 020000  ST.NRS= 000003
OP.RCD= 000005  P.CNTI= 000024  RESTRT  001124R  SA.S4 = 040000  ST.OFL= 000003
OP.RD = 000041  P.CPSP= 000042  RESTR1  001260R  SA.VCE= 000177  ST.SUB= 000040
OP.RPL= 000024  P.CRF = 000000  RESTR2  001234R  SA.VEC= 000177  ST.SUC= 000000
OP.SCC= 000004  P.CTMO= 000020  RES1    000056R  SBADR   000102R  ST.WPR= 000006
OP.SHC= 000102  P.CYL = 000050  RES2    000060R  SCC     004614R  SVR0    000062R
OP.SND= 000004  P.DEV = 000034  RG.FLG= 040000  SC.DIS= 000400  SVR1    000064R
OP.SUC= 000012  P.ELGF= 000034  RG.OWN= 100000  SC.DUP= 000200  SVR2    000066R
OP.WR = 000042  P.FBBK= 000034  RH70  = 001000  SC.INV= 000100  SVR3    000070R
OTOA$ = 104420  P.FLGS= 000011  RINTR   000256R  SC.IOP= 000100  SVR4    000072R
OUTPKT  007450R  P.GRP = 000046  RLIM  = 000004  SC.NVL= 000040  SVR5    000074R
PA      000504R  P.HSTI= 000020  RQDXSA  000252R  SC.STO= 000040  SVR6    000076R
PACKET  010240R  P.HTMO= 000020  RSECH   000636R  SEND    004704R  SYSCNT  000052R
PAKHED  010102R  P.LBN = 000034  RSECL   000634R  SEQACC  004010R  S.EXT = 000014
PARPRE= 002000  P.LGDT= 000014  RSPACK  000276R  SETTAB  005536R  S.FLAG= 000017
PASCNT  000034R  P.MEDI= 000034  RSPCHK  007052R  SETUP   003032R  S.PRGS= 000020
PDPF11= 000002  P.MLUN= 000014  RSPLEN  000272R  SE.ACT= 000010  S.TEXT= 000000
PDPLSI= 020000  P.MOD = 000012  RSPONC  000260R  SE.DIS= 000001  S.TOUT= 000024
PDP44 = 100000  P.OPCD= 000010  RSPVIR  000274R  SE.IDL= 000000  TABLEW  000656R
PDP60 = 004000  P.OTRF= 000014  RSTRT   000112R  SE.LOC= 000002  TEMP    000530R
PDP70 = 010000  P.OVRL= 000034  RVA     000756R  SE.SUP= 000004  TEND    000756R
PH.ALO= 000001  P.RBN = 000014  RY0     010550R  SFTFLG  000626R  TIMER = 027340
PH.DLG= 000010  P.RBNS= 000056  RY1     010560R  SNDDAT  006512R  TIMOUT= 005670
PH.OVR= 000002  P.RCTC= 000057  RY3     010564R  SNDDUP  006652R  TRPDFD= 000023
PH.RW = 000004  P.RCTS= 000054  RY4     010574R  SNDSTP  002554R  TRY     000646R
PICKBK  004000R  P.RGID= 000034  RY5     010612R  SOFCNT  000042R  TSTOFL  003330R
PIRQ$ = 000004  P.RGOF= 000040  RY6     010642R  SOFER$= 104406  UF.CMR= 000001
PKTSIZ= 000060  P.RSVD= 000011  RY7     010670R  SOFPAS  000046R  UF.CMW= 000002
POLL    006764R  P.SFTW= 000040  RY8     010714R  SPOINT  000032R  UF.INA= 040000
POPSP = 005726  P.SHST= 000042  R.MESG= 000000  SPSIZ = 000040  UF.RMV= 000200
POPSP2= 022626  P.SHUN= 000040  R.TEXT= 000002  SR.CMP= 004000  UF.RPL= 010000
PRHMS$= 000002  P.STS = 000012  R.TYPE= 000001  SR.PAR= 000001  UF.SCH= 004000
PRINTE  005562R  P.SZOF= 000012  R18EA   000762R  SR.REP= 000004  UF.SCL = 002000
PRNMSG  000526R  P.TIME= 000024  R18PA   000760R  SR.RON= 000000  UF.WBN= 000040
PRTNUM= 000017  P.TRCK= 000044  R22EA   000766R  SR.SEQ= 002000  UF.WPE= 020000
PRTSTS  001742R  P.UNFL= 000016  R22PA   000764R  SR.SUM= 000010  UF.WPS= 010000
PRTY  = 000000  P.UNIT= 000004  R6    =%000006  SR1     000016R  UF.Y76= 000004
PRTY0 = 000000  P.UNSZ= 000044  R7    =%000007  SR2     000020R  UNITOF  010014R
PRTY1 = 000040  P.UNTI= 000024  SANOTO  010004R  SR3     000022R  UNITPT  000650R
PRTY2 = 000100  P.USER= 000026  SAVEA   000524R  SR4     000024R  UNSZM   000044R
PRTY3 = 000140  P.VRSN= 000014  SAVPA   000522R  START   001000R  UNSZL   000040R
PRTY4 = 000200  P.VSER= 000050  SA.CMD= 170000  STAT    000026R  UPPER = 000002
```

SYMBOL TABLE

```
USTACK= 000001          WBUFRQ  000140R         W18EA   000774R         .C.ABO= 000014          .R.RCD= 000020
VA      000502R         WBUFSZ  000142R         W18PA   000772R         .C.DUS= 000014          .R.SND= 000020
VECTOR  000010R         WDBN    006476R         W22EA   001000R         .C.LOC= 000022          .R.SUP= 000014
WAIT    004764R         WDER    000116R         W22PA   000776R         .C.RCD= 000034          .T.DEF= 000040
WARN1   010024R         WDTO    000114R         XFLAG   000005R         .C.SND= 000034          .T.FAT= 000120
WARN2   010030R         WORK    000652R         XSECH   000632R         .C.SUP= 000050          .T.INF= 000060
WARN3   010034R         WRITE   004416R         XSECL   000630R         .R.ABO= 000014          .T.QUE= 000070
WASADR  000104R         WRITEA  004430R         ZERO    010044R         .R.DUS= 000026          .T.SPL= 000140
WBUFEA  000136R         WRTDBN  005762R         $$      = 000000        .R.LOC= 000020          .T.TER= 000100
WBUFPA  000134R         WVA     000770R

. ABS.  000000    000
        013472    001
ERRORS DETECTED:  0
VIRTUAL MEMORY USED:  13795 WORDS  ( 54 PAGES)
DYNAMIC MEMORY:  19748 WORDS  ( 75 PAGES)
ELAPSED TIME:  00:02:14
XRQADO,XRQADO/CR/-SP=DDXCOM,XRQADO
```

DEC/X11 SYSTEM EXERS....B1
DEC/X11 SYSTEM EXERS....C1
DEC/X11 SYSTEM EXERS....D1
DEC/X11 SYSTEM EXERS....E1
DEC/X11 SYSTEM EXERS....F1
DEC/X11 SYSTEM EXERS....G1
DEC/X11 SYSTEM EXERS....H1
DEC/X11 SYSTEM EXERS....I1
DEC/X11 SYSTEM EXERS....J1
DEC/X11 SYSTEM EXERS....K1
MODULE HEADER BLOCK L1
MODULE HEADER BLOCK M1
MODULE STORAGE AREA N1

MORE MODULE STORAGE B2
MODULE PRIVATE DATA C2
EQUATES D2
RQDX BIT DEFINITIONS....E2
COMMAND/MESSAGE DESC....F2
COMMAND MODIFIERS G2
CONTROLLER FLAGS H2
END PACKET OFFSETS I2
ERROR LOG FLAGS ...J2
DUP DEFINITIONS ...K2
OFFSET AND FLAG DEFI....L2
OFFSET AND FLAG DEFI....M2
OFFSET AND FLAG DEFI....N2

MODULE CODE J3
MODULE CODE C3
MODULE CODE D3
MODULE CODE E3
MODULE CODE F3
MODULE CODE G3
MODULE CODE H3
MODULE CODE I3
MODULE CODE J3
MODULE CODE K3
MODULE CODE L3
MODULE CODE M3
MODULE CODE N3

MODULE CODE B4
MODULE CODE C4
MODULE CODE D4
MODULE CODE E4
MODULE CODE F4
MODULE CODE G4
MODULE CODE H4
MODULE CODE I4
MODULE CODE J4
MODULE CODE K4
MODULE CODE L4
MODULE CODE M4
MODULE CODE N4

MODULE CODE B5
MODULE CODE C5
MODULE CODE D5
MODULE CODE E5
MODULE CODE F5
MODULE CODE G5
MODULE CODE H5
MODULE CODE I5
MODULE CODE J5
MODULE CODE K5
MODULE CODE L5
MODULE CODE M5
MODULE CODE N5

MODULE CODE B6
MODULE MESSAGES C6
MODULE MESSAGES D6
MODULE MESSAGES E6
MODULE MESSAGES F6
MORE MODULE MESSAGES....G6
MORE MODULE MESSAGES....H6
SYMBOL TABLE I6
SYMBOL TABLE J6
SYMBOL TABLE K6

```
;    0001  0     module ZRQAM1 (
;    0002  0
;    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                 ident = 'V01.6',
;    0005  0                 addressing_mode (absolute),
;    0006  0                 environment (noeis)
;    0007  0                 ) ;
;    0008  0
;    0009  1     begin
;    0010  1
;    0011  1
; C  0012  1     %(
; C  0013  1
; C  0014  1                                     IDENTIFICATION
; C  0015  1
; C  0016  1             PRODUCT CODE:           AC-Y398D-MC
; C  0017  1
; C  0018  1             PRODUCT NAME:           C/RQADO RQDX1/RUX50 EXERCISER
; C  0019  1
; C  0020  1             PRODUCT DATE:           11 APR 84
; C  0021  1
; C  0022  1             MAINTAINER:             DIAGNOSTIC ENGINEERING
; C  0023  1
; C  0024  1             AUTHOR:                 RAVINDER K. KARWAN
; C  0025  1                                     BOB POWERS
; C  0026  1
; C  0027  1
; C  0028  1             Copyright (C) 1983, 1984
; C  0029  1
; C  0030  1             Digital Equipment Corporation, Maynard, Massachusetts  01754
; C  0031  1
; C  0032  1             This software is furnished under a license for use only on a single
; C  0033  1             computer system and may be copied only with the inclusion of the
; C  0034  1             above copyright notice.  This software, or any other copies thereof,
; C  0035  1             may not be provided or otherwise made available to any other person
; C  0036  1             except for use on such system and to one who agrees to these license
; C  0037  1             terms.  Title to and ownership of the software shall at all times
; C  0038  1             remain in DEC.
; C  0039  1
; C  0040  1             the information in this document is subject to change without notice
; C  0041  1             and should not be construed as a commitment by Digital Equipment
; C  0042  1             Corporation.
; C  0043  1
; C  0044  1             DEC assumes no responsibility for the use or reliability of its
; C  0045  1             software on equipment which is not supplied by DEC.
; C  0046  1
; C  0047  1
; C  0048  1             The following are trademarks of Digital Equipment Corporation:
; C  0049  1
; C  0050  1             DIGITAL         PDP             UNIBUS          MASSBUS
; C  0051  1             DEC             DECUS           DECTAPE
```

```
;  C 0052  1
;  C 0053  1
;  C 0054  1
;  C 0055  1
;  C 0056  1
;  C 0057  1
;  C 0058  1
;  C 0059  1                    TABLE OF CONTENTS
;  C 0060  1
;  C 0061  1
;  C 0062  1
;  C 0063  1          1.0       GENERAL INFORMATION
;  C 0064  1          1.1       PROGRAM ABSTRACT
;  C 0065  1          1.2       SYSTEM REQUIREMENTS
;  C 0066  1          1.2.1     HARDWARE REQUIREMENTS
;  C 0067  1          1.2.2     SOFTWARE REQUIREMENTS
;  C 0068  1          1.3       RELATED DOCUMENTS AND STANDARDS
;  C 0069  1          1.4       DIAGNOSTIC HIERARCHY PREREQUISITES
;  C 0070  1          1.5       ASSUMPTIONS
;  C 0071  1          1.6       MEMORY MAP
;  C 0072  1
;  C 0073  1          2.0       OPERATING INSTRUCTIONS
;  C 0074  1          2.1       HARDWARE QUESTIONS
;  C 0075  1          2.2       SOFTWARE QUESTIONS
;  C 0076  1
;  C 0077  1          3.0       ERROR TYPES
;  C 0078  1          3.1       ERROR INFORMATION
;  C 0079  1          3.2       INITIALIZATION ERRORS
;  C 0080  1          3.3       EXERCISER ERRORS
;  C 0081  1          3.4       ERROR LOG MESSAGES
;  C 0082  1          3.5       MSCP ERRORS
;  C 0083  1          3.6       SAMPLE ERROR STATEMENT
;  C 0084  1
;  C 0085  1          4.0       PERFORMANCE AND PROGRESS REPORTS
;  C 0086  1
;  C 0087  1          5.0       TEST SUMMARY
;  C 0088  1          5.1       INITIALIZATION SUBTEST
;  C 0089  1          5.2       EXERCISER
;  C 0090  1          5.3       DROP UNIT SUMMARY
;  C 0091  1
;  C 0092  1          6.0       ERROR LIST
;  C 0093  1
;  C 0094  1          7.0       DATA PATTERNS
```

```
; C 0095  1
; C 0096  1
; C 0097  1
; C 0098  1
; 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
```

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

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

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE REQUIREMENTS

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

1.2.2 SOFTWARE REQUIREMENTS

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

1.3 RELATED DOCUMENTS AND STANDARDS

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

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

NONE

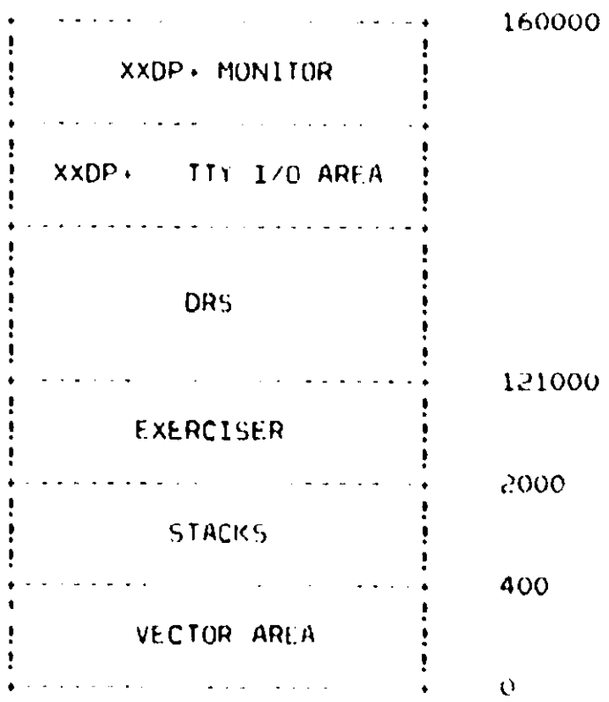
1.5 ASSUMPTIONS

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

```
;  C 0146  1
;  C 0147  1
;  C 0148  1
;  C 0149  1
;  C 0150  1
;  C 0151  1
;  C 0152  1
;  C 0153  1
;  C 0154  1
;  C 0155  1
;  C 0156  1
;  C 0157  1
;  C 0158  1
;  C 0159  1
;  C 0160  1
;  C 0161  1
;  C 0162  1
;  C 0163  1
;  C 0164  1
;  C 0165  1
;  C 0166  1
;  C 0167  1
;  C 0168  1
;  C 0169  1
;  C 0170  1
;  C 0171  1
;  C 0172  1
;  C 0173  1
;  C 0174  1
;  C 0175  1
;  C 0176  1
;  C 0177  1
;  C 0178  1
;  C 0179  1
;  C 0180  1
;  C 0181  1
;  C 0182  1
;  C 0183  1
;  C 0184  1
;  C 0185  1
```

1.6 MEMORY MAP

Memory layout on 28k machine - XXDP environment



In a machine with more memory, free space will occur between the exerciser and the DRS.

```
; C 0186  1
; C 0187  1
; C 0188  1
; C 0189  1
; C 0190  1
; C 0191  1
; C 0192  1
; C 0193  1
; C 0194  1
; 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
```

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 anuwers 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 (0) ?

   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 (0) 172150 ?

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

```
; C 0235  1              4.  VECTOR ADDRESS (O) 154 ?
; C 0236  1
; C 0237  1                  Answer with the interrupt vector ofthe same RQDX1 or RUX50
; C 0238  1                  controller described in the above question. A vector address
; C 0239  1                  in the range of 4 to 774 may be specified. 154 is the default.
; C 0240  1
; C 0241  1              5.  BR LEVEL (D) 4 ?
; C 0242  1
; C 0243  1                  Answer with the bus request interrupt level used by  the
; C 0244  1                  above controller. Levels 4 through 7 are acceptable. 4 is the
; C 0245  1                  default.
; C 0246  1
; C 0247  1
; C 0248  1              6.  DRIVE NUMBER (D) 0 ?
; C 0249  1
; C 0250  1                  Enter the logical unit number for one drive associated
; C 0251  1                  with the IP address above. Drive numbers are in the
; C 0252  1                  range of 0 through 15. The number entered here must
; C 0253  1                  match the unit plug on the front  panel of the drive, and
; C 0254  1                  must be within the range implied by the jumper (L)NO 7)
; C 0255  1                  on the RQDX1 or RUX50 controller board. 0 is the default
; C 0256  1                  answer.
; C 0257  1
; C 0258  1
; C 0259  1              7.  TEST ENTIRE CUSTOMER DATA AREA OF THIS DISK (L) Y?
; C 0260  1
; C 0261  1                  This question is asked to give the opportunity of limiting the
; C 0262  1                  addressing range over which the testing will be performed.
; C 0263  1                  An affirmative answer will cause no limits to be imposed for
; C 0264  1                  the unit in question. A negative answer will cause limits
; C 0265  1                  to be imposed, as defined by the following four questions.
; C 0266  1
; C 0267  1
; C 0268  1              8.  LOWER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?
; C 0269  1
; C 0270  1                  Enter in octal the less significant 16 bit word of the lowest
; C 0271  1                  LBN address in the test range. The value may be from 000000
; C 0272  1                  to 177777.
; C 0273  1
; C 0274  1
; C 0275  1              9.  HIGHER OCTAL WORD OF BEGINNING LBN ADDRESS (O) 0?
; C 0276  1
; C 0277  1                  Enter in octal the more significant 16-bit word of the
; C 0278  1                  lowest LBN address in the test range.
; C 0279  1
; C 0280  1
; C 0281  1             10.  LOWER OCTAL WORD OF ENDING LBN ADDRESS (O) 150477?
; C 0282  1
; C 0283  1                  Enter in octal the less significant 16-bit word of the
; C 0284  1                  highest LBN address in the test range. 150477 is the
; C 0285  1                  highest LBN address for an RD52.
; C 0286  1
; C 0287  1
```

```
; C 0288  1
; C 0289  1
; C 0290  1
; C 0291  1
; C 0292  1
; C 0293  1
; C 0294  1
; C 0295  1
; C 0296  1
; C 0297  1
; C 0298  1
; C 0299  1
; C 0300  1
; C 0301  1
; C 0302  1
; C 0303  1
; C 0304  1
```

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

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

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

```
; C 0305  1
; C 0306  1
; C 0307  1
; C 0308  1
; C 0309  1
; C 0310  1
; C 0311  1
; C 0312  1
; C 0313  1
; C 0314  1
; C 0315  1
; C 0316  1
```

12. EXERCISE ON CUSTOMER DATA AREA ON THIS DISK UNIT (Y) ?

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

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

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

```
; C 0317  1
; C 0318  1
; C 0319  1
; C 0320  1
; C 0321  1
; C 0322  1
; C 0323  1
; C 0324  1
; C 0325  1
; C 0326  1
; C 0327  1
; C 0328  1
; C 0329  1
; C 0330  1
; C 0331  1
; C 0332  1
; C 0333  1
; C 0334  1
; C 0335  1
; C 0336  1
; C 0337  1
; C 0338  1
; C 0339  1
; C 0340  1
; C 0341  1
; C 0342  1
; C 0343  1
; C 0344  1
; C 0345  1
; C 0346  1
; C 0347  1
; C 0348  1
; C 0349  1
; C 0350  1
; C 0351  1
; C 0352  1
; C 0353  1
; C 0354  1
; C 0355  1
; C 0356  1
; C 0357  1
; C 0358  1
; C 0359  1
; C 0360  1
; C 0361  1
; C 0362  1
```

2.2 SOFTWARE QUESTIONS

Software Parameter Questions

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

1. CHANGE SW (L) Y ?

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

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

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

3. ERROR LIMIT (D) 32 ?

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

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

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

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

```
; C 0363  1
; C 0364  1
; C 0365  1
; C 0366  1
; C 0367  1
; C 0368  1
; C 0369  1
; C 0370  1
; C 0371  1
; C 0372  1
; C 0373  1
; C 0374  1
; C 0375  1
; C 0376  1
; C 0377  1
; C 0378  1
; C 0379  1
; C 0380  1
; C 0381  1
; C 0382  1
; C 0383  1
; C 0384  1
; C 0385  1
; C 0386  1
; C 0387  1
; C 0388  1
; C 0389  1
; C 0390  1
; C 0391  1
; C 0392  1
; C 0393  1
; C 0394  1
; C 0395  1
; C 0396  1
; C 0397  1
; C 0398  1
; C 0399  1
; C 0400  1
; C 0401  1
; C 0402  1
; C 0403  1
; C 0404  1
; C 0405  1
; C 0406  1
; C 0407  1
; C 0408  1
; C 0409  1
; C 0410  1
; C 0411  1
; C 0412  1
; C 0413  1
```

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

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

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

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

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

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

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

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

9. HALT ON OTHER HARD ERRORS (#s 33, 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 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
; 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
```

10. HALT ON SOFT ERRORS (0s 50-54) (L) N ?

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

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

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

12. RANDOM SEEK MODE (L) Y ?

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

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

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

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

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

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

    THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED DISK UNITS.

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

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

```
; C 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
; C 0492  1
; C 0493  1
; C 0494  1
; C 0495  1
; C 0496  1
; C 0497  1
; C 0498  1
```

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

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

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

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

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

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

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

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

```
; C 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
; C 0541  1
; C 0542  1
; C 0543  1
; C 0544  1
; C 0545  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

```
; C 0595  1
; C 0596  1
; C 0597  1
; C 0598  1
; C 0599  1
; C 0600  1
; C 0601  1
; C 0602  1
; C 0603  1
; C 0604  1
; C 0605  1
; C 0606  1
; C 0607  1
; C 0608  1
; C 0609  1
; C 0610  1
; C 0611  1
; C 0612  1
; C 0613  1
; C 0614  1
; C 0615  1
; C 0616  1
; C 0617  1
; C 0618  1
; C 0619  1
; C 0620  1
; C 0621  1
; C 0622  1
; C 0623  1
; C 0624  1
; C 0625  1
; C 0626  1
; C 0627  1
; C 0628  1
; C 0629  1
; C 0630  1
; C 0631  1
; C 0632  1
; C 0633  1
; C 0634  1
; C 0635  1
; C 0636  1
; C 0637  1
; C 0638  1
; C 0639  1
; C 0640  1
; C 0641  1
```

3.3 EXERCISER ERRORS

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

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

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

3.4 ERROR LOG MESSAGES

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

3.5 MSCP ERRORS

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

```
; C 0642  1          3.6   SAMPLE ERROR STATEMENT
; C 0643  1                ------------------------
; C 0644  1
; C 0645  1
; C 0646  1          The errors listed by the exerciser are usually very des
; C 0647  1          criptive and are self explanatory.  The following is an example
; C 0648  1          error statement.  This error statement is the extended error
; C 0649  1          message.
; C 0650  1
; C 0651  1
; C 0652  1          (example)                          (comments)
; C 0653  1
; C 0654  1          DISK XXX                           !DISK UNIT NUMBER
; C 0655  1          INVALID COMMAND                    !MAJOR STATUS CODE RECEIVED BACK
; C 0656  1          SUB-CODE XXXX                      !SUB-CODE OF GIVEN COMMAND
; C 0657  1          COMMAND: READ                      !COMMAND GIVEN TO DRIVE
; C 0658  1          LBN: XXXXX                         !LOGICAL BLOCK NUMBER GIVEN
; C 0659  1          BYTE COUNT IN COMMAND XXXXX        !NUMBER OF BYTES WANTED TO READ
; C 0660  1          ACTUAL # OF BYTES TRANSFERRED XXXXX !NUMBER OF BYTES ACTUALLY READ
; C 0661  1
; C 0662  1
; C 0663  1
; C 0664  1          The status code in an end messages is broken into two
; C 0665  1          pieces. The first 5 bits represent the major status which is given
; C 0666  1          by the "invalid command" message.  The 11 remaining bits represent
; C 0667  1          the sub-code, which tells in greater detail the error in the con
; C 0668  1          troller.  The LBN is the logical block on the disk the controller
; C 0669  1          was trying to read.  The byte count refers to the number of bytes
; C 0670  1          the controller was going to read off the LBN. The actual number of
; C 0671  1          bytes transferred refers to the number of bytes read before the
; C 0672  1          error.
```

```
; 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
; C 0699  1
; C 0700  1
; C 0701  1
; C 0702  1
; C 0703  1
; C 0704  1
; C 0705  1
; C 0706  1
```

4.0 PERFORMANCE AND PROGRESS REPORTS

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

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

The sub categories are:

1. disk related errors

2. seek (or format) related errors

3. controller or drive related errors

4. host (the CPU) related errors.

All numeric values are in decimal radix.

UNT #	TYPE	# OF READS	BYTS READ	# OF WRITES	BYTES WRITTEN	HRD ERS DAT	HRD ERS SEK	HRD ERS DRV	HRD ERS HST	SFT ERS DAT	SFT ERS SEK	SFT ERS DRV	SFT ERS HST
X	XXXX	XXXX	XXXX	XXXXX	XXXXXX	X	X	X	X	X	X	X	X
.	..	..	..	...	....	.	.	.	.	.	.	.	.
.	..	..	..	...	....	.	.	.	.	.	.	.	.

```
; C 0707  1
; C 0708  1
; C 0709  1
; C 0710  1
; C 0711  1
; C 0712  1
; C 0713  1
; C 0714  1
; C 0715  1
; C 0716  1
; C 0717  1
; C 0718  1
; C 0719  1
; C 0720  1
; C 0721  1
; C 0722  1
; C 0723  1
; C 0724  1
; C 0725  1
; C 0726  1
; C 0727  1
; C 0728  1
; C 0729  1
; C 0730  1
; C 0731  1
; C 0732  1
; C 0733  1
; C 0734  1
; C 0735  1
; C 0736  1
; C 0737  1
; C 0738  1
; C 0739  1
; C 0740  1
; C 0741  1
; C 0742  1
; C 0743  1
; C 0744  1
; C 0745  1
; C 0746  1
; C 0747  1
; C 0748  1
; C 0749  1
; C 0750  1
; C 0751  1
; C 0752  1
; C 0753  1
; C 0754  1
; C 0755  1
; C 0756  1
```

5.0 TEST SUMMARY

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

5.1 INITIALIZATION SUBTEST

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

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

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

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

5.2 EXERCISER

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

```
;  C 0757  1              If a read/write error occurs during this test,  then the
;  C 0758  1              controller will initiate an  appropriate number of re-
;  C 0759  1              tries.  If all retries fail, then a hard error will be repor-
;  C 0760  1              ted to the host,  an error message  will  be displayed on the
;  C 0761  1              console  terminal and  the error will be tallied for the sum
;  C 0762  1              mary report. The unit will be dropped if the hard error count
;  C 0763  1              has exceeded the specified limit.
;  C 0764  1
;  C 0765  1
;  C 0766  1
;  C 0767  1          5.3   DROP UNIT SUMMARY
;  C 0768  1                ----------------
;  C 0769  1              During the Initialization Subtest, individual units will
;  C 0770  1              be dropped from  the test  sequence if they are unable  to be
;  C 0771  1              brought online or the operator specified drive does not match
;  C 0772  1              the hardware.
;  C 0773  1
;  C 0774  1              During  the Exercise,  the program will  drop a unit for
;  C 0775  1              one of  three reasons.  The normal path is for  each unit  to
;  C 0776  1              complete the transfer of N megabytes, where N is specified by
;  C 0777  1              the  operator  during SW  questioning  and  be  soft-dropped.
;  C 0778  1              Otherwise,  a unit will be hard-dropped if the number of hard
;  C 0779  1              errors encountered exceeds  the operator-specified limit,  or
;  C 0780  1              if a fatal error is detected. Units hard-dropped may later be
;  C 0781  1              added to the  test cycle.  However,  statistics for the hard
;  C 0782  1              added  unit  will be  cleared to  zero;  if a transfer  limit
;  C 0783  1              was specified,  in which case the unit was soft dropped,  the
;  C 0784  1              statistics may  or may not be cleared  depending on the oper-
;  C 0785  1              ators answer to Software question 12.
```

```
; 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
; C 0828  1
; C 0829  1
; C 0830  1
; C 0831  1
; C 0832  1
; C 0833  1
; C 0834  1
```

6.0 ERROR CODES GENERATED BY THIS EXERCISER

SYSTEM FATAL ERRORS

1 More than 4 units specified

DRIVE FATAL ERRORS

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

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

HARD ERRORS

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

```
; C 0888  1                41  Drive error.                    See #40.
; C 0889  1
; C 0890  1                42  Host write compare error.       Error detected when Host
; C 0891  1                                                    CPU compared the data
; C 0892  1                                                    written and read back. May
; C 0893  1                                                    be a problem with the Host
; C 0894  1                                                    or Controller RAM.
; C 0895  1
; C 0896  1                43 Message from internal diagnostics  See #40.
; C 0897  1
; C 0898  1                44  Duplicate unit number detected  ---
; C 0899  1                    by the Controller.
; C 0900  1
; C 0901  1                45  Unknown end code received.      Problem with the Control-
; C 0902  1                                                    ler microcode or the port/
; C 0903  1                                                    DMA interface.
; C 0904  1
; C 0905  1
; C 0906  1                SOFT ERRORS
; C 0907  1                ------------
; C 0908  1
; C 0909  1                50  Controller error.              See error-log packet for
; C 0910  1                                                    details as the exact cause
; C 0911  1                                                    may not be evident.
; C 0912  1
; C 0913  1                51  Host memory access error.      See #50.
; C 0914  1
; C 0915  1                52  Disk transfer error.           See #50.
; C 0916  1
; C 0917  1                53  'Standard Disk Interconnect'   See #50.
; C 0918  1                    error.
; C 0919  1
; C 0920  1                54  'Small Disk' error.            See #50.
```

7.0 DATA PATTERNS

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

```
;  C 0973  1          Pattern 9       0000    000000      0 000 000 000 000 000
;  C 0974  1                          0000    000000      0 000 000 000 000 000
;  C 0975  1                          0000    000000      0 000 000 000 000 000
;  C 0976  1                          FFFF    177777      1 111 111 111 111 111
;  C 0977  1                          FFFF    177777      1 111 111 111 111 111
;  C 0978  1                          FFFF    177777      1 111 111 111 111 111
;  C 0979  1                          0000    000000      0 000 000 000 000 000
;  C 0980  1                          0000    000000      0 000 000 000 000 000
;  C 0981  1                          FFFF    177777      1 111 111 111 111 111
;  C 0982  1                          FFFF    177777      1 111 111 111 111 111
;  C 0983  1                          0000    000000      0 000 000 000 000 000
;  C 0984  1                          FFFF    177777      1 111 111 111 111 111
;  C 0985  1                          0000    000000      0 000 000 000 000 000
;  C 0986  1                          FFFF    177777      1 111 111 111 111 111
;  C 0987  1                          0000    000000      0 000 000 000 000 000
;  C 0988  1                          FFFF    177777      1 111 111 111 111 111
;  C 0989  1
;  C 0990  1          Pattern 10      B6D9    133331      1 011 011 011 011 001
;  C 0991  1
;  C 0992  1          Pattern 11      5555    052525      0 101 010 101 010 101
;  C 0993  1                          5555    052525      0 101 010 101 010 101
;  C 0994  1                          5555    052525      0 101 010 101 010 101
;  C 0995  1                          AAAA    125252      1 010 101 010 101 010
;  C 0996  1                          AAAA    125252      1 010 101 010 101 010
;  C 0997  1                          AAAA    125252      1 010 101 010 101 010
;  C 0998  1                          5555    052525      0 101 010 101 010 101
;  C 0999  1                          5555    052525      0 101 010 101 010 101
;  C 1000  1                          AAAA    125252      1 010 101 010 101 010
;  C 1001  1                          AAAA    125252      1 010 101 010 101 010
;  C 1002  1                          5555    052525      0 101 010 101 010 101
;  C 1003  1                          AAAA    125252      1 010 101 010 101 010
;  C 1004  1                          5555    052525      0 101 010 101 010 101
;  C 1005  1                          AAAA    125252      1 010 101 010 101 010
;  C 1006  1                          5555    052525      0 101 010 101 010 101
;  C 1007  1                          AAAA    125252      1 010 101 010 101 010
```

```
; C 1008  1          Pattern 12      2D2D    026455      0 010 110 100 101 101
; C 1009  1                          2D2D    026455      0 010 110 100 101 101
; C 1010  1                          2D2D    026455      0 010 110 100 101 101
; C 1011  1                          D2D2    151322      1 101 001 011 010 010
; C 1012  1                          D2D2    151322      1 101 001 011 010 010
; C 1013  1                          D2D2    151322      1 101 001 011 010 010
; C 1014  1                          2D2D    026455      0 010 110 100 101 101
; C 1015  1                          2D2D    026455      0 010 110 100 101 101
; C 1016  1                          D2D2    151322      1 101 001 011 010 010
; C 1017  1                          D2D2    151322      1 101 001 011 010 010
; C 1018  1                          2D2D    026455      0 010 110 100 101 101
; C 1019  1                          2D2D    026455      0 010 110 100 101 101
; C 1020  1                          D2D2    151322      1 101 001 011 010 010
; C 1021  1                          2D2D    026455      0 010 110 100 101 101
; C 1022  1                          D2D2    151322      1 101 001 011 010 010
; C 1023  1                          2D2D    026455      0 010 110 100 101 101
; C 1024  1                          D2D2    151322      1 101 001 011 010 010
; C 1025  1                          2D2D    026455      0 010 110 100 101 101
; C 1026  1                          D2D2    151322      1 101 001 011 010 010
; C 1027  1                          2D2D    026455      0 010 110 100 101 101
; C 1028  1
; C 1029  1          Pattern 13      6DB6    066666      0 110 110 110 110 110
; C 1030  1
; C 1031  1          Pattern 14      0001    000001      0 000 000 000 000 001
; C 1032  1                          0002    000002      0 000 000 000 000 010
; C 1033  1                          0004    000004      0 000 000 000 000 100
; C 1034  1                          0008    000010      0 000 000 000 001 000
; C 1035  1                          0010    000020      0 000 000 000 010 000
; C 1036  1                          0020    000040      0 000 000 000 100 000
; C 1037  1                          0040    000100      0 000 000 001 000 000
; C 1038  1                          0080    000200      0 000 000 010 000 000
; C 1039  1                          0100    000400      0 000 000 100 000 000
; C 1040  1                          0200    001000      0 000 001 000 000 000
; C 1041  1                          0400    002000      0 000 010 000 000 000
; C 1042  1                          0800    004000      0 000 100 000 000 000
; C 1043  1                          1000    010000      0 001 000 000 000 000
; C 1044  1                          2000    020000      0 010 000 000 000 000
; C 1045  1                          4000    040000      0 100 000 000 000 000
; C 1046  1                          8000    100000      1 000 000 000 000 000
```

```
; C 1047  1          Pattern 15      FFFE    177776      1 111 111 111 111 110
; C 1048  1                          FFFD    177775      1 111 111 111 111 101
; C 1049  1                          FFFB    177773      1 111 111 111 111 011
; C 1050  1                          FFF7    177767      1 111 111 111 110 111
; C 1051  1                          FFEF    177757      1 111 111 111 101 111
; C 1052  1                          FFDF    177737      1 111 111 111 011 111
; C 1053  1                          FFBF    177677      1 111 111 110 111 111
; C 1054  1                          FF7F    177577      1 111 111 101 111 111
; C 1055  1                          FEFF    177377      1 111 111 011 111 111
; C 1056  1                          FDFF    176777      1 111 110 111 111 111
; C 1057  1                          FBFF    175777      1 111 101 111 111 111
; C 1058  1                          F7FF    173777      1 111 011 111 111 111
; C 1059  1                          EFFF    167777      1 110 111 111 111 111
; C 1060  1                          DFFF    157777      1 101 111 111 111 111
; C 1061  1                          BFFF    137777      1 011 111 111 111 111
; C 1062  1                          7FFF    077777      0 111 111 111 111 111
; C 1063  1
; C 1064  1          Pattern 16      B6D9    133331      1 011 011 011 011 001
; C 1065  1                          B6D9    133331      1 011 011 011 011 001
; C 1066  1                          B6D9    133331      1 011 011 011 011 001
; C 1067  1                          DB6C    155554      1 101 101 101 101 100
; C 1068  1                          DB6C    155554      1 101 101 101 101 100
; C 1069  1                          DB6C    155554      1 101 101 101 101 100
; C 1070  1                          B6D9    133331      1 011 011 011 011 001
; C 1071  1                          B6D9    133331      1 011 011 011 011 001
; C 1072  1                          DB6C    155554      1 101 101 101 101 100
; C 1073  1                          DB6C    155554      1 101 101 101 101 100
; C 1074  1                          B6D9    133331      1 011 011 011 011 001
; C 1075  1                          DB6C    155554      1 101 101 101 101 100
; C 1076  1                          B6D9    133331      1 011 011 011 011 001
; C 1077  1                          DB6C    155554      1 101 101 101 101 100
; C 1078  1                          B6D9    133331      1 011 011 011 011 001
; C 1079  1                          DB6C    155554      1 101 101 101 101 100
```

```
;  C 1080  1          Pattern 17      (LBN)*   (LBN)          (LBN)
;  C 1081  1                          8D36     106466   1 000 110 100 110 110
;  C 1082  1                          8D36     106466   1 000 110 100 110 110
;  C 1083  1                          72C9     071311   0 111 001 011 001 001
;  C 1084  1                          72C9     071311   0 111 001 011 001 001
;  C 1085  1                          72C9     071311   0 111 001 011 001 001
;  C 1086  1                          8D36     106466   1 000 110 100 110 110
;  C 1087  1                          8D36     106466   1 000 110 100 110 110
;  C 1088  1                          8D36     106466   1 000 110 100 110 110
;  C 1089  1                          8D36     106466   1 000 110 100 110 110
;  C 1090  1                          72C9     071311   0 111 001 011 001 001
;  C 1091  1                          72C9     071311   0 111 001 011 001 001
;  C 1092  1                          72C9     071311   0 111 001 011 001 001
;  C 1093  1                          72C9     071311   0 111 001 011 001 001
;  C 1094  1                          72C9     071311   0 111 001 011 001 001
;  C 1095  1                          8D36     106466   1 000 110 100 110 110
;  C 1096  1                          8D36     106466   1 000 110 100 110 110
;  C 1097  1                          8D36     106466   1 000 110 100 110 110
;  C 1098  1                          8D36     106466   1 000 110 100 110 110
;  C 1099  1                          8D36     106466   1 000 110 100 110 110
;  C 1100  1                          8D36     106466   1 000 110 100 110 110
;  C 1101  1
;  C 1102  1              * This word position contains the number of the locical
;  C 1103  1                block to be written.
;  C 1104  1
;  C 1105  1
;  C 1106  1
;  C 1107  1          Pattern 18      8D36     106466   1 000 110 100 110 110
;  C 1108  1                          (LBN)    (LBN)          (LBN)
;  C 1109  1                          72C9     071311   0 111 001 011 001 001
;  C 1110  1                          8D36     106466   1 000 110 100 110 110
;  C 1111  1                          8D36     106466   1 000 110 100 110 110
;  C 1112  1                          8D36     106466   1 000 110 100 110 110
;  C 1113  1                          72C9     071311   0 111 001 011 001 001
;  C 1114  1                          72C9     071311   0 111 001 011 001 001
;  C 1115  1                          72C9     071311   0 111 001 011 001 001
;  C 1116  1                          72C9     071311   0 111 001 011 001 001
;  C 1117  1                          8D36     106466   1 000 110 100 110 110
;  C 1118  1                          8D36     106466   1 000 110 100 110 110
;  C 1119  1                          8D36     106466   1 000 110 100 110 110
;  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                          72C9     071311   0 111 001 011 001 001
;  C 1126  1                          72C9     071311   0 111 001 011 001 001
;  C 1127  1                          72C9     071311   0 111 001 011 001 001
```

```
;  C 1128  1          Pattern 19      (LBN)    (LBN)          (LBN)
;  C 1129  1                          B999     134631   1 011 100 110 011 001
;  C 1130  1                          B999     134631   1 011 100 110 011 001
;  C 1131  1                          4666     043146   0 100 011 001 100 110
;  C 1132  1                          4666     043146   0 100 011 001 100 110
;  C 1133  1                          4666     043146   0 100 011 001 100 110
;  C 1134  1                          B999     134631   1 011 100 110 011 001
;  C 1135  1                          B999     134631   1 011 100 110 011 001
;  C 1136  1                          B999     134631   1 011 100 110 011 001
;  C 1137  1                          B999     134631   1 011 100 110 011 001
;  C 1138  1                          4666     043146   0 100 011 001 100 110
;  C 1139  1                          4666     043146   0 100 011 001 100 110
;  C 1140  1                          4666     043146   0 100 011 001 100 110
;  C 1141  1                          4666     043146   0 100 011 001 100 110
;  C 1142  1                          4666     043146   0 100 011 001 100 110
;  C 1143  1                          B999     134631   1 011 100 110 011 001
;  C 1144  1                          B999     134631   1 011 100 110 011 001
;  C 1145  1                          B999     134631   1 011 100 110 011 001
;  C 1146  1                          B999     134631   1 011 100 110 011 001
;  C 1147  1                          B999     134631   1 011 100 110 011 001
;  C 1148  1                          B999     134631   1 011 100 110 011 001
;  C 1149  1
;  C 1150  1          Pattern 20      B999     134631   1 011 100 110 011 001
;  C 1151  1                          (LBN)    (LBN)          (LBN)
;  C 1152  1                          4666     043146   0 100 011 001 100 110
;  C 1153  1                          B999     134631   1 011 100 110 011 001
;  C 1154  1                          B999     134631   1 011 100 110 011 001
;  C 1155  1                          B999     134631   1 011 100 110 011 001
;  C 1156  1                          4666     043146   0 100 011 001 100 110
;  C 1157  1                          4666     043146   0 100 011 001 100 110
;  C 1158  1                          4666     043146   0 100 011 001 100 110
;  C 1159  1                          4666     043146   0 100 011 001 100 110
;  C 1160  1                          B999     134631   1 011 100 110 011 001
;  C 1161  1                          B999     134631   1 011 100 110 011 001
;  C 1162  1                          B999     134631   1 011 100 110 011 001
;  C 1163  1                          B999     134631   1 011 100 110 011 001
;  C 1164  1                          B999     134631   1 011 100 110 011 001
;  C 1165  1                          4666     043146   0 100 011 001 100 110
;  C 1166  1                          4666     043146   0 100 011 001 100 110
;  C 1167  1                          4666     043146   0 100 011 001 100 110
;  C 1168  1                          4666     043146   0 100 011 001 100 110
;  C 1169  1                          4666     043146   0 100 011 001 100 110
;  C 1170  1                          4666     043146   0 100 011 001 100 110
;  C 1171  1
;  C 1172  1          Pattern 21      (LBN)    (LBN)          (LBN)
;  C 1173  1
;    1174  *       )%
```

```
;    1175  1      %sbttl 'PROGRAM HEADER'
;    1176  1
;    1177  1      library 'ZRQAD0.L16';                    ! RDRX EXERCISER GLOBAL LIBRARY
;    1178  1
;    1179  1      require 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;    2670  1
;    2671  1      literal
;    2672  1          DS$NBR_OF_TESTS = 1;                 ! NUMBER OF TESTS IN THIS DIAGNOSTIC
;    2673  1
;    2674  1      EQUALS;
;    2675  1
;    2676  1      POINTER (ALL);
;    2677  1
;    2678  1      !+
;    2679  1      ! THE PROGRAM HEADER IS THE INTERFACE BETWEEN
;    2680  1      ! THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
;    2681  1      !
;    2682  1
;    2683  1      HEADER (%ascii'ZRQA', %ascii'D', %ascii'0', 32767, 1, PRI00);
```

```
;    2684  1     %sbttl 'DISPATCH TABLE'
;    2685  1
;    2686  1     !+
;    2687  1     ! THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST.
;    2688  1     ! IT IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST.
;    2689  1     !-
;    2690  1
;    2691  1     DISPATCH (DS$NBR_OF_TESTS);
```

```
;    2692  1     %sbttl 'GLOBAL DATA SECTION'
;    2693  1
;    2694  1     !+
;    2695  1     ! THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
;    2696  1     ! IN MORE THAN ONE TEST.
;    2697  1     !-
;    2698  1
;    2699  1     psect
;    2700  1         global = $FFF$ (read, write, noexecute, global, concatenate);
;    2701  1
;    2702  1     global
;    2703  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    2704  1                           ! RUN-TIME CONTROLLER STATUS TABLES
;    2705  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    2706  1                           ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    2707  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    2708  1                           ! DRIVER CONTROLLER TABLES
;    2709  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    2710  1                           ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    2711  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    2712  1                           ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    2713  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    2714  1                           ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    2715  1
;    2716  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
;    2717  1                           !CONTAINS LO+ HI LBN FIELDS FOR SEQUENTIAL     !ZZZ
;    2718  1                           !I/O TRANSFER FOR EACH UNIT.                   !ZZZ
;    2719  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    2720  1                           ! STATISTICS TABLES
;    2721  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    2722  1                           ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    2723  1
;    2724  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),      !BUFFER FOR DUP  ZZZ
;    2725  1                           !INFO FROM RECEIVE + SEND CMDS                   ZZZ
;    2726  1          RK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED] INITIAL (BYTE (REP     !ZZZ
;    2727  1                MAX_UNITS OF (1))),          !CURRENT TRACK DIRECTION      ZZZ
;    2728  1         RDM_CNT : WORD INITIAL (RDM_LEN),   !NO OF RANDOM NOS   \\KEEP    ZZZ
;    2729  1         RANDOM : VECTOR [RDM_LEN, WORD],    !RANDOM NO. TABLE   //TOGETHER ZZZ
;    2730  1
;    2731  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    2732  1                           ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    2733  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
;    2734  1                           ! MSCP PACKET POOL
;    2735  1         IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    2736  1                           ! ADDRESS OF AN MSCP PACKET (INTERUPT PROCESSING)
;    2737  1         PKT_USE : vector [PKT_CNT, byte, signed],
;    2738  1                           ! MSCP PACKET POOL ALLOCATION TABLE
;    2739  1         RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;    2740  1                           ! RETURN PACKET POOL
;    2741  1         RP_USE : vector [RP_CNT, byte, signed],
;    2742  1                           ! RETURN PACKET POOL ALLOCATION TABLE
;    2743  1         RP_INDX : word,     ! CURRENT RETURN PACKET INDEX
;    2744  1         RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
```

```
;    2745  1                          ! CURRENT RETURN PACKET ADDRESS
;    2746  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;    2747  1                          ! ERROR-LOG PACKET SAVE AREA
;    2748  1          BUFF_ADDR : vector [MAX_BUF_CNT],                  ! TABLE OF I/O BUFFER DESCRIPTORS
;    2749  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],     ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;    2750  1          IODQ : vector [IODQ_LEN, byte],                    ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;    2751  1          IODQ_IN : word,                                    ! I/O DONE QUEUE IN POINTER
;    2752  1          IODQ_OUT : word,                                   ! I/O DONE QUEUE OUT POINTER
;    2753  1          ENTRY_REASON : byte,                               ! CURRENT OPERATOR COMMAND
;    2754  1          EOP_FLAG : byte,                                   ! END-OF-PASS FLAG
;    2755  1          DUP_FLAGS : WORD,                                  !DUP FLAGS       ZZZ
;    2756  1          CCTLR : word,                                      ! NUMBER OF "CURRENT" CONTROLLER
;    2757  1          CDISK : word,                                      ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2758  1          CUOFF : word,                                      ! CURRENT UNIT CST OFFSET
;    2759  1          CTLR_CNT : word,                                   ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;    2760  1          DUR : vector [MAX_UNITS, byte],                    ! DROP UNIT REASON
;    2761  1          QIO : vector [MAX_CTLR, byte],                     ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;    2762  1          FREE_MEM_ADDR,                                     ! START OF FREE MEMORY
;    2763  1          BYTS_PER_QIO : word,                               ! SIZE (BYTES) OF AN I/O BUFFER
;    2764  1          ST_CODE : word,                                    ! CURRENT STATUS CODE
;    2765  1          SB_CODE : word,                                    ! CURRENT SUB-CODE
;    2766  1          STEP : word,                                       ! CURRENT STEP IN HARD_INIT
;    2767  1          OF_RC : signed word,                               ! OFFSET (0 OR 2) TO READ IP OR SA
;    2768  1          SA_REG : word,                                     ! STORAGE FOR SA REGISTER READS AND WRITES
;    2769  1          CMD_TIME : word,                                   ! COMMAND TIMEOUT VALUE (IN SECONDS)
;    2770  1          NEX : word,                                        ! NON-EXISTENT MEMORY TRAP INDICATOR
;    2771  1          CRN_LOW : word,                                    ! COMMAND REF NUMBER OF LAST COMMAND SENT
;    2772  1          CRN_HIGH : word,                                   ! COMMAND REF NUMBER (HI ORDER)
;    2773  1          CREDIT_BAL : word,                                 ! CREDIT BALANCE
;    2774  1          NEXT_PKT_USE : byte,                               ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;    2775  1          HOURS : byte,                                      ! TIME OF DAY (HOURS)
;    2776  1          MINUTES : byte,                                    ! TIME OF DAY (MINUTES)
;    2777  1          CLK_TICKS : word,                                  ! TIME OF DAY (LINE-CLOCK TICKS)
;    2778  1          CLK_PRESENT : byte,                                ! FLAG INDICATES IF LINE-CLOCK PRESENT
;    2779  1          HOE_FLAG : byte,                                   ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;    2780  1
;    2781  1          S_PATTERN : WORD,                 !PATTERN FOR DUP WRITES          ZZZ
;    2782  1          S_DUPPKT : WORD,                  !DBN BYTE COUNTER                ZZZ
;    2783  1          P_INDEX : SIGNED WORD,            !CURRENT MESSAGE PACKET INDEX    ZZZ
;    2784  1          FORCED_ERROR : byte,                                ! "FORCED ERROR" DETECTED IN LAST READ
;    2785  1          FER_LBN : word,                                     ! LBN OF THE "FORCED ERROR" BLOCK
;    2786  1          FER_BC : word,                                      ! BYTE COUNT OF THE "FORCED ERROR' BLOCK
;    2787  1          INIT_OCCURED : byte initial (byte (FALSE)),         ! EXERCISER INITIALIZATION COMPLETE
;    2788  1          ADDR_VECT_OK : byte initial (byte (FALSE));         ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    2789  1
;    2790  1      ERRTBL;
```

```
;    2791  1     %sbttl 'GLOBAL TEXT SECTION'
;    2792  1
;    2793  1     !+
;    2794  1     ! THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
;    2795  1     ! MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
;    2796  1     ! MORE THAN ONE TEST.
;    2797  1     !-
;    2798  1
;    2799  1     global bind
;    2800  1     !
;    2801  1     ! HARDWARE DIALOG
;    2802  1     !
;    2803  1         HWQ1  = uplit (%asciz'IP address'),
;    2804  1         HWQ2  = uplit (%asciz'Vector'),
;    2805  1         HWQ3  = uplit (%asciz'BR level'),
;    2806  1         HWQ4  = uplit (%asciz'Drive number'),                              !ZZZ
;    2807  1         HWQ5  = uplit (%asciz'Test entire customer area of this disk'),    !ZZZ
;    2808  1         HWQ6A = uplit (%asciz'Lower octal word of beginning LBN address'), !ZZZ
;    2809  1         HWQ6B = uplit (%asciz'Higher octal word of beginning LBN address'),!ZZZ
;    2810  1         HWQ7A = uplit (%asciz'Lower octal word of ending LBN address'),    !ZZZ
;    2811  1         HWQ7B = uplit (%asciz'Higher octal word of ending LBN address'),   !ZZZ
;    2812  1         HWQ8  = uplit (%asciz'Write on customer data area on this disk'),
;    2813  1         HWQ9  = uplit (%asciz'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
;    2814  1         HWQ10 = uplit (%asciz'Also run DUP exerciser'),                    !ZZZ
;    2815  1         HWQ11 = uplit (%asciz'Write on diagnostic area'),                  !ZZZ
;    2816  1     !
;    2817  1     ! SOFTWARE DIALOG
;    2818  1     !
;    2819  1         SWQ1  = uplit (%asciz'Hard Error limit'),
;    2820  1         SWQ2  = uplit (%asciz'Transfer limit in megabytes (0 for "Quick pass")'),
;    2821  1         SWQ4  = uplit (%asciz'Random seek mode'),
;    2822  1         SWQ7  = uplit (%asciz'Read-compares performed at the Controller'),
;    2823  1         SWQ9  = uplit (%asciz'Write-compares performed at the Controller'),
;    2824  1         SWQ10 = uplit (%asciz'Check all Writes at host by reading'),
;    2825  1         SWQ11 = uplit (%asciz'User-defined data pattern'),
;    2826  1         SWQ12 = uplit (%asciz'Select pre-defined data pattern (0 for Sequential selection)'),
;    2827  1         SWQ13 = uplit (%asciz'Number of words in data pattern (16 maximum)'),
;    2828  1         SWQ14 = uplit (%asciz'Pattern value (No leading zeros allowed)'),
;    2829  1         SWQ15 = uplit (%asciz'Clear statistical tables after printing'),
;    2830  1         SWQ17 = uplit (%asciz'Percentage of "Fixed Disk" operations out of total operations'),
;    2831  1         SWQ19 = uplit (%asciz'Units to be selected at random (No, implies sequential)'),
;    2832  1         SWQ20 = uplit (%asciz'Want to rewrite blocks when "Forced Error" detected on reads'),
;    2833  1         SWQ21 = uplit (%asciz'Do you want to halt on other Hard Errors (@s 31-34, 36 37, 39-45)'),
;    2834  1         SWQ22 = uplit (%asciz'Do you want to halt on Soft Errors (@s 50-54)'),
;    2835  1         SWQ23 = uplit (%asciz'Do you want to halt on Bad-block Hard Errors (@s 35, 38)'),
;    2836  1         SWQ24 = uplit (%asciz'Enter time as HHMM (Example: 1305    No leading zeros allowd) '),
;    2837  1         SWQ25 = uplit (%asciz'Count each retry on a Read/Write error as a seperate Soft Error'),
;    2838  1         SWM1  = uplit (%asciz'The remaining questions only apply to unprotected disks'),
;    2839  1         NULL  = uplit (%asciz''),
;    2840  1
;    2841  1     !++
;    2842  1     !  THE FOLLOWING DBMs ARE DEBUG MESSAGES, AND SHOULD BE REMOVED BEFORE
;    2843  1     !  RELEASING THE PROGRAM.  THEY INCLUDE THE NAMES OF EACH ROUTINE, PLUS
```

```
;    2844  1      !  FORMAT STATEMENTS FOR PRINTING OUT OTHER INFORMATION.
;    2845  1      !
;    2846  1
;    2847  1          DBM5    = uplit (%asciz'%N%A** Drop unit %D2'),
;    2848  1          DBM12   = uplit (%asciz'%N%A** PROC_RETPKT: Conn ID %O6%A received'),
;    2849  1          DBM15   = uplit (%asciz'%N%A** Multi-drive test'),
;    2850  1          DBM18   = uplit (%asciz'%N%A** FATAL_ERROR: RETPKT not available'),
;    2851  1          DBM19   = uplit (%asciz'%N%A** FSET_UPAR: Can''t find disk %D3%A in CST %D1'),
;    2852  1          DBM20   = uplit (%asciz'%N%A** Bad conn ID %O6%A received from %O6'),
;    2853  1          DBM21   = uplit (%asciz'%N%A** Message type %O2%A received in MSCP packet'),
;    2854  1          DBM22   = uplit (%asciz'%N%A** SEQUEN: RETPKT not available'),
;    2855  1          DBM23   = uplit (%asciz'%N%A** Error in SET_CTLR_CHAR'),
;    2856  1          DBM25   = uplit (%asciz'%N%A** Ctlr timeout = %D3%A. seconds'),
;    2857  1          DBM26   = uplit (%asciz'%N%A** Error in UNIT_INIT'),
;    2858  1          DBM27   = uplit (%asciz'%N%A** UNIT_INIT: RETPKT has bad ENDCODE'),
;    2859  1          DBM28A  = uplit (%asciz'%N%A** Unit size (Lo) = %D5%A,'),
;    2860  1          DBM28B  = uplit (%asciz'%N%A** Unit size (Hi) = %D5%A,'),
;    2861  1          DBM29   = uplit (%asciz'%N%A** ACCESS: RETPKT has bad ENDCODE'),
;    2862  1          DBM32   = uplit (%asciz'%N%A** QIO_UNIT: CST %D1%A no unit selected'),
;    2863  1          DBM101  = uplit (%asciz'%N%A** Unit # is: %O6'),
;    2864  1          DBM104  = uplit (%asciz'%N%A** Removable disk is selected'),
;    2865  1          DBM105  = uplit (%asciz'%N%A** Fixed disk is selected'),
;    2866  1          DBM107  = uplit (%asciz'%N%A** Illegal function: %O6'),
;    2867  1          DBM108  = uplit (%asciz'%N%A** Command ref # %O6%A/%O6%A (Oct) not sent by Host'),
;    2868  1          DBM109  = uplit (%asciz'%N%A** Unknown Error Log format %O3%A received'),
;    2869  1      !   DBM110  = uplit (%asciz'%N%A** Error-Log save area full'),
;    2870  1          DBM111  = uplit (%asciz'%N%A** Op-code %O3%A, End-code %O3%A for ref # %O6%A/%O6%A (8)'),
;    2871  1          DBM112  = uplit (%asciz'%N%A** Cmd-bc %O6%A/%O6%A Rsp-bc %O6%A/%O6%A for %O6%A/%O6%A (8)'),
;    2872  1          DBM120  = uplit (%asciz'%N%A** Response already received for cmd %O6%A/%O6%A (8)'),
;    2873  1          DBM121  = uplit (%asciz'%N%A** Failure to send command after # %O6%A/%O6%A (8)'),
;    2874  1      !
;    2875  1      ! DROP UNIT MESSAGES
;    2876  1      !
;    2877  1          DU_MSG = uplit (%asciz'%N%AUNIT%D2%A DROPPED - '),
;    2878  1          DU_RSN = uplit (
;    2879  1                   uplit (%asciz'%AUSER COMMAND%N'),
;    2880  1                   uplit (%asciz'%ACONFIGURATION ERROR%N'),
;    2881  1                   uplit (%asciz'%AINIT ERROR%N'),
;    2882  1                   uplit (%asciz'%ATRANSFER LIMIT REACHED%N'),
;    2883  1                   uplit (%asciz'%AERROR LIMIT REACHED%N'),
;    2884  1                   uplit (%asciz'%AUNRECOVERABLE DRIVE ERROR%N'),
;    2885  1                   uplit (%asciz'%AUNRECOVERABLE CONTROLLER ERROR%N'),
;    2886  1                   uplit (%asciz'%AFAILED TO COME ONLINE%N'),
;    2887  1                   uplit (%asciz'%AFAILED TO ACCESS EITHER FIRST OR LAST TRACK DURING INIT%N'),
;    2888  1                   uplit (%asciz'%ADISK WRITE PROTECTED%N'),
;    2889  1                   uplit (%asciz'%ACOMMAND TIME OUT%N')) : vector [11],
;    2890  1      !
;    2891  1      ! SYSTEM MESSAGES (PRINTF)
;    2892  1      !
;    2893  1          MSG_01  = uplit (%asciz'%N%APOWER DELAY - WAITING'),
;    2894  1          MSG_02  = uplit (%asciz'%N%AFUNCTIONAL TEST STARTED'),
;    2895  1          MSG_03  = uplit (%asciz'%N%N%AEXERCISER STARTED%N'),
;    2896  1      !
```

```
;    2897  1      ! REPORT MESSAGES (PRINTS)
;    2898  1      !
;    2899  1          RPT1  = uplit (%asciz'%N%N%AUNT DSK%S8%A# OF    # BYTES    # OF     # BYTES'),
;    2900  1          RPT2  = uplit (%asciz'%A  --HARD ERRORS--  --SOFT ERRORS--'),
;    2901  1          RPT3  = uplit (%asciz'%N%A #    ^  TYPE  READS     READ   WRITES   WRITTEN'),
;    2902  1          RPT4  = uplit (%asciz'%A  SEK DAT DRV HST SEK DAT DRV HST'),
;    2903  1          RPT5  = uplit (%asciz'%N%A--- --- ---- ----- --------- ------ ---------'),
;    2904  1          RPT6  = uplit (%asciz'%A --- --- --- --- --- --- --- ---'),
;    2905  1          RPT7  = uplit (%asciz'%N%D2%D4%S2%T'),
;    2906  1          RPT8  = uplit (%asciz'%D4%Z3%D3%A,%Z3%A,%Z3'),
;    2907  1          RPT9  = uplit (%asciz'%D4%D4%D4%D4%D4%D4%D4%D4%D4'),
;    2908  1          RPT10 = uplit (%asciz'%N%A .   .  CNTR     .  .........    .  .........'),
;    2909  1          RPT11 = uplit (%asciz'%A   .   .%D4%A   .   .   .%D4%A   .'),
;    2910  1      !
;    2911  1      ! GENERAL ERROR MESSAGES
;    2912  1      !
;    2913  1      !       SYSTEM FATAL (ERRSF)
;    2914  1      !
;    2915  1          EGS_01 = uplit (%asciz'TOO MANY UNITS'),
;    2916  1          EGS_02 = uplit (%asciz'NOT ENOUGH FREE MEMORY FOR ALLOCATING READ/WRITE BUFFERS'),
;    2917  1      !
;    2918  1      !       DRIVE FATAL (ERRDF)
;    2919  1      !
;    2920  1          EGD_10 = uplit (%asciz'REGISTER EXISTENCE TEST FAILED'),
;    2921  1          EGD_11 = uplit (%asciz'VECTOR TEST FAILED'),
;    2922  1          EGD_12 = uplit (%asciz'BR LEVEL TEST FAILED'),
;    2923  1          EGD_13 = uplit (%asciz'INIT SEQUENCE FAILED'),
;    2924  1          EGD_14 = uplit (%asciz'FATAL CONTROLLER ERROR'),
;    2925  1          EGD_15 = uplit (%asciz'ONLINE FAILED'),
;    2926  1          EGD_16 = uplit (%asciz'WRITE-PROTECT CONFLICT'),
;    2927  1          EGD_17 = uplit (%asciz'ACCESS FAILED'),
;    2928  1          EGD_18 = uplit (%asciz'FATAL I/O ERROR'),
;    2929  1      !   EGD_19 = uplit (%asciz'CONTROLLER TIMEOUT'),
;    2930  1          EGD_19 = uplit (%asciz'DISK TYPE UNKNOWN TO EXERCISER')
;    2931  1          EGD_20 = uplit (%asciz'FAILED TO SEND SET-CONTROLLER-CHARACTERISTICS COMMAND'),
;    2932  1          EGD_21 = uplit (%asciz'SET-CONTROLLER-CHARACTERISTICS RESPONSE HAS BAD ENDCODE OR FLAGS IN ERROR'),
;    2933  1          EGD_22 = uplit (%asciz'FAILED TO SEND ON LINE COMMAND'),
;    2934  1          EGD_23 = uplit (%asciz'ON-LINE RESPONSE HAS BAD ENDCODE'),
;    2935  1          EGD_24 = uplit (%asciz'ON-LINE RESPONSE HAS UNKNOWN DEVICE'),
;    2936  1      !
;    2937  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    2938  1      !
;    2939  1          EGH_30 = uplit (%asciz'I/O REQUEST FAILED'),
;    2940  1      !
;    2941  1      ! BASIC ERROR MESSAGES (PRINTB)
;    2942  1      !
;    2943  1      !       SYSTEM FATAL (ERRSF)
;    2944  1      !
;    2945  1          EBS_01 = uplit (%asciz'%MORE THAN %D2%A UNITS SPECIFIED'),
;    2946  1      !
;    2947  1      !       DRIVE FATAL (ERRDF)
;    2948  1      !
;    2949  1          EBD_10 = uplit (%asciz'%A* NO RESPONSE AT ADDRESS %O6'),
```

```
;    2950  1          EBD_12 = uplit (%asciz'%A* INCORRECT BR LEVEL FOR DRIVE %O6'),
;    2951  1          EBD_13 = uplit (%asciz'%A* STEP %D1%A READ ERROR'),
;    2952  1          EBD_14 = uplit (%asciz'%A* BAD SA CODE FROM DRIVE %O6'),
;    2953  1          EBD_18 = uplit (%asciz'%A* DISK%D2%A WENT OFFLINE'),
;    2954  1          EBD_19 = uplit (%asciz'%A* DRIVE %O6%A NOT PROCESSING COMMAND PACKETS'),
;    2955  1          EBD_24 = uplit (%asciz'%A* DISK%D2%A WENT TO THE "AVAILABLE" STATE'),
;    2956  1      !
;    2957  1      !
;    2958  1      !       HARD or SOFT (ERRHRD or ERRSOFT)
;    2959  1      !
;    2960  1      EH_0 = UPLIT (%ASCIZ' - UNRECOGNIZED MESSAGE TYPE'),                    !ZZZ
;    2961  1      EH_1 = UPLIT (%ASCIZ' - UNRECOGNIZED CONNECTION ID'),                   !ZZZ
;    2962  1      EH_2 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN MESSAGE'),                  !ZZZ
;    2963  1      EH_3 = UPLIT (%ASCIZ' - UNRECOGNIZED RETURN PACKET'),                   !ZZZ
;    2964  1      EH_4 = UPLIT (%ASCIZ' - UNRECOGNIZED CRN'),                             !ZZZ
;    2965  1      EH_5 = UPLIT (%ASCIZ' - UNRECOGNIZED OPCODE'),                          !ZZZ
;    2966  1      EH_6 = UPLIT (%ASCIZ' - MSCP STATUS CODE ERR'),                         !ZZZ
;    2967  1      EH_7 = UPLIT (%ASCIZ' - DUP STATUS CODE ERR'),                          !ZZZ
;    2968  1      EH_8 = UPLIT (%ASCIZ' - UNRECOGNIZED STATUS CODE'),                     !ZZZ
;    2969  1      EH_9 = UPLIT (%ASCIZ' - LBN HOST COMPARE ERR'),                         !ZZZ
;    2970  1      EH_10 = UPLIT (%ASCIZ' - DBN HOST COMPARE ERR'),                        !ZZZ
;    2971  1      EH_12 = UPLIT (%ASCIZ' - UNABLE TO LOAD DUP MEDIA'),                    !ZZZ
;    2972  1      EH_13 = UPLIT (%ASCIZ' - ERR IN DUP PKT WHEN USING CTLR LC PROG'),      !ZZZ
;    2973  1      !
;    2974  1          ERR_00  = uplit (%asciz'%A* DISK%D2'),
;    2975  1          ERR_COD = uplit (
;    2976  1                    uplit (%asciz'%AINVALID COMMAND'),
;    2977  1                    uplit (%asciz'%ACOMMAND ABORTED'),
;    2978  1                    uplit (%asciz'%AUNIT OFFLINE'),
;    2979  1                    uplit (%asciz'%ATRANSITION TO AVAILABLE STATE'),
;    2980  1                    uplit (%asciz'%AMEDIA FORMAT ERROR'),
;    2981  1                    uplit (%asciz'%AWRITE-PROTECTED'),
;    2982  1                    uplit (%asciz'%ADEVICE COMPARE ERROR'),
;    2983  1                    uplit (%asciz'%ADATA ERROR'),
;    2984  1                    uplit (%asciz'%AHOST BUFFER ACCESS ERROR'),
;    2985  1                    uplit (%asciz'%ACONTROLLER ERROR'),
;    2986  1                    uplit (%asciz'%ADRIVE ERROR'),
;    2987  1                    uplit (%asciz'%AMESSAGE FROM INTERNAL DIAGNOSTICS'),
;    2988  1                    uplit (%asciz'%AHOST COMPARE ERROR '),
;    2989  1                    uplit (%asciz'%ACOMMAND TIMEOUT')) : vector [14],
;    2990  1      !
;    2991  1      !       ERROR LOG MESSAGE (ERRSOFT)
;    2992  1      !
;    2993  1          ELG_00  = uplit (%asciz'%AERROR LOG MESSAGE RECEIVED:%N'),
;    2994  1          ELG_FMT = uplit (
;    2995  1                    uplit (%asciz'%A* CONTROLLER ERROR%N'),
;    2996  1                    uplit (%asciz'%A* HOST MEMORY ACCESS ERROR%N'),
;    2997  1                    uplit (%asciz'%A* DISK%D2%A - DISK TRANSFER ERROR%N'),
;    2998  1                    uplit (%asciz'%A* DISK%D2%A - "STANDARD DISK INTERCONNECT" ERROR%N'),
;    2999  1                    uplit (%asciz'%A* DISK%D2%A - "SMALL DISK" ERROR%N')) : vector [5],
;    3000  1      !
;    3001  1      ! EXTENDED ERROR MESSAGES (PRINTX)
;    3002  1      !
```

```
;    3003  1          EX_SA  = uplit (%asciz'%N%A* SA: %O6'),
;    3004  1          EX_SC  = uplit (%asciz'%N%A* STATUS CODE: %O2'),
;    3005  1          EX_SBO = uplit (%asciz'%O4'),
;    3006  1          EX_SB  = uplit (%asciz'%N%A* SUB_CODE: '),
;    3007  1          EX_CMD = uplit (%asciz'%N%A* COMMAND: '),
;    3008  1          EX_RD  = uplit (%asciz'%AREAD'),
;    3009  1          EX_WRT = uplit (%asciz'%AWRITE'),
;    3010  1          EX_CMP = uplit (%asciz'%A-COMPARE'),
;    3011  1          EX_ONL = uplit (%asciz'%AONLINE'),
;    3012  1          EX_ACC = uplit (%asciz'%AACCESS'),
;    3013  1          EX_OP  = uplit (%asciz'%O3'),
;    3014  1          EX_BB  = uplit (%asciz'%N%A* BAD BLOCK (Host replaceable): %D5%A, (OCT %O6%A)'),
;    3015  1          EX_BB1 = uplit (%asciz'%N%A* 1st BAD BLOCK (Host replaceable): %D5%A, (OCT %O6%A)'),
;    3016  1          EX_BBU = uplit (%asciz'%N%A* BAD BLOCK REPORTED (Replaced): %D%A, (OCT %O6%A)'),
;    3017  1          EX_LBN = uplit (%asciz'%N%A* LBN: %D5%A, (OCT %O6%A)'),
;    3018  1          EX_PBN = uplit (%asciz'%N%A* PBN: %D5%A, (OCT %O6%A)'),
;    3019  1          EX_LBR = uplit (%asciz'%N%A* LBN: (READ) %D5%A, (OCT %O6%A)'),
;    3020  1          EX_LBW = uplit (%asciz'%N%A* LBN: (WRITE) %D5%A, (OCT %O6%A)'),
;    3021  1          EX_RBN = uplit (%asciz'%N%A* REPLACEMENT BLOCK NO. %D5%A, (OCT %O6%A)'),
;    3022  1          EX_CBC = uplit (%asciz'%N%A* BYTE COUNT IN COMMAND: %D5%A.'),
;    3023  1          EX_CBR = uplit (%asciz'%N%A* BYTE COUNT IN READ COMMAND: %D5%A.'),
;    3024  1          EX_CBW = uplit (%asciz'%N%A* BYTE COUNT IN WRITE COMMAND: %D5%A.'),
;    3025  1          EX_BC  = uplit (%asciz'%N%A* ACTUAL # OF BYTES TRANSFERRED: %D5%A.'),
;    3026  1          EX_BD  = uplit (%asciz'%N%A* I/O BUFFER ADDRESS (32 bits): %O6%A %O6'),
;    3027  1          EX_BDR = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR READ (32 bits): %O6%A %O6'),
;    3028  1          EX_BDW = uplit (%asciz'%N%A* I/O BUFFER ADDRESS FOR WRITE (32 bits): %O6%A %O6'),
;    3029  1          EX_RP  = uplit (%asciz'%N%ACONTENTS OF COMMAND/RESPONSE PACKET SAVE AREA:%N'),
;    3030  1          EX_WRD = uplit (%asciz'%A %O6'),
;    3031  1          EX_TIM = uplit (%asciz'%N%A* TIME: %Z2%A:%Z2%A HOURS%N'),
;    3032  1
;    3033  1
;    3034  1          XX13 = UPLIT (%ASCIZ'%N%A * DISK : %D2'),                          !ZZZ
;    3035  1          XX23 = UPLIT (%ASCIZ'%N%ADBN: %D5%A, (OCT %O6%A)'),                !ZZZ
;    3036  1          XX32 = UPLIT (%ASCIZ'%N%ABYTE NUMBER: %D3'),                       !ZZZ
;    3037  1          XX33 = UPLIT (%ASCIZ'%N%ARANDOM WRITTEN WORD :%B16'),              !ZZZ
;    3038  1          XX34 = UPLIT (%ASCIZ'%N%ARANDOM READ WORD bin:%B16%A oct:%O6'),    !ZZZ
;    3039  1
;    3040  1      !
;    3041  1      ! CONFIGURATION ERROR MESSAGES (PRINTF)
;    3042  1      !
;    3043  1          CER_01 = uplit (%asciz'%N%ADUPLICATE UNIT:%D,%A AT IP: %O6'),
;    3044  1          CER_02 = uplit (%asciz'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSES'),
;    3045  1      !
;    3046  1      ! ERROR/EVENT SUB CODES (PRINTX)
;    3047  1      !
;    3048  1          SC_SDI = uplit (%asciz'%ASPIN-DOWN IGNORED'),
;    3049  1          SC_CON = uplit (%asciz'%ASTILL CONNECTED'),
;    3050  1          SC_DUP = uplit (%asciz'%ADUPLICATE UNIT NUMBER'),
;    3051  1          SC_ONL = uplit (%asciz'%AALREADY ONLINE'),
;    3052  1          SC_SON = uplit (%asciz'%ASTILL ONLINE'),
;    3053  1          SC_UNK = uplit (%asciz'%AUNIT UNKNOWN OR ONLINE TO ANOTHER CONTROLLER'),
;    3054  1          SC_VOL = uplit (%asciz'%ANO VOLUME MOUNTED OR DRIVE DISABLED BY SWITCH'),
;    3055  1          SC_IOP = uplit (%asciz'%AUNIT INOPERATIVE (RD51/52 write fault)'),
```

```
;    3056  1          SC_DIS = uplit (%asciz'%AUNIT DISABLED BY FIELD SERVICE OR INTERNAL DIAGNOSTICS'),
;    3057  1          SC_FER = uplit (%asciz'%A"FORCED ERROR" DETECTED WHILE ACCESSING FCT OR RCT'),
;    3058  1          SC_FE2 = uplit (%asciz'%ASECTOR HAD BEEN WRITTEN WITH "FORCED ERROR" MODIFIER'),
;    3059  1          SC_ISH = uplit (%asciz'%AFCT OR RCT UNREADABLE - INVALID SECTOR HEADER'),
;    3060  1          SC_HS2 = uplit (%asciz'%AHEADER COMPARE ERROR (Valid header not found)'),
;    3061  1          SC_DST = uplit (%asciz'%AFCT OR RCT UNREADABLE - DATA SYNC TIMEOUT'),
;    3062  1          SC_DS2 = uplit (%asciz'%ADATA SYNC NOT FOUND (Data sync timeout)'),
;    3063  1          SC_ECC = uplit (%asciz'%AFCT OR RCT UNREADABLE - UNCORRECTABLE ECC ERROR'),
;    3064  1          SC_ECD = uplit (%asciz'%AUNCORRECTABLE ECC ERROR'),
;    3065  1          SC_RCT = uplit (%asciz'%ARCT CORRUPTED'),
;    3066  1          SC_FUL = uplit (%asciz'%ANO REPLACEMENT BLOCK AVAILABLE (RCT full)'),
;    3067  1          SC_576 = uplit (%asciz'%ADISK NOT FORMATTED WITH 512 BYTE SECTORS'),
;    3068  1          SC_FCT = uplit (%asciz'%ADISK NOT FORMATTED OR FCT CORRUPTED'),
;    3069  1          SC_EC1 = uplit (%asciz'%AONE SYMBOL ECC ERROR'),
;    3070  1          SC_EC2 = uplit (%asciz'%ATWO SYMBOL ECC ERROR'),
;    3071  1          SC_EC3 = uplit (%asciz'%ATHREE SYMBOL ECC ERROR'),
;    3072  1          SC_EC4 = uplit (%asciz'%AFOUR SYMBOL ECC ERROR'),
;    3073  1          SC_EC5 = uplit (%asciz'%AFIVE SYMBOL ECC ERROR'),
;    3074  1          SC_EC6 = uplit (%asciz'%ASIX SYMBOL ECC ERROR'),
;    3075  1          SC_EC7 = uplit (%asciz'%ASEVEN SYMBOL ECC ERROR'),
;    3076  1          SC_EC8 = uplit (%asciz'%AEIGHT SYMBOL ECC ERROR'),
;    3077  1          SC_EC9 = uplit (%asciz'%ACORRECTABLE ERROR IN ECC FIELD'),
;    3078  1          SC_SWP = uplit (%asciz'%AUNIT SOFTWARE WRITE PROTECTED'),
;    3079  1          SC_HWP = uplit (%asciz'%AUNIT HARDWARE WRITE PROTECTED'),
;    3080  1          SC_ODA = uplit (%asciz'%AODD TRANSFER ADDRESS'),
;    3081  1          SC_ODB = uplit (%asciz'%AODD BYTE COUNT'),
;    3082  1          SC_NXM = uplit (%asciz'%ANON-EXISTENT HOST MEMORY'),
;    3083  1          SC_PAR = uplit (%asciz'%AHOST MEMORY PARITY ERROR'),
;    3084  1          SC_CTO = uplit (%asciz'%ACOMMAND TIMOUT OR RETRY LIMIT EXCEEDED'),
;    3085  1          SC_SDS = uplit (%asciz'%ASERIALIZER/DESERIALIZER OVERRUN OR UNDERRUN'),
;    3086  1          SC_EDC = uplit (%asciz'%A"ERROR DETECTION CODE" ERROR'),
;    3087  1          SC_IDS = uplit (%asciz'%AINCONSISTENT INTERNAL DATA STRUCTURE'),
;    3088  1          SC_SRT = uplit (%asciz'%ADRIVE COMMAND TIMEOUT (No response or seek incomplete)'),
;    3089  1          SC_SRI = uplit (%asciz'%ACONTROLLER DETECTED TRANSMISSION OR PROTOCOL ERROR'),
;    3090  1          SC_POE = uplit (%asciz'%APOSITION ERROR (Mis-seek)'),
;    3091  1          SC_RDY = uplit (%asciz'%ALOST READ/WRITE READY DURING/BETWEEN TRANSFERS'),
;    3092  1          SC_CLK = uplit (%asciz'%ADRIVE CLOCK DROPOUT'),
;    3093  1          SC_RSP = uplit (%asciz'%ALOST RECEIVER READY BETWEEN SECTORS'),
;    3094  1          SC_SUR = uplit (%asciz'%ADRIVE DETECTED ERROR'),
;    3095  1          SC_PSP = uplit (%asciz'%ACONTROLLER DETECTED PULSE OR STATE PARITY ERROR'),
;    3096  1      !
;    3097  1      ! CONTROLLER GENERIC ERROR CODES
;    3098  1      !
;    3099  1          CNTR_ERR = uplit (
;    3100  1                     uplit (%asciz'%ACONTROLLER TIMEOUT'),
;    3101  1                     uplit (%asciz'%AENVELOPE/PACKET READ ERROR (Parity or timeout)'),
;    3102  1                     uplit (%asciz'%AENVELOPE/PACKET WRITE ERROR (Parity or timeout)'),
;    3103  1                     uplit (%asciz'%ACONTROLLER ROM AND RAM PARITY ERROR'),
;    3104  1                     uplit (%asciz'%ACONTROLLER RAM PARITY ERROR'),
;    3105  1                     uplit (%asciz'%ACONTROLLER ROM PARITY ERROR'),
;    3106  1                     uplit (%asciz'%ARING READ ERROR (Parity or timeout)'),
;    3107  1                     uplit (%asciz'%ARING WRITE ERROR (Parity or timeout)'),
;    3108  1                     uplit (%asciz'INTERRUPT MASTER FAILURE'),
```

```
;    3109  1                    uplit (%asciz'%AHOST ACCESS TIMEOUT (Higher level protocol dependent)'),
;    3110  1                    uplit (%asciz'%ACREDIT LIMIT EXCEEDED'),
;    3111  1                    uplit (%asciz'%AQ-BUS MASTER ERROR'),
;    3112  1                    uplit (%asciz'%ACONTROLLER FATAL ERROR'),
;    3113  1                    uplit (%asciz'%AINSTRUCTION LOOP TIMEOUT'),
;    3114  1                    uplit (%asciz'%AILLEGAL VIRTUAL CIRCUIT ID'),
;    3115  1                    uplit (%asciz'%AINTERRUPT VECTOR ILLEGAL'),
;    3116  1                    uplit (%asciz'%AMAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
;    3117  1                    uplit (%asciz'%AMAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
;    3118  1                    uplit (%asciz'%ACONTROLLER RAM ERROR (Non parity)'),
;    3119  1                    uplit (%asciz'%AINIT SEQUENCE ERROR'),
;    3120  1                    uplit (%asciz'%AHIGHER LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
;    3121  1                    uplit (%asciz'%APURGE/POLL HARDWARE FAILURE'),
;    3122  1                    uplit (%asciz'%AMAPPING REGISTER READ FAILURE (Parity or timeout)')) : vector [23],
;    3123  1          !
;    3124  1          ! RD/RX CONTROLLER DEPENDENT ERRORS CODES
;    3125  1          !
;    3126  1              RDRX_ERR = uplit (
;    3127  1                    uplit (%asciz'%AT11 CPU FAILURE'),
;    3128  1                    uplit (%asciz'%ANON-PARITY RAM ERROR'),
;    3129  1                    uplit (%asciz'%ASTATE MACHINE FAILURE - T11 ADDRESS REGISTER'),
;    3130  1                    uplit (%asciz'%ASTATE MACHINE FAILURE - Q-BUS ADDRESS REGISTER'),
;    3131  1                    uplit (%asciz'%ASTATE MACHINE FAILURE - CRC REGISTER'),
;    3132  1                    uplit (%asciz'%ASTATE MACHINE FAILURE - SERIALIZER/DESERIALIZER REGISTER'),
;    3133  1                    uplit (%asciz'%ASTATE MACHINE FAILURE - WRONG HARDWARE VERSION')) : vector [7],
;    3134  1          !
;    3135  1          ! PRINTOUTS THAT FAKE THE DRS ERROR MESSAGES
;    3136  1          !
;    3137  1              DF_MSG  = uplit (%asciz'%N%AZRQA DEV FTL  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3138  1              HRD_MSG = uplit (%asciz'%N%AZRQA HRD ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06'),
;    3139  1              SFT_MSG = uplit (%asciz'%N%AZRQA SFT ERR  %Z5%A ON UNIT %Z2%A TST 001 SUB 000 PC: %06%N'),
;    3140  1              HRD_SUB = uplit (%asciz'%N%AI/O REQUEST FAILED%N'),
;    3141  1          !
;    3142  1          ! MISCELLANEOUS
;    3143  1          !
;    3144  1              SPACE4    = uplit (%asciz'%S4'),
;    3145  1              CRLF      = uplit (%asciz'%N')
;    3146  1              DASH      = uplit (%asciz'%A - '),
;    3147  1              ASTERISK  = uplit (%asciz'%A* ');
```

```
;    3148  1      %sbttl 'DEFAULT HARDWARE P-TABLE'
;    3149  1
;    3150  1      !+
;    3151  1      ! THE DEFAULT HARDWARE P TABLE CONTAINS DEFAULT VALUES OF
;    3152  1      ! THE TEST DEVICE PARAMETERS.  THE STRUCTURE OF THIS TABLE
;    3153  1      ! IS IDENTICAL TO THE STRUCTURE OF THE HARDWARE P TABLES,
;    3154  1      ! AND IS USED AS A "TEMPLATE" FOR BUILDING THE P TABLES.
;    3155  1      !-
;    3156  1
;    3157  1      BGNHW (DEFTBL);
;    3158  1
;    3159  1      global
;    3160  1          HWPT_IP_ADDR : word initial (INIT_IP_ADDR),          ! IP ADDRESS
;    3161  1          HWPT_VECTOR : word initial (INIT_INTR_VECT),         ! VECTOR ADDRESS
;    3162  1          HWPT_BR_LEVEL : word initial (INIT_BR_LEVEL),        ! BR LEVEL
;    3163  1          HWPT_DISK : WORD INITIAL (%o'100340'),               !PROTECT, WHOLE DISK, DUP WT, RUN DUP   ZZZ
;    3164  1                                                               !DUP WT, DK 0        ZZZ
;    3165  1          HWPTS0_LBN : word initial (0),                       ! STARTING TRACK LO ZZZ
;    3166  1          HWPTS1_LBN : word initial (0),                       ! STARTING TRACK HI ZZZ
;    3167  1          HWPTE0_LBN : word initial (%o'177777'),              ! ENDING TRACK LO   ZZZ
;    3168  1          HWPTE1_LBN : word initial (0);                       ! ENDING TRACK HI   ZZZ
;    3169  1
;    3170  1      ENDHW;
```

```
;    3171  1      %sbttl 'SOFTWARE P TABLE
;    3172  1
;    3173  1      !+
;    3174  1      ! THE SOFTWARE TABLE CONTAINS VARIOUS DATA USED BY THE
;    3175  1      ! PROGRAM AS OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE
;    3176  1      ! SET UP AT ASSEMBLY TIME AND MAY BE VARIED BY THE OPERATOR
;    3177  1      ! AT RUN TIME.
;    3178  1      !
;    3179  1
;    3180  1      BGNSW (SFPTBL);
;    3181  1
;    3182  1      global
;    3183  1          SWP_ERROR : word initial (32),                              ! HARD ERROR LIMIT FOR DROPPING UNIT
;    3184  1          SWP_XFER : WORD INITIAL (0),                                !XFER LIMIT, DEFAULT = QUICK PASS       !ZZZ
;    3185  1          SWP_FLAGS : word initial (SWF_RDM or SWF_CRC or SWF_HWC or SWF_FER or SWF_HRD or SWF_BLK),  ! FLAGS (SEE DOCUMEN
TATION)
;    3186  1          SWP_DPAT : word initial (0),                                ! DATA PATTERN NUMBER
;    3187  1          SWP_RAT : word initial (99),                                ! RD51/52 OPERATION RATIO
;    3188  1          SWP_TIME : word initial (0),                                ! START TIME (HHMM)
;    3189  1          DUPROUND : WORD INITIAL (11),              !NO OF I/Os PER DBN TEST ZZZ
;    3190  1
;    3191  1      !       THE NEXT TWO LOCATIONS SHOULD BE TOGETHER
;    3192  1
;    3193  1          SWP_UCNT : word initial (MAX_UDP_CNT),                      ! USER DATA PATTERN COUNT
;    3194  1          SWP_UDPAT : vector [MAX_UDP_CNT, word];                     ! USER DATA PATTERN
;    3195  1
;    3196  1      ENDSW;
```

```
;    3197  1     %sbttl 'PROTECTION TABLE
;    3198  1
;    3199  1     !+
;    3200  1     ! THIS TABLE IS USED BY THE RUNTIME SERVICES
;    3201  1     ! TO PROTECT THE LOAD MEDIA.
;    3202  1     !
;    3203  1
;    3204  1     BGNPROT (0, -1, 6);
;    3205  1
;    3206  1     !1ST ARG =      OFFSET INTO P-TABLE FOR CSR ADDRESS
;    3207  1     !2ND ARG =      OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;    3208  1     !3RD ARG =      OFFSET INTO P-TABLE FOR DRIVE NUMBER
;    3209  1
;    3210  1     ENDPROT;
;    3211  1     end
;    3212  1
;    3213  0     eludom
```

```
                        .TITLE  ZRQAM1 RD/RX EXERCISER
                        .IDENT  /V01.6/
                        .ENABL  AMA

000000                  .PSECT  $CODE$,  RO
000000  132     122     121     L$NAME::.ASCII  /ZRQ/
000003  101                             .ASCII  /A/
000004  000                             .BYTE   0
000005  000                             .BYTE   0
000006  000                             .BYTE   0
000007  000                             .BYTE   0
000010                          L$REV::
000010  104                             .ASCII  /D/
000011  060                             .ASCII  /0/
000012  000006                  L$UNIT::.WORD   T$PTHV
000014  077777                  L$TIML::.WORD   77777
000016  000006                  L$HPCP::.WORD   L$HARD
000020  000006                  L$SPCP::.WORD   L$SOFT
000022  022176                  L$HPTP::.WORD   L$HW
000024  022222                  L$SPTP::.WORD   L$SW
000026  000006                  L$LADP::.WORD   L$LAST
000030  000000                  L$STA:: .WORD   0
000032  000000                  L$CO::  .WORD   0
000034  000001                  L$DTYP::.WORD   1
000036  000000                  L$APT:: .WORD   0
000040  000124                  L$DTP:: .WORD   L$DISPATCH
000042  000000                  L$PRIO::.WORD   0
000044  000000                  L$ENVI::.WORD   0
000046  000000                  L$EXP1::.WORD   0
000050                          L$MREV::
000050  003                             .BYTE   3
000051  003                             .BYTE   3
000052  000000                  L$EF::  .WORD   0
```

```
000054  000000                          .WORD   0
000056  000000                  L$SPC:: .WORD   0
000060  0000006                 L$DEVP::.WORD   L$DVTP
000062  0000006                 L$REPP::.WORD   L$RPT
000064  000000                  L$EXP4::.WORD   0
000066  000000                  L$EXP5::.WORD   0
000070  0000006                 L$AUT:: .WORD   L$AU
000072  0000006                 L$DUT:: .WORD   L$DU
000074  000000                  L$LUN:: .WORD   0
000076  0000006                 L$DESP::.WORD   L$DESC
000100  104035                  L$LOAD::.WORD    73743
000102  000126'                 L$ETP:: .WORD   L$ERRTBL
000104  0000006                 L$ICP:: .WORD   L$INIT
000106  0000006                 L$CCP:: .WORD   L$CLEAN
000110  0000006                 L$ACP:: .WORD   L$AUTO
000112  022304'                 L$PRT:: .WORD   L$PROT
000114  000000                  L$TEST::.WORD   0
000116  000000                  L$DLY:: .WORD   0
000120  000000                  L$HIME::.WORD   0
000122  000001                  D$PCNT::.WORD   1
000124  0000006                 L$DISPATCH::
                                        .WORD   T1

000126                          ERRTYP::.BLKW   1
000130                          ERRNBR::.BLKW   1
000132                          ERRMSG::.BLKW   1
000134                          ERRBLK::.BLKW   1
000136  111     120     040     P.AAA:  .ASCII  /IP /
000141  141     144     144             .ASCII  /add/
000144  162     145     163             .ASCII  /res/
000147  163     000     000             .ASCII  /s/<00><00>
000152  126     145     143     P.AAB:  .ASCII  /Vec/
000155  164     157     162             .ASCII  /tor/
000160  000     000                     .ASCII  <00><00>
000162  102     122     040     P.AAC:  .ASCII  /BR /
000165  154     145     166             .ASCII  /lev/
000170  145     154     000             .ASCII  /el/<00>
000173  000                             .ASCII  <00>
000174  104     162     151     P.AAD:  .ASCII  /Dri/
000177  166     145     040             .ASCII  /ve /
000202  156     165     155             .ASCII  /num/
000205  142     145     162             .ASCII  /ber/
000210  000     000                     .ASCII  <00><00>
000212  124     145     163     P.AAE:  .ASCII  /Tes/
000215  164     040     145             .ASCII  /t e/
000220  156     164     151             .ASCII  /nti/
000223  162     145     040             .ASCII  /re /
000226  143     165     163             .ASCII  /cus/
000231  164     157     155             .ASCII  /tom/
000234  145     162     040             .ASCII  /er
000237  141     162     145             .ASCII  /are/
000242  141     040     157             .ASCII  /a o/
000245  146     040     164             .ASCII  /f t/
000250  150     151     163             .ASCII  /hig/
```

```
000253  040  144  151                .ASCII  / di/
000256  163  153  000                .ASCII  /sk/<00>
000261  000                         .ASCII  <00>
000262  114  157  167  P.AAF:        .ASCII  /Low/
000265  145  162  040                .ASCII  /er /
000270  157  143  164                .ASCII  /oct/
000273  141  154  040                .ASCII  /al /
000276  167  157  162                .ASCII  /wor/
000301  144  040  157                .ASCII  /d o/
000304  146  040  142                .ASCII  /f b/
000307  145  147  151                .ASCII  /egi/
000312  156  156  151                .ASCII  /nni/
000315  156  147  040                .ASCII  /ng /
000320  114  102  116                .ASCII  /LBN/
000323  040  141  144                .ASCII  / ad/
000326  144  162  145                .ASCII  /dre/
000331  163  163  000                .ASCII  /ss/<00>
000334  110  151  147  P.AAG:        .ASCII  /Hig/
000337  150  145  162                .ASCII  /her/
000342  040  157  143                .ASCII  / oc/
000345  164  141  154                .ASCII  /tal/
000350  040  167  157                .ASCII  / wo/
000353  162  144  040                .ASCII  /rd /
000356  157  146  040                .ASCII  /of /
000361  142  145  147                .ASCII  /beg/
000364  151  156  156                .ASCII  /inn/
000367  151  156  147                .ASCII  /ing/
000372  040  114  102                .ASCII  / LB/
000375  116  040  141                .ASCII  /N a/
000400  144  144  162                .ASCII  /ddr/
000403  145  163  163                .ASCII  /ess/
000406  000  000                     .ASCII  <00><00>
000410  114  157  167  P.AAH:        .ASCII  /Low/
000413  145  162  040                .ASCII  /er /
000416  157  143  164                .ASCII  /oct/
000421  141  154  040                .ASCII  /al /
000424  167  157  162                .ASCII  /wor/
000427  144  040  157                .ASCII  /d o/
000432  146  040  145                .ASCII  /f e/
000435  156  144  151                .ASCII  /ndi/
000440  156  147  040                .ASCII  /ng /
000443  114  102  116                .ASCII  /LBN/
000446  040  141  144                .ASCII  / ad/
000451  144  162  145                .ASCII  /dre/
000454  163  163  000                .ASCII  /ss/<00>
000457  000                         .ASCII  <00>
000460  110  151  147  P.AAI:        .ASCII  /Hig/
000463  150  145  162                .ASCII  /her/
000466  040  157  143                .ASCII  / oc/
000471  164  141  154                .ASCII  /tal/
000474  040  167  157                .ASCII  / wo/
000477  162  144  040                .ASCII  /rd /
000502  157  146  040                .ASCII  /of /
```

```
000505  145  156  144                    .ASCII  /end/
000510  151  156  147                    .ASCII  /ing/
000513  040  114  102                    .ASCII  / LB/
000516  116  040  141                    .ASCII  /N a/
000521  144  144  162                    .ASCII  /ddr/
000524  145  163  163                    .ASCII  /ess/
000527  000                             .ASCII  <00>
000530  127  162  151          P.AAJ:    .ASCII  /Wri/
000533  164  145  040                    .ASCII  /te /
000536  157  156  040                    .ASCII  /on /
000541  143  165  163                    .ASCII  /cus/
000544  164  157  155                    .ASCII  /tom/
000547  145  162  040                    .ASCII  /er /
000552  144  141  164                    .ASCII  /dat/
000555  141  040  141                    .ASCII  /a a/
000560  162  145  141                    .ASCII  /rea/
000563  040  157  156                    .ASCII  / on/
000566  040  164  150                    .ASCII  / th/
000571  151  163  040                    .ASCII  /is /
000574  144  151  163                    .ASCII  /dis/
000577  153  000  000                    .ASCII  /k/<00><00>
000602  052  052  040          P.AAK:    .ASCII  /** /
000605  127  101  122                    .ASCII  /WAR/
000610  116  111  116                    .ASCII  /NIN/
000613  107  040  055                    .ASCII  /G -/
000616  040  103  125                    .ASCII  / CU/
000621  123  124  117                    .ASCII  /STO/
000624  115  105  122                    .ASCII  /MER/
000627  040  104  101                    .ASCII  / DA/
000632  124  101  040                    .ASCII  /TA /
000635  101  122  105                    .ASCII  /ARE/
000640  101  040  115                    .ASCII  /A M/
000643  101  131  040                    .ASCII  /AY /
000646  102  105  040                    .ASCII  /BE /
000651  117  126  105                    .ASCII  /OVE/
000654  122  127  122                    .ASCII  /RWR/
000657  111  124  124                    .ASCII  /ITT/
000662  105  116  041                    .ASCII  /EN!/
000665  040  056  056                    .ASCII  / ../
000670  056  040  103                    .ASCII  /. C/
000673  117  116  106                    .ASCII  /ONF/
000676  111  122  115                    .ASCII  /IRM/
000701  000                             .ASCII  <00>
000702  101  154  163          P.AAL:    .ASCII  /Als/
000705  157  040  162                    .ASCII  /o r/
000710  165  156  040                    .ASCII  /un /
000713  104  125  120                    .ASCII  /DUP/
000716  040  145  170                    .ASCII  / ex/
000721  145  162  143                    .ASCII  /erc/
000724  151  163  145                    .ASCII  /ise/
000727  162  000  000                    .ASCII  /r/<00><00>
000732  127  162  151          P.AAM:    .ASCII  /Wri/
000735  164  145  040                    .ASCII  /te /
```

```
000740  157  156  040                .ASCII  /on /
000743  144  151  141                .ASCII  /dia/
000746  147  156  157                .ASCII  /gno/
000751  163  164  151                .ASCII  /sti/
000754  143  040  141                .ASCII  /c a/
000757  162  145  141                .ASCII  /rea/
000762  000  000                     .ASCII  <00><00>
000764  110  141  162     P.AAN:     .ASCII  /Har/
000767  144  040  105                .ASCII  /d E/
000772  162  162  157                .ASCII  /rro/
000775  162  040  154                .ASCII  /r l/
001000  151  155  151                .ASCII  /imi/
001003  164  000  000                .ASCII  /t/<00><00>
001006  124  162  141     P.AAO:     .ASCII  /Tra/
001011  156  163  146                .ASCII  /nsf/
001014  145  162  040                .ASCII  /er /
001017  154  151  155                .ASCII  /lim/
001022  151  164  040                .ASCII  /it /
001025  151  156  040                .ASCII  /in /
001030  155  145  147                .ASCII  /meg/
001033  141  142  171                .ASCII  /aby/
001036  164  145  163                .ASCII  /tes/
001041  040  050  060                .ASCII  / (0/
001044  040  146  157                .ASCII  / fo/
001047  162  040  042                .ASCII  /r "/
001052  121  165  151                .ASCII  /Qui/
001055  143  153  040                .ASCII  /ck /
001060  160  141  163                .ASCII  /pas/
001063  163  042  051                .ASCII  /s")/
001066  000  000                     .ASCII  <00><00>
001070  122  141  156     P.AAP:     .ASCII  /Ran/
001073  144  157  155                .ASCII  /dom/
001076  040  163  145                .ASCII  / se/
001101  145  153  040                .ASCII  /ek /
001104  155  157  144                .ASCII  /mod/
001107  145  000  000                .ASCII  /e/<00><00>
001112  122  145  141     P.AAQ:     .ASCII  /Rea/
001115  144  055  143                .ASCII  /d c/
001120  157  155  160                .ASCII  /omp/
001123  141  162  145                .ASCII  /are/
001126  163  040  160                .ASCII  /s p/
001131  145  162  146                .ASCII  /erf/
001134  157  162  155                .ASCII  /orm/
001137  145  144  040                .ASCII  /ed /
001142  141  164  040                .ASCII  /at /
001145  164  150  145                .ASCII  /the/
001150  040  103  157                .ASCII  / Co/
001153  156  164  162                .ASCII  /ntr/
001156  157  154  154                .ASCII  /oll/
001161  145  162  000                .ASCII  /er/<00>
001164  127  162  151     P.AAR:     .ASCII  /Wri/
001167  164  145  055                .ASCII  /te /
001172  143  157  155                .ASCII  /com/
```

```
001175  160  141  162                .ASCII  /par/
001200  145  163  040                .ASCII  /es /
001203  160  145  162                .ASCII  /per/
001206  146  157  162                .ASCII  /for/
001211  155  145  144                .ASCII  /med/
001214  040  141  164                .ASCII  / at/
001217  040  164  150                .ASCII  / th/
001222  145  040  103                .ASCII  /e C/
001225  157  156  164                .ASCII  /ont/
001230  162  157  154                .ASCII  /rol/
001233  154  145  162                .ASCII  /ler/
001236  000  000                     .ASCII  <00><00>
001240  103  150  145     P.AAS:     .ASCII  /Che/
001243  143  153  040                .ASCII  /ck /
001246  141  154  154                .ASCII  /all/
001251  040  127  162                .ASCII  / Wr/
001254  151  164  145                .ASCII  /ite/
001257  163  040  141                .ASCII  /s a/
001262  164  040  110                .ASCII  /t H/
001265  157  163  164                .ASCII  /ost/
001270  040  142  171                .ASCII  / by/
001273  040  162  145                .ASCII  / re/
001276  141  144  151                .ASCII  /adi/
001301  156  147  000                .ASCII  /ng/<00>
001304  125  163  145     P.AAT:     .ASCII  /Use/
001307  162  055  144                .ASCII  /r d/
001312  145  146  151                .ASCII  /efi/
001315  156  145  144                .ASCII  /ned/
001320  040  144  141                .ASCII  / da/
001323  164  141  040                .ASCII  /ta /
001326  160  141  164                .ASCII  /pat/
001331  164  145  162                .ASCII  /ter/
001334  156  000                     .ASCII  /n/<00>
001336  123  145          P.AAU:     .ASCII  /Sel/
001341  145  143  164                .ASCII  /ect/
001344  040  160  162                .ASCII  / pr/
001347  145  055  144                .ASCII  /e-d/
001352  145  146  151                .ASCII  /efi/
001355  156  145  144                .ASCII  /ned/
001360  040  144  141                .ASCII  / da/
001363  164  141  040                .ASCII  /ta /
001366  160  141  164                .ASCII  /pat/
001371  164  145  162                .ASCII  /ter/
001374  156  040  050                .ASCII  /n (/
001377  060  040  146                .ASCII  /0 f/
001402  157  162  040                .ASCII  /or /
001405  123  145  161                .ASCII  /Seq/
001410  165  145  156                .ASCII  /uen/
001413  164  151  141                .ASCII  /tia/
001416  154  040  163                .ASCII  /l s/
001421  145  154  145                .ASCII  /ele/
001424  143  164  151                .ASCII  /cti/
001427  157  156  051                .ASCII  /on)/
```

```
001462  000  000                    .ASCII  <00><00>
001464  116  165  155    P.AAV:     .ASCII  /Num/
001467  142  145  162               .ASCII  /ber/
001472  040  157  146               .ASCII  / of/
001475  040  167  157               .ASCII  / wo/
001500  162  144  163               .ASCII  /rds/
001503  040  151  156               .ASCII  / in/
001506  040  144  141               .ASCII  / da/
001511  164  141  040               .ASCII  /ta /
001514  160  141  164               .ASCII  /pat/
001517  164  145  162               .ASCII  /ter/
001522  156  040  050               .ASCII  /n (/
001525  061  066  040               .ASCII  /16 /
001530  155  141  170               .ASCII  /max/
001533  151  155  165               .ASCII  /imu/
001536  155  051  000               .ASCII  /m)/<00>
001541  000                        .ASCII  <00>
001542  120  141  164    P.AAW:     .ASCII  /Pat/
001545  162  145  162               .ASCII  /ter/
001550  156  040  166               .ASCII  /n v/
001553  141  154  165               .ASCII  /alu/
001556  145  040  050               .ASCII  /e (/
001561  116  157  040               .ASCII  /No /
001564  154  145  141               .ASCII  /lea/
001567  144  151  156               .ASCII  /din/
001572  147  040  172               .ASCII  /g z/
001575  145  162  157               .ASCII  /ero/
001600  163  040  141               .ASCII  /s a/
001603  154  154  157               .ASCII  /llo/
001606  167  145  144               .ASCII  /wed/
001611  051  000  000               .ASCII  /)/<00><00>
001614  103  154  145    P.AAX:     .ASCII  /Cle/
001617  141  162  040               .ASCII  /ar /
001622  163  164  141               .ASCII  /sta/
001625  164  151  163               .ASCII  /tis/
001630  164  151  143               .ASCII  /tic/
001633  141  154  040               .ASCII  /al /
001636  164  141  142               .ASCII  /tab/
001641  154  145  163               .ASCII  /les/
001644  040  141  146               .ASCII  / af/
001647  164  145  162               .ASCII  /ter/
001652  040  160  162               .ASCII  / pr/
001655  151  156  164               .ASCII  /int/
001660  151  156  147               .ASCII  /ing/
001663  000                        .ASCII  <00>
001664  120  145  162    P.AAY:     .ASCII  /Per/
001667  143  145  156               .ASCII  /cen/
001672  164  141  147               .ASCII  /tag/
001675  145  040  157               .ASCII  /e o/
001700  146  040  040               .ASCII  /f  /
001703  106  151  170               .ASCII  /Fix/
001706  145  144  040               .ASCII  /ed /
001711  104  151  163               .ASCII  /Dis/
```

```
001664  153  042  040            .ASCII  /k" /
001667  157  160  145            .ASCII  /ope/
001672  162  141  164            .ASCII  /rat/
001675  151  157  156            .ASCII  /ion/
001700  163  040  157            .ASCII  /s o/
001703  165  164  040            .ASCII  /ut /
001706  157  146  040            .ASCII  /of /
001711  164  157  164            .ASCII  /tot/
001714  141  154  040            .ASCII  /al /
001717  157  160  145            .ASCII  /ope/
001722  162  141  164            .ASCII  /rat/
001725  151  157  156            .ASCII  /ion/
001730  163  000                 .ASCII  /s/<00>
001732  125  156  151  P.AAZ:     .ASCII  /Uni/
001735  164  163  040            .ASCII  /t  /
001740  164  157  040            .ASCII  /to /
001743  142  145  040            .ASCII  /be /
001746  163  145  154            .ASCII  /sel/
001751  145  143  164            .ASCII  /ect/
001754  145  144  040            .ASCII  /ed /
001757  141  164  040            .ASCII  /at /
001762  162  141  156            .ASCII  /ran/
001765  144  157  155            .ASCII  /dom/
001770  040  050  116            .ASCII  / (N/
001773  157  054  040            .ASCII  /o, /
001776  151  155  160            .ASCII  /imp/
002001  154  151  145            .ASCII  /lie/
002004  163  040  163            .ASCII  /s s/
002007  145  161  165            .ASCII  /equ/
002012  145  156  164            .ASCII  /ent/
002015  151  141  154            .ASCII  /ial/
002020  051  000                 .ASCII  /)/<00>
002022  127  141  156  P.ABA:     .ASCII  /Wan/
002025  164  040  164            .ASCII  /t t/
002030  157  040  162            .ASCII  /o r/
002033  145  167  162            .ASCII  /ewr/
002036  151  164  145            .ASCII  /ite/
002041  040  142  154            .ASCII  / bl/
002044  157  143  153            .ASCII  /ock/
002047  163  040  167            .ASCII  /s w/
002052  150  145  156            .ASCII  /hen/
002055  040  042  106            .ASCII  / "F/
002060  157  162  143            .ASCII  /orc/
002063  145  144  040            .ASCII  /ed /
002066  105  162  162            .ASCII  /Err/
002071  157  162  042            .ASCII  /or"/
002074  040  144  145            .ASCII  / de/
002077  164  145  143            .ASCII  /tec/
002102  164  145  144            .ASCII  /ted/
002105  040  157  156            .ASCII  / on/
002110  040  162  145            .ASCII  / re/
002113  141  144  163            .ASCII  /ads/
002116  000  000                 .ASCII  <00><00>
```

```
002120  104  157  040  P.ABB:  .ASCII  /Do /
002123  171  157  165          .ASCII  /you/
002126  040  167  141          .ASCII  / wa/
002131  156  164  040          .ASCII  /nt /
002134  164  157  040          .ASCII  /to /
002137  150  141  154          .ASCII  /hal/
002142  164  040  157          .ASCII  /t o/
002145  156  040  157          .ASCII  /n o/
002150  164  150  145          .ASCII  /the/
002153  162  040  110          .ASCII  /r H/
002156  141  162  144          .ASCII  /ard/
002161  040  105  162          .ASCII  / Er/
002164  162  157  162          .ASCII  /ror/
002167  163  040  050          .ASCII  /s (/
002172  043  163  040          .ASCII  /#s /
002175  063  061  055          .ASCII  /31-/
002200  063  064  054          .ASCII  /34,/
002203  040  063  066          .ASCII  / 36/
002206  055  063  067          .ASCII  /-37/
002211  054  040  063          .ASCII  /, 3/
002214  071  055  064          .ASCII  /9-4/
002217  065  051  000          .ASCII  /5)/<00>
002222  104  157  040  P.ABC:  .ASCII  /Do /
002225  171  157  165          .ASCII  /you/
002230  040  167  141          .ASCII  / wa/
002233  156  164  040          .ASCII  /nt /
002236  164  157  040          .ASCII  /to /
002241  150  141  154          .ASCII  /hal/
002244  164  040  157          .ASCII  /t o/
002247  156  040  123          .ASCII  /n S/
002252  157  146  164          .ASCII  /oft/
002255  040  105  162          .ASCII  / Er/
002260  162  157  162          .ASCII  /ror/
002263  163  040  050          .ASCII  /s (/
002266  043  163  040          .ASCII  /#s /
002271  065  060  055          .ASCII  /50-/
002274  065  064  051          .ASCII  /54)/
002277  000                    .ASCII  <00>
002300  104  157  040  P.ABD:  .ASCII  /Do /
002303  171  157  165          .ASCII  /you/
002306  040  167  141          .ASCII  / wa/
002311  156  164  040          .ASCII  /nt /
002314  164  157  040          .ASCII  /to /
002317  150  141  154          .ASCII  /hal/
002322  164  040  157          .ASCII  /t o/
002325  156  040  102          .ASCII  /n B/
002330  141  144  055          .ASCII  /ad /
002333  142  154  157          .ASCII  /blo/
002336  143  153  040          .ASCII  /ck /
002341  110  141  162          .ASCII  /Har/
002344  144  040  105          .ASCII  /d E/
002347  162  162  157          .ASCII  /rro/
002352  162  163  040          .ASCII  /rs /
```

```
002355  050     043     163             .ASCII  /(#s/
002360  040     063     065             .ASCII  / 35/
002363  054     040     063             .ASCII  /, 3/
002366  070     051     000             .ASCII  /8)/<00>
002371  000                             .ASCII  <00>
002372  105     156     164     P.ABE:  .ASCII  /Ent/
002375  145     162     040             .ASCII  /er /
002400  164     151     155             .ASCII  /tim/
002403  145     040     141             .ASCII  /e a/
002406  163     040     110             .ASCII  /s H/
002411  110     115     115             .ASCII  /HMM/
002414  040     050     105             .ASCII  / (E/
002417  170     141     155             .ASCII  /xam/
002422  160     154     145             .ASCII  /ple/
002425  072     040     061             .ASCII  /: 1/
002430  063     060     065             .ASCII  /305/
002433  040     055     040             .ASCII  / - /
002436  040     116     157             .ASCII  / No/
002441  040     154     145             .ASCII  / le/
002444  141     144     151             .ASCII  /adi/
002447  156     147     040             .ASCII  /ng /
002452  172     145     162             .ASCII  /zer/
002455  157     163     040             .ASCII  /os /
002460  141     154     154             .ASCII  /all/
002463  157     167     144             .ASCII  /owd/
002466  051     000                     .ASCII  /)/<00>
002470  103     157     165     P.ABF:  .ASCII  /Cou/
002473  156     164     040             .ASCII  /nt /
002476  145     141     143             .ASCII  /eac/
002501  150     040     162             .ASCII  /h r/
002504  145     164     162             .ASCII  /etr/
002507  171     040     157             .ASCII  /y o/
002512  156     040     141             .ASCII  /n a/
002515  040     122     145             .ASCII  / Re/
002520  141     144     057             .ASCII  /ad.<57>
002523  127     162     151             .ASCII  /Wri/
002526  164     145     040             .ASCII  /te /
002531  145     162     162             .ASCII  /err/
002534  157     162     040             .ASCII  /or /
002537  141     163     040             .ASCII  /as /
002542  141     040     163             .ASCII  /a s/
002545  145     160     145             .ASCII  /epe/
002550  162     141     164             .ASCII  /rat/
002553  145     040     123             .ASCII  /e S/
002556  157     146     164             .ASCII  /oft/
002561  040     105     162             .ASCII  / Er/
002564  162     157     162             .ASCII  /ror/
002567  000                             .ASCII  <00>
002570  124     150     145     P.ABG:  .ASCII  /The/
002573  040     162     145             .ASCII  / re/
002576  155     141     151             .ASCII  /mai/
002601  156     151     156             .ASCII  /nin/
002604  147     040     161             .ASCII  /g q/
```

```
002607  165  145  163                    .ASCII  /ues/
002612  164  151  157                    .ASCII  /tio/
002615  156  163  040                    .ASCII  /ns /
002620  157  156  154                    .ASCII  /onl/
002623  171  040  141                    .ASCII  /y a/
002626  160  160  154                    .ASCII  /ppl/
002631  171  040  164                    .ASCII  /y t/
002634  157  040  165                    .ASCII  /o u/
002637  156  160  162                    .ASCII  /npr/
002642  157  164  145                    .ASCII  /ote/
002645  143  164  145                    .ASCII  /cte/
002650  144  040  144                    .ASCII  /d d/
002653  151  163  153                    .ASCII  /isk/
002656  163  000                        .ASCII  /s/<00>
002660  000  000              P.ABH:    .ASCII  <00><00>
002662  045  116  045        P.ABI:    .ASCII  /%N%/
002665  101  052  052                    .ASCII  /A**/
002670  040  104  162                    .ASCII  / Dr/
002673  157  160  040                    .ASCII  /op /
002676  165  156  151                    .ASCII  /uni/
002701  164  040  045                    .ASCII  /t %/
002704  104  062  000                    .ASCII  /D2/<00>
002707  000                            .ASCII  <00>
002710  045  116  045        P.ABJ:    .ASCII  /%N%/
002713  101  052  052                    .ASCII  /A**/
002716  040  120  122                    .ASCII  / PR/
002721  117  103  137                    .ASCII  /OC_/
002724  122  105  124                    .ASCII  /RET/
002727  120  113  124                    .ASCII  /PKT/
002732  072  040  103                    .ASCII  /: C/
002735  157  156  156                    .ASCII  /onn/
002740  040  111  104                    .ASCII  / ID/
002743  040  045  117                    .ASCII  / %O/
002746  066  045  101                    .ASCII  /6%A/
002751  040  162  145                    .ASCII  / re/
002754  143  145  151                    .ASCII  /cei/
002757  166  145  144                    .ASCII  /ved/
002762  000  000                        .ASCII  <00><00>
002764  045  116  045        P.ABK:    .ASCII  /%N%/
002767  101  052  052                    .ASCII  /A**/
002772  040  115  165                    .ASCII  / Mu/
002775  154  164  151                    .ASCII  /lti/
003000  055  144  162                    .ASCII  /-dr/
003003  151  166  145                    .ASCII  /ive/
003006  040  164  145                    .ASCII  / te/
003011  163  164  000                    .ASCII  /st/<00>
003014  045  116  045        P.ABL:    .ASCII  /%N%/
003017  101  052  052                    .ASCII  /A**/
003022  040  106  101                    .ASCII  / FA/
003025  124  101  114                    .ASCII  /TAL/
003030  137  105  122                    .ASCII  /_ER/
003033  122  117  122                    .ASCII  /ROR/
003036  072  040  122                    .ASCII  /: R/
```

```
003041  105  124  120          .ASCII  /ETP/
003044  113  124  040          .ASCII  /KT /
003047  156  157  164          .ASCII  /no!/
003052  040  141  166          .ASCII  / av/
003055  141  151  154          .ASCII  /ail/
003060  141  142  154          .ASCII  /abl/
003063  145  000  000          .ASCII  /e/<00><00>
003066  045  116  045  P.ABM:  .ASCII  /%N%/
003071  101  052  052          .ASCII  /A**/
003074  040  106  123          .ASCII  / FS/
003077  105  124  137          .ASCII  /ET_/
003102  125  120  101          .ASCII  /UPA/
003105  122  072  040          .ASCII  /R: /
003110  103  141  156          .ASCII  /Can/
003113  047  164  040          .ASCII  /'t /
003116  146  151  156          .ASCII  /fin/
003121  144  040  144          .ASCII  /d d/
003124  151  163  153          .ASCII  /isk/
003127  040  045  104          .ASCII  / %D/
003132  063  045  101          .ASCII  /3%A/
003135  040  151  156          .ASCII  / in/
003140  040  103  123          .ASCII  / CS/
003143  124  040  045          .ASCII  /T %/
003146  104  061  000          .ASCII  /D1/<00>
003151  000                    .ASCII  <00>
003152  045  116  045  P.ABN:  .ASCII  /%N%/
003155  101  052  052          .ASCII  /A**/
003160  040  102  141          .ASCII  / Ba/
003163  144  040  143          .ASCII  /d c/
003166  157  156  156          .ASCII  /onn/
003171  040  111  104          .ASCII  / ID/
003174  040  045  117          .ASCII  / %O/
003177  066  045  101          .ASCII  /6%A/
003202  040  162  145          .ASCII  / re/
003205  143  145  151          .ASCII  /cei/
003210  166  145  144          .ASCII  /ved/
003213  040  146  162          .ASCII  / fr/
003216  157  155  040          .ASCII  /om /
003221  045  117  066          .ASCII  /%O6/
003224  000  000                .ASCII  <00><00>
003226  045  116  045  P.ABU:  .ASCII  /%N%/
003231  101  052  052          .ASCII  /A**/
003234  040  115  145          .ASCII  / Me/
003237  163  163  141          .ASCII  /ssa/
003242  147  145  040          .ASCII  /ge /
003245  164  171  160          .ASCII  /typ/
003250  145  040  045          .ASCII  /e %/
003253  117  066  045          .ASCII  /O6%/
003256  101  040  162          .ASCII  /A r/
003261  145  143  145          .ASCII  /ece/
003264  151  166  145          .ASCII  /ive/
003267  144  040  151          .ASCII  /d i/
003272  156  040  115          .ASCII  /n M/
```

```
003325  123  103  120                .ASCII  /SCP/
003330  040  160  141                .ASCII  / pa/
003335  143  153  145                .ASCII  /cke/
003340  164  000                     .ASCII  /t/<00>
003343  045  116  045     P.ABP:     .ASCII  /%N%/
003346  101  052  052                .ASCII  /A**/
003351  040  123  105                .ASCII  / SE/
003354  121  125  105                .ASCII  /QUE/
003357  116  072  040                .ASCII  /N: /
003362  122  105  124                .ASCII  /RET/
003365  120  113  124                .ASCII  /PKT/
003370  040  156  157                .ASCII  / no/
003373  164  040  141                .ASCII  /t a/
003376  166  141  151                .ASCII  /vai/
003401  154  141  142                .ASCII  /lab/
003404  154  145  000                .ASCII  /le/<00>
003407  045  116  045     P.ABQ:     .ASCII  /%N%/
003412  101  052  052                .ASCII  /A**/
003415  040  105  162                .ASCII  / Er/
003420  162  157  162                .ASCII  /ror/
003423  040  151  156                .ASCII  / in/
003426  040  123  105                .ASCII  / SE/
003431  124  137  103                .ASCII  /T_C/
003434  124  114  122                .ASCII  /TLR/
003437  137  103  110                .ASCII  /_CH/
003442  101  122  000                .ASCII  /AR/<00>
003445  045  116  045     P.ABR:     .ASCII  /%N%/
003450  101  052  052                .ASCII  /A**/
003453  040  103  164                .ASCII  / Ct/
003456  154  162  040                .ASCII  /lr /
003461  164  151  155                .ASCII  /tim/
003464  145  157  165                .ASCII  /eou/
003467  164  040  075                .ASCII  /t =/
003472  040  045  104                .ASCII  / %D/
003475  063  045  101                .ASCII  /3%A/
003500  056  040  163                .ASCII  /. s/
003503  145  143  157                .ASCII  /eco/
003506  156  144  163                .ASCII  /nds/
003511  000  000                     .ASCII  <00><00>
003513  045  116  045     P.ABS:     .ASCII  /%N%/
003516  101  052  052                .ASCII  /A**/
003521  040  105  162                .ASCII  / Er/
003524  162  157  162                .ASCII  /ror/
003527  040  151  156                .ASCII  / in/
003532  040  125  116                .ASCII  / UN/
003535  111  124  137                .ASCII  /IT_/
003540  111  116  111                .ASCII  /INI/
003543  124  000                     .ASCII  /T/<00>
003545  045  116  045     P.ABT:     .ASCII  /%N%/
003550  101  052  052                .ASCII  /A**/
003553  040  125  116                .ASCII  / UN/
003556  111  124  137                .ASCII  /IT_/
003561  111  116  111                .ASCII  /INI/
```

```
003531  124  072  040            .ASCII  /T: /
003534  122  105  124            .ASCII  /RET/
003537  120  113  124            .ASCII  /PKT/
003542  040  150  141            .ASCII  / ha/
003545  163  040  142            .ASCII  /s b/
003550  141  144  040            .ASCII  /ad /
003553  105  116  104            .ASCII  /END/
003556  103  117  104            .ASCII  /COD/
003561  105  000  000            .ASCII  /E/<00><00>
003564  045  116  045  P.ABU:    .ASCII  /%N%/
003567  101  052  052            .ASCII  /A**/
003572  040  125  156            .ASCII  / Un/
003575  151  164  040            .ASCII  /it /
003600  163  151  172            .ASCII  /siz/
003603  145  040  050            .ASCII  /e (/
003606  114  157  051            .ASCII  /Lo)/
003611  040  075  040            .ASCII  / = /
003614  045  104  065            .ASCII  /%D5/
003617  045  101  056            .ASCII  /%A./
003622  000  000                 .ASCII  <00><00>
003624  045  116  045  P.ABV:    .ASCII  /%N%/
003627  101  052  052            .ASCII  /A**/
003632  040  125  156            .ASCII  / Un/
003635  151  164  040            .ASCII  /it /
003640  163  151  172            .ASCII  /siz/
003643  145  040  050            .ASCII  /e (/
003646  110  151  051            .ASCII  /Hi)/
003651  040  075  040            .ASCII  / = /
003654  045  104  065            .ASCII  /%D5/
003657  045  101  056            .ASCII  /%A./
003662  000  000                 .ASCII  <00><00>
003664  045  116  045  P.ABW:    .ASCII  /%N%/
003667  101  052  052            .ASCII  /A**/
003672  040  101  103            .ASCII  / AC/
003675  103  105  123            .ASCII  /CES/
003700  123  072  040            .ASCII  /S: /
003703  122  105  124            .ASCII  /RET/
003706  120  113  124            .ASCII  /PKT/
003711  040  150  141            .ASCII  / ha/
003714  163  040  142            .ASCII  /s b/
003717  141  144  040            .ASCII  /ad /
003722  105  116  104            .ASCII  /END/
003725  103  117  104            .ASCII  /COD/
003730  105  000                 .ASCII  /E/<00>
003732  045  116  045  P.ABX:    .ASCII  /%N%/
003735  101  052  052            .ASCII  /A**/
003740  040  121  111            .ASCII  / QI/
003743  117  137  125            .ASCII  /O_U/
003746  116  111  124            .ASCII  /NIT/
003751  072  040  103            .ASCII  /: C/
003754  123  124  040            .ASCII  /ST /
003757  045  104  061            .ASCII  /%D1/
003762  045  101  040            .ASCII  /%A /
```

```
003765  156  157  040            .ASCII  /no /
003770  165  156  151            .ASCII  /uni/
003773  164  040  163            .ASCII  /t s/
003776  145  154  145            .ASCII  /ele/
004001  143  164  145            .ASCII  /cte/
004004  144  000                 .ASCII  /d/<00>
004006  045  116  045  P.ABY:     .ASCII  /%N%/
004011  101  052  052            .ASCII  /A**/
004014  040  125  156            .ASCII  / Un/
004017  151  164  040            .ASCII  /it /
004022  043  040  151            .ASCII  /# i/
004025  163  072  040            .ASCII  /s: /
004030  045  117  066            .ASCII  /%06/
004033  000                      .ASCII  <00>
004034  045  116  045  P.ABZ:     .ASCII  /%N%/
004037  101  052  052            .ASCII  /A**/
004042  040  122  145            .ASCII  / Re/
004045  155  157  166            .ASCII  /mov/
004050  141  142  154            .ASCII  /abl/
004053  145  040  144            .ASCII  /e d/
004056  151  163  153            .ASCII  /isk/
004061  040  151  163            .ASCII  / is/
004064  040  163  145            .ASCII  / se/
004067  154  145  143            .ASCII  /lec/
004072  164  145  144            .ASCII  /ted/
004075  000                      .ASCII  <00>
004076  045  116  045  P.ACA:     .ASCII  /%N%/
004101  101  052  052            .ASCII  /A**/
004104  040  106  151            .ASCII  / Fi/
004107  170  145  144            .ASCII  /xed/
004112  040  144  151            .ASCII  / di/
004115  163  153  040            .ASCII  /sk /
004120  151  163  040            .ASCII  /is /
004123  163  145  154            .ASCII  /sel/
004126  145  143  164            .ASCII  /ect/
004131  145  144  000            .ASCII  /ed/<00>
004134  045  116  045  P.ACB:     .ASCII  /%N%/
004137  101  052  052            .ASCII  /A**/
004142  040  111  154            .ASCII  / Il/
004145  154  145  147            .ASCII  /leg/
004150  141  154  040            .ASCII  /al /
004153  146  165  156            .ASCII  /fun/
004156  143  164  151            .ASCII  /cti/
004161  157  156  072            .ASCII  /on:/
004164  040  045  117            .ASCII  / %O/
004167  066  000  000            .ASCII  /6/<00><00>
004172  045  116  045  P.ACC:     .ASCII  /%N%/
004175  101  052  052            .ASCII  /A**/
004200  040  103  157            .ASCII  / Co/
004203  155  155  141            .ASCII  /mma/
004206  156  144  040            .ASCII  /nd /
004211  162  145  146            .ASCII  /ref/
004214  040  043  040            .ASCII  / # /
```

```
004217  045  117  066              .ASCII  /%06/
004222  045  101  057              .ASCII  /%A/<57>
004225  045  117  066              .ASCII  /%06/
004230  045  101  040              .ASCII  /%A /
004233  050  117  143              .ASCII  /(Oc/
004236  164  051  040              .ASCII  /t) /
004241  156  157  164              .ASCII  /not/
004244  040  163  145              .ASCII  / se/
004247  156  164  040              .ASCII  /nt /
004252  142  171  040              .ASCII  /by /
004255  110  157  163              .ASCII  /Hos/
004260  164  000                   .ASCII  /t/<00>
004262  045  116  045    P.ACD:    .ASCII  /%N%/
004265  101  052  052              .ASCII  /A**/
004270  040  125  156              .ASCII  / Un/
004273  153  156  157              .ASCII  /kno/
004276  167  156  040              .ASCII  /wn /
004301  105  162  162              .ASCII  /Err/
004304  157  162  040              .ASCII  /or /
004307  114  157  147              .ASCII  /Log/
004312  040  146  157              .ASCII  / fo/
004315  162  155  141              .ASCII  /rma/
004320  164  040  045              .ASCII  /t %/
004323  117  063  045              .ASCII  /03%/
004326  101  040  162              .ASCII  /A r/
004331  145  143  145              .ASCII  /ece/
004334  151  166  145              .ASCII  /ive/
004337  144  000  000              .ASCII  /d/<00><00>
004342  045  116  045    P.ACE:    .ASCII  /%N%/
004345  101  052  052              .ASCII  /A**/
004350  040  117  160              .ASCII  / Op/
004353  055  143  157              .ASCII  /-co/
004356  144  145  040              .ASCII  /de /
004361  045  117  063              .ASCII  /%03/
004364  045  101  054              .ASCII  /%A,/
004367  040  105  156              .ASCII  / En/
004372  144  055  143              .ASCII  /d-c/
004375  157  144  145              .ASCII  /ode/
004400  040  045  117              .ASCII  / %O/
004403  063  045  101              .ASCII  /3%A/
004406  040  146  157              .ASCII  / fo/
004411  162  040  162              .ASCII  /r r/
004414  145  146  040              .ASCII  /ef /
004417  045  040  045              .ASCII  /% %/
004422  117  066  045              .ASCII  /O6%/
004425  101  057  045              .ASCII  /A/<57>/%/
004430  117  066  045              .ASCII  /O6%/
004433  101  040  050              .ASCII  /A (/
004436  070  051  000              .ASCII  /8)/<00>
004441  000                       .ASCII  <00>
004442  045  116  045    P.ACF:    .ASCII  /%N%/
004445  101  052  052              .ASCII  /A**/
004450  040  103  155              .ASCII  / Cm/
```

```
004453  144  055  142                .ASCII  /d-b/
004456  143  040  045                .ASCII  /c %/
004461  117  066  045                .ASCII  /06%/
004464  101  057  045                .ASCII  /A/<57>/%/
004467  117  066  045                .ASCII  /06%/
004472  101  040  122                .ASCII  /A R/
004475  163  160  055                .ASCII  /sp-/
004500  142  143  040                .ASCII  /bc /
004503  045  117  066                .ASCII  /%06/
004506  045  101  057                .ASCII  /%A/<57>
004511  045  117  066                .ASCII  /%06/
004514  045  101  040                .ASCII  /%A /
004517  146  157  162                .ASCII  /for/
004522  040  045  117                .ASCII  / %O/
004525  066  045  101                .ASCII  /6%A/
004530  057  045  117                .ASCII  <57>/%O/
004533  066  045  101                .ASCII  /6%A/
004536  040  050  070                .ASCII  / (8/
004541  051  000  000                .ASCII  /)/<00><00>
004544  045  116  045  P.ACG:        .ASCII  /%N%/
004547  101  052  052                .ASCII  /A**/
004552  040  122  145                .ASCII  / Re/
004555  163  160  157                .ASCII  /spo/
004560  156  163  145                .ASCII  /nse/
004563  040  141  154                .ASCII  / al/
004566  162  145  141                .ASCII  /rea/
004571  144  171  040                .ASCII  /dy /
004574  162  145  143                .ASCII  /rec/
004577  145  151  166                .ASCII  /eiv/
004602  145  144  040                .ASCII  /ed /
004605  146  157  162                .ASCII  /for/
004610  040  143  155                .ASCII  / cm/
004613  144  040  045                .ASCII  /d %/
004616  117  066  045                .ASCII  /06%/
004621  101  057  045                .ASCII  /A/<57>/%/
004624  117  066  045                .ASCII  /06%/
004627  101  040  050                .ASCII  /A (/
004632  070  051  000                .ASCII  /8)/<00>
004635  000                         .ASCII  <00>
004636  045  116  045  P.ACH:        .ASCII  /%N%/
004641  101  052  052                .ASCII  /A**/
004644  040  106  141                .ASCII  / fa/
004647  151  154  165                .ASCII  /ilu/
004652  162  145  040                .ASCII  /re /
004655  164  157  040                .ASCII  /to /
004660  163  145  156                .ASCII  /sen/
004663  144  040  143                .ASCII  /d c/
004666  155  155  155                .ASCII  /omm/
004671  141  156  144                .ASCII  /and/
004674  040  141  146                .ASCII  / af/
004677  164  145  162                .ASCII  /ter/
004702  040  043  040                .ASCII  / # /
004705  045  117  066                .ASCII  /%06/
```

```
004710  045  101  057                .ASCII  /%A/<57>
004713  045  117  060                .ASCII  /%06/
004716  045  101  040                .ASCII  /%A /
004721  050  070  051                .ASCII  /(8)/
004724  000  000                     .ASCII  <00><00>
004726  045  116  045       P.ACI:   .ASCII  /%N%/
004731  101  125  116                .ASCII  /AUN/
004734  111  124  045                .ASCII  /IT%/
004737  104  062  045                .ASCII  /D2%/
004742  101  040  104                .ASCII  /A D/
004745  122  117  120                .ASCII  /ROP/
004750  120  105  104                .ASCII  /PED/
004753  040  055  040                .ASCII  / - /
004756  000  000                     .ASCII  <00><00>
004760  045  101  125       P.ACK:   .ASCII  /%AU/
004763  123  105  122                .ASCII  /SER/
004766  040  103  117                .ASCII  / CO/
004771  115  115  101                .ASCII  /MMA/
004774  116  104  045                .ASCII  /ND%/
004777  116  000  000                .ASCII  /N/<00><00>
005002  045  101  103       P.ACL:   .ASCII  /%AC/
005005  117  116  106                .ASCII  /ONF/
005010  111  107  125                .ASCII  /IGU/
005013  122  101  124                .ASCII  /RAT/
005016  111  117  116                .ASCII  /ION/
005021  040  105  122                .ASCII  / ER/
005024  122  117  122                .ASCII  /ROR/
005027  045  116  000                .ASCII  /%N/<00>
005032  045  101  111       P.ACM:   .ASCII  /%AI/
005035  116  111  124                .ASCII  /NIT/
005040  040  105  122                .ASCII  / ER/
005043  122  117  122                .ASCII  /ROR/
005046  045  116  000                .ASCII  /%N/<00>
005051  000                         .ASCII  <00>
005052  045  101  124       P.ACN:   .ASCII  /%AT/
005055  122  101  116                .ASCII  /RAN/
005060  123  106  105                .ASCII  /SFE/
005063  122  040  114                .ASCII  /R L/
005066  111  115  111                .ASCII  /IMI/
005071  124  040  122                .ASCII  /T R/
005074  105  101  103                .ASCII  /EAC/
005077  110  105  104                .ASCII  /HED/
005102  045  116  000                .ASCII  /%N/<00>
005105  000                         .ASCII  <00>
005106  045  101  105       P.ACO:   .ASCII  /%AE/
005111  122  122  117                .ASCII  /RRO/
005114  122  040  114                .ASCII  /R L/
005117  111  115  111                .ASCII  /IMI/
005122  124  040  122                .ASCII  /T R/
005125  105  101  103                .ASCII  /EAC/
005130  110  105  104                .ASCII  /HED/
005133  045  116  000                .ASCII  /%N/<00>
005136  045  101  125       P.ACP:   .ASCII  /%AU/
```

```
005141  116  122  105                .ASCII  /NRE/
005144  103  117  126                .ASCII  /COV/
005147  105  122  101                .ASCII  /ERA/
005152  102  114  105                .ASCII  /BLE/
005155  040  104  122                .ASCII  / DR/
005160  111  126  105                .ASCII  /IVE/
005163  040  105  122                .ASCII  / ER/
005166  122  117  122                .ASCII  /ROR/
005171  045  116  000                .ASCII  /%N/<00>
005174  045  101  125  P.ACQ:        .ASCII  /%AU/
005177  116  122  105                .ASCII  /NRE/
005202  103  117  126                .ASCII  /COV/
005205  105  122  101                .ASCII  /ERA/
005210  102  114  105                .ASCII  /BLE/
005213  040  103  117                .ASCII  / CO/
005216  116  124  122                .ASCII  /NTR/
005221  117  114  114                .ASCII  /OLL/
005224  105  122  040                .ASCII  /ER /
005227  105  122  122                .ASCII  /ERR/
005232  117  122  045                .ASCII  /OR%/
005235  116  000  000                .ASCII  /N/<00><00>
005240  045  101  106  P.ACR:        .ASCII  /%AF/
005243  101  111  114                .ASCII  /AIL/
005246  105  104  040                .ASCII  /ED /
005251  124  117  040                .ASCII  /TO /
005254  103  117  115                .ASCII  /COM/
005257  105  040  117                .ASCII  /E O/
005262  116  114  111                .ASCII  /NLI/
005265  116  105  045                .ASCII  /NE%/
005270  116  000                     .ASCII  /N/<00>
005272  045  101  106  P.ACS:        .ASCII  /%AF/
005275  101  111  114                .ASCII  /AIL/
005300  105  104  040                .ASCII  /ED /
005303  124  117  040                .ASCII  /TO /
005306  101  103  103                .ASCII  /ACC/
005311  105  123  123                .ASCII  /ESS/
005314  040  105  111                .ASCII  / EI/
005317  124  110  105                .ASCII  /THE/
005322  122  040  106                .ASCII  /R F/
005325  111  122  123                .ASCII  /IRS/
005330  124  040  117                .ASCII  /T O/
005333  122  040  114                .ASCII  /R L/
005336  101  123  124                .ASCII  /AST/
005341  040  124  122                .ASCII  / TR/
005344  101  103  113                .ASCII  /ACK/
005347  040  104  125                .ASCII  / DU/
005352  122  111  116                .ASCII  /RIN/
005355  107  040  111                .ASCII  /G I/
005360  116  111  124                .ASCII  /NIT/
005363  045  116  000                .ASCII  /%N/<00>
005366  045  101  104  P.ACT:        .ASCII  /%AD/
005371  111  123  113                .ASCII  /ISK/
005374  040  127  122                .ASCII  / WR/
```

```
005377      111      124      105                    .ASCII  /ITE/
005402      040      120      122                    .ASCII  / PR/
005405      117      124      105                    .ASCII  /OTE/
005410      103      124      105                    .ASCII  /CTE/
005413      104      045      116                    .ASCII  /D%N/
005416      000      000                             .ASCII  <00><00>
005420      045      101      103          P.ACU:    .ASCII  /%AC/
005423      117      115      115                    .ASCII  /OMM/
005426      101      116      104                    .ASCII  /AND/
005431      040      124      111                    .ASCII  / TI/
005434      115      105      040                    .ASCII  /ME /
005437      117      125      124                    .ASCII  /OUT/
005442      045      116      000                    .ASCII  /%N/<00>
005445      000                                      .ASCII  <00>
005446   004760'                             P.ACJ:    .WORD   P.ACK
005450   005002'                                       .WORD   P.ACL
005452   005032'                                       .WORD   P.ACM
005454   005052'                                       .WORD   P.ACN
005456   005106'                                       .WORD   P.ACO
005460   005136'                                       .WORD   P.ACP
005462   005174'                                       .WORD   P.ACQ
005464   005240'                                       .WORD   P.ACR
005466   005272'                                       .WORD   P.ACS
005470   005366'                                       .WORD   P.ACT
005472   005420'                                       .WORD   P.ACU
005474      045      116      045          P.ACV:    .ASCII  /%N%/
005477      101      120      117                    .ASCII  /APO/
005502      127      105      122                    .ASCII  /WER/
005505      040      104      105                    .ASCII  / DE/
005510      114      101      131                    .ASCII  /LAY/
005513      040      055      040                    .ASCII  / - /
005516      127      101      111                    .ASCII  /WAI/
005521      124      111      116                    .ASCII  /TIN/
005524      107      000                             .ASCII  /G/<00>
005526      045      116      045          P.ACW:    .ASCII  /%N%/
005531      101      106      125                    .ASCII  /AFU/
005534      116      103      124                    .ASCII  /NCT/
005537      111      117      116                    .ASCII  /ION/
005542      101      114      040                    .ASCII  /AL /
005545      124      105      123                    .ASCII  /TES/
005550      124      040      123                    .ASCII  /T S/
005553      124      101      122                    .ASCII  /TAR/
005556      124      105      104                    .ASCII  /TED/
005561      000                                      .ASCII  <00>
005562      045      116      045          P.ACX:    .ASCII  /%N%/
005565      116      045      101                    .ASCII  /N%A/
005570      105      130      105                    .ASCII  /EXE/
005573      123      103      111                    .ASCII  /SCI/
005576      123      105      122                    .ASCII  /SER/
005601      040      123      124                    .ASCII  / ST/
005604      101      122      124                    .ASCII  /ART/
005607      105      104      045                    .ASCII  /ED%/
005612      116      000                             .ASCII  /N/<00>
```

```
005614  045  116  045  P.ACY:  .ASCII  /■N■/
005617  116  045  101         .ASCII  /N■A/
005622  125  116  124         .ASCII  /UNT/
005625  040  104  123         .ASCII  / DS/
005630  113  045  123         .ASCII  /K■S/
005633  070  045  101         .ASCII  /8■A/
005636  043  040  117         .ASCII  /# O/
005641  106  040  040         .ASCII  /F  /
005644  040  043  040         .ASCII  / # /
005647  102  131  124         .ASCII  /BYT/
005652  105  123  040         .ASCII  /ES /
005655  040  040  043         .ASCII  /  #/
005660  040  117  106         .ASCII  / OF/
005663  040  040  040         .ASCII  /   /
005666  040  043  040         .ASCII  / # /
005671  102  131  124         .ASCII  /BYT/
005674  105  123  000         .ASCII  /ES/<00>
005677  000                  .ASCII  <00>
005700  045  101  040  P.ACZ:  .ASCII  /■A /
005703  040  055  055         .ASCII  / --/
005706  110  101  122         .ASCII  /HAR/
005711  104  040  105         .ASCII  /D E/
005714  122  122  117         .ASCII  /RRO/
005717  122  123  055         .ASCII  /RS-/
005722  055  040  055         .ASCII  /- -/
005725  055  123  117         .ASCII  /-SO/
005730  106  124  040         .ASCII  /FT /
005733  105  122  122         .ASCII  /ERR/
005736  117  122  123         .ASCII  /ORS/
005741  055  055  000         .ASCII  /--/<00>
005744  045  116  045  P.ADA:  .ASCII  /■N■/
005747  101  040  043         .ASCII  /A #/
005752  040  040  040         .ASCII  /   /
005755  043  040  040         .ASCII  /#  /
005760  124  131  120         .ASCII  /TYP/
005763  105  040  040         .ASCII  /E  /
005766  122  105  101         .ASCII  /REA/
005771  104  123  040         .ASCII  /DS /
005774  040  040  040         .ASCII  /   /
005777  040  122  105         .ASCII  / RE/
006002  101  104  040         .ASCII  /AD /
006005  040  040  127         .ASCII  /  W/
006010  122  111  124         .ASCII  /RIT/
006013  105  123  040         .ASCII  /ES /
006016  040  040  127         .ASCII  /  W/
006021  122  111  124         .ASCII  /RIT/
006024  124  105  116         .ASCII  /TEN/
006027  000                  .ASCII  <00>
006030  045  101  040  P.ADB:  .ASCII  /■A /
006033  040  123  105         .ASCII  / SE/
006036  113  040  104         .ASCII  /K D/
006041  101  124  040         .ASCII  /AT /
006044  104  122  126         .ASCII  /DRV/
```

```
006047  040     110     123             .ASCII  / HS/
006052  124     040     123             .ASCII  /T S/
006055  105     113     040             .ASCII  /EK /
006060  104     101     124             .ASCII  /DAT/
006063  040     104     122             .ASCII  / DR/
006066  126     040     110             .ASCII  /V H/
006071  123     124     000             .ASCII  /ST/<00>
006074  045     116     045     P.ADC:  .ASCII  /%N%/
006077  101     055     055             .ASCII  /A--/
006102  055     040     055             .ASCII  /- -/
006105  055     055     040             .ASCII  /-- /
006110  055     055     055             .ASCII  /---/
006113  055     040     040             .ASCII  /-  /
006116  055     055     055             .ASCII  /---/
006121  055     055     040             .ASCII  /-- /
006124  040     055     055             .ASCII  / --/
006127  055     055     055             .ASCII  /---/
006132  055     055     055             .ASCII  /---/
006135  055     040     055             .ASCII  /- -/
006140  055     055     055             .ASCII  /---/
006143  055     055     040             .ASCII  /-- /
006146  040     055     055             .ASCII  / --/
006151  055     055     055             .ASCII  /-- /
006154  055     055     055             .ASCII  /---/
006157  055     000     000             .ASCII  /-/<00><00>
006162  045     101     040     P.ADD:  .ASCII  /%A /
006165  055     055     055             .ASCII  /---/
006170  040     055     055             .ASCII  / --/
006173  055     040     055             .ASCII  /- -/
006176  055     055     040             .ASCII  /-- /
006201  055     055     055             .ASCII  /---/
006204  040     055     055             .ASCII  / --/
006207  055     040     055             .ASCII  /- -/
006212  055     055     040             .ASCII  /-- /
006215  055     055     055             .ASCII  /---/
006220  040     055     055             .ASCII  / --/
006223  055     000     000             .ASCII  /-/<00><00>
006226  045     116     045     P.ADE:  .ASCII  /%N%/
006231  104     062     045             .ASCII  /D2%/
006234  104     064     045             .ASCII  /D4%/
006237  123     062     045             .ASCII  /S2%/
006242  124     000                     .ASCII  /T/<00>
006244  045     104     064     P.ADF:  .ASCII  /%D4/
006247  045     132     063             .ASCII  /%Z3/
006252  045     104     063             .ASCII  /%D3/
006255  045     101     054             .ASCII  /%A,/
006260  045     132     063             .ASCII  /%Z3/
006263  045     101     054             .ASCII  /%A,/
006266  045     132     063             .ASCII  /%Z3/
006271  000                             .ASCII  <00>
006272  045     104     064     P.ADG:  .ASCII  /%D4/
006275  045     104     064             .ASCII  /%D4/
006300  045     104     064             .ASCII  /%D4/
```

```
006303  045  104  064          .ASCII  /%D4/
006306  045  104  064          .ASCII  /%D4/
006311  045  104  064          .ASCII  /%D4/
006314  045  104  064          .ASCII  /%D4/
006317  045  104  064          .ASCII  /%D4/
006322  000  000               .ASCII  <00><00>
006324  045  116  045  P.ADH:   .ASCII  /%N%/
006327  101  040  056          .ASCII  /A ./
006332  040  040  040          .ASCII  /   /
006335  056  040  040          .ASCII  /.  /
006340  103  116  124          .ASCII  /CNT/
006343  122  040  040          .ASCII  /R  /
006346  040  040  040          .ASCII  /   /
006351  040  056  040          .ASCII  / . /
006354  040  056  056          .ASCII  / ../
006357  056  056  056          .ASCII  /.../
006362  056  056  056          .ASCII  /.../
006365  056  040  040          .ASCII  /.  /
006370  040  040  040          .ASCII  /   /
006373  040  056  040          .ASCII  / . /
006376  040  056  056          .ASCII  / ../
006401  056  056  056          .ASCII  /.../
006404  056  056  056          .ASCII  /.../
006407  056  000  000          .ASCII  /./<00><00>
006412  045  101  040  P.ADI:   .ASCII  /%A /
006415  040  040  056          .ASCII  /  ./
006420  040  040  040          .ASCII  /   /
006423  056  045  104          .ASCII  /.%D/
006426  064  045  101          .ASCII  /4%A/
006431  040  040  040          .ASCII  /   /
006434  056  040  040          .ASCII  /.  /
006437  040  056  040          .ASCII  / . /
006442  040  040  056          .ASCII  /  ./
006445  045  104  064          .ASCII  /%D4/
006450  045  101  040          .ASCII  /%A /
006453  040  040  056          .ASCII  /  ./
006456  000  000               .ASCII  <00><00>
006460  124  117  117  P.ADJ:   .ASCII  /TOO/
006463  040  115  101          .ASCII  / MA/
006466  116  131  040          .ASCII  /NY /
006471  125  116  111          .ASCII  /UNI/
006474  124  123  000          .ASCII  /TS/<00>
006477  000                    .ASCII  <00>
006500  116  117  124  P.ADK:   .ASCII  /NOT/
006503  040  105  116          .ASCII  / EN/
006506  117  125  107          .ASCII  /OUG/
006511  110  040  106          .ASCII  /H F/
006514  122  105  105          .ASCII  /REE/
006517  040  115  105          .ASCII  / ME/
006522  115  117  122          .ASCII  /MOR/
006525  131  040  106          .ASCII  /Y F/
006530  117  122  040          .ASCII  /OR /
006533  101  114  114          .ASCII  /ALL/
```

```
006536  117     103     101             .ASCII  /OCA/
006541  124     111     116             .ASCII  /TIN/
006544  107     040     122             .ASCII  /G R/
006547  105     101     104             .ASCII  /EAD/
006552  057     127     122             .ASCII  <57>/WR/
006555  111     124     105             .ASCII  /ITE/
006560  040     102     125             .ASCII  / BU/
006563  106     106     105             .ASCII  /FFE/
006566  122     123     000             .ASCII  /RS/<00>
006571  000                             .ASCII  <00>
006572  122     105     107     P.ADL:  .ASCII  /REG/
006575  111     123     124             .ASCII  /IST/
006600  105     122     040             .ASCII  /ER /
006603  105     130     111             .ASCII  /EXI/
006606  123     124     105             .ASCII  /STE/
006611  116     103     105             .ASCII  /NCE/
006614  040     124     105             .ASCII  / TE/
006617  123     124     040             .ASCII  /ST /
006622  106     101     111             .ASCII  /FAI/
006625  114     105     104             .ASCII  /LED/
006630  000     000                     .ASCII  <00><00>
006632  126     105     103     P.ADM:  .ASCII  /VEC/
006635  124     117     122             .ASCII  /TOR/
006640  040     124     105             .ASCII  / TE/
006643  123     124     040             .ASCII  /ST /
006646  106     101     111             .ASCII  /FAI/
006651  114     105     104             .ASCII  /LED/
006654  000     000                     .ASCII  <00><00>
006656  102     122     040     P.ADN:  .ASCII  /BR /
006661  114     105     126             .ASCII  /LEV/
006664  105     114     040             .ASCII  /EL /
006667  124     105     123             .ASCII  /TES/
006672  124     040     106             .ASCII  /T F/
006675  101     111     114             .ASCII  /AIL/
006700  105     104     000             .ASCII  /ED/<00>
006703  000                             .ASCII  <00>
006704  111     116     111     P.ADO:  .ASCII  /INI/
006707  124     040     123             .ASCII  /T S/
006712  105     121     125             .ASCII  /EQU/
006715  105     116     103             .ASCII  /ENC/
006720  105     040     106             .ASCII  /E F/
006723  101     111     114             .ASCII  /AIL/
006726  105     104     000             .ASCII  /ED/<00>
006731  000                             .ASCII  <00>
006732  106     101     124     P.ADP:  .ASCII  /FAT/
006735  101     114     040             .ASCII  /AL /
006740  103     117     116             .ASCII  /CON/
006743  124     122     117             .ASCII  /TRO/
006746  114     114     105             .ASCII  /LLE/
006751  122     040     105             .ASCII  /R E/
006754  122     122     117             .ASCII  /RRO/
006757  122     000     000             .ASCII  /R/<00><00>
006762  117     116     114     P.ADQ:  .ASCII  /ONL/
```

```
006765  111  116  105                 .ASCII  /INE/
006770  040  106  101                 .ASCII  / FA/
006773  111  114  105                 .ASCII  /ILE/
006776  104  000                      .ASCII  /D/<00>
007000  127  122  111       P.ADR:    .ASCII  /WRI/
007003  124  105  055                 .ASCII  /TE -/
007006  120  122  117                 .ASCII  /PRO/
007011  124  105  103                 .ASCII  /TEC/
007014  124  040  103                 .ASCII  /T C/
007017  117  116  106                 .ASCII  /ONF/
007022  114  111  103                 .ASCII  /LIC/
007025  124  000  000                 .ASCII  /T/<00><00>
007030  101  103  103       P.ADS:    .ASCII  /ACC/
007033  105  123  123                 .ASCII  /ESS/
007036  040  106  101                 .ASCII  / FA/
007041  111  114  105                 .ASCII  /ILE/
007044  104  000                      .ASCII  /D/<00>
007046  106  101  124       P.ADT:    .ASCII  /FAT/
007051  101  114  040                 .ASCII  /AL /
007054  111  057  117                 .ASCII  /I/<57>/O/
007057  040  105  122                 .ASCII  / ER/
007062  120  117  122                 .ASCII  /ROR/
007065  000                           .ASCII  <00>
007066  104  111  123       P.ADU:    .ASCII  /DIS/
007071  113  040  124                 .ASCII  /K T/
007074  131  120  105                 .ASCII  /YPE/
007077  040  125  116                 .ASCII  / UN/
007102  113  116  117                 .ASCII  /KNO/
007105  127  116  040                 .ASCII  /WN /
007110  124  117  040                 .ASCII  /TO /
007113  105  130  105                 .ASCII  /EXE/
007116  122  103  111                 .ASCII  /RCI/
007121  123  105  122                 .ASCII  /SER/
007124  000  000                      .ASCII  <00><00>
007126  106  101  111       P.ADV:    .ASCII  /FAI/
007131  114  105  104                 .ASCII  /LED/
007134  040  124  117                 .ASCII  / TO/
007137  040  123  105                 .ASCII  / SE/
007142  116  104  040                 .ASCII  /ND /
007145  123  105  124                 .ASCII  /SET/
007150  055  103  117                 .ASCII  / CO/
007153  116  124  122                 .ASCII  /NTR/
007156  117  114  114                 .ASCII  /OLL/
007161  105  122  055                 .ASCII  /ER -/
007164  103  110  101                 .ASCII  /CHA/
007167  122  101  103                 .ASCII  /RAC/
007172  124  105  122                 .ASCII  /TER/
007175  111  123  124                 .ASCII  /IST/
007200  111  103  123                 .ASCII  /ICS/
007203  040  103  117                 .ASCII  / CO/
007206  115  115  101                 .ASCII  /MMA/
007211  116  104  000                 .ASCII  /ND <00>
007214  123  105  124       P.ADW:    .ASCII  /SET/
```

```
007217  055  103  117          .ASCII  / CO/
007222  116  124  122          .ASCII  /NTR/
007225  117  114  114          .ASCII  /OLL/
007230  105  122  055          .ASCII  /ER -/
007233  103  110  101          .ASCII  /CHA/
007236  122  101  103          .ASCII  /RAC/
007241  124  105  122          .ASCII  /TER/
007244  111  123  124          .ASCII  /IST/
007247  111  103  123          .ASCII  /ICS/
007252  040  122  105          .ASCII  / RE/
007255  123  120  117          .ASCII  /SPO/
007260  116  123  105          .ASCII  /NSE/
007263  040  110  101          .ASCII  / HA/
007266  123  040  102          .ASCII  /S B/
007271  101  104  040          .ASCII  /AD /
007274  105  116  104          .ASCII  /END/
007277  103  117  104          .ASCII  /COD/
007302  105  040  117          .ASCII  /E O/
007305  122  040  106          .ASCII  /R F/
007310  114  101  107          .ASCII  /LAG/
007313  123  040  111          .ASCII  /S I/
007316  116  040  105          .ASCII  /N E/
007321  122  122  117          .ASCII  /RRO/
007324  122  000                .ASCII  /R/<00>
007326  106  101  111  P.ADX:   .ASCII  /FAI/
007331  114  105  104          .ASCII  /LED/
007334  040  124  117          .ASCII  / TO/
007337  040  123  105          .ASCII  / SE/
007342  116  104  040          .ASCII  /ND /
007345  117  116  055          .ASCII  /ON -/
007350  114  111  116          .ASCII  /LIN/
007353  105  040  103          .ASCII  /E C/
007356  117  115  115          .ASCII  /OMM/
007361  101  116  104          .ASCII  /AND/
007364  000  000                .ASCII  <00><00>
007366  117  116  055  P.AD1:   .ASCII  /ON -/
007371  114  111  116          .ASCII  /LIN/
007374  105  040  122          .ASCII  /E R/
007377  105  123  120          .ASCII  /ESP/
007402  117  116  123          .ASCII  /ONS/
007405  105  040  110          .ASCII  /E H/
007410  101  123  040          .ASCII  /AS /
007413  102  101  104          .ASCII  /BAD/
007416  040  105  116          .ASCII  / EN/
007421  104  103  117          .ASCII  /DCO/
007424  104  105  000          .ASCII  DE/<00>
007427  000                    .ASCII  <00>
007430  117  116  055  P.AD2:   .ASCII  /ON -/
007433  114  111  116          .ASCII  /LIN/
007436  105  040  122          .ASCII  /E R/
007441  105  123  120          .ASCII  /ESP/
007444  117  116  123          .ASCII  /ONS/
007447  105  040  110          .ASCII  /E H/
```

```
007452  101  123  040                .ASCII  /AS /
007455  125  116  113                .ASCII  /UNK/
007460  116  117  127                .ASCII  /NOW/
007463  116  040  104                .ASCII  /N D/
007466  105  126  111                .ASCII  /EVI/
007471  103  105  000                .ASCII  /CE/<00>
007474  111  057  117    P.AEA:      .ASCII  /I/<57>/O/
007477  040  122  105                .ASCII  / RE/
007502  121  125  105                .ASCII  /QUE/
007505  123  124  040                .ASCII  /ST /
007510  106  101  111                .ASCII  /FAI/
007513  114  105  104                .ASCII  /LED/
007516  000  000                     .ASCII  <00><00>
007520  045  101  115    P.AEB:      .ASCII  /%AM/
007523  117  122  105                .ASCII  /ORE/
007526  040  124  110                .ASCII  / TH/
007531  101  116  040                .ASCII  /AN /
007534  045  104  062                .ASCII  /%D2/
007537  045  101  040                .ASCII  /%A /
007542  125  116  111                .ASCII  /UNI/
007545  124  123  040                .ASCII  /TS /
007550  123  120  105                .ASCII  /SPE/
007553  103  111  106                .ASCII  /CIF/
007556  111  105  104                .ASCII  /IED/
007561  000                         .ASCII  <00>
007562  045  101  052    P.AEC:      .ASCII  /%A*/
007565  040  116  117                .ASCII  / NO/
007570  040  122  105                .ASCII  / RE/
007573  123  120  117                .ASCII  /SPO/
007576  116  123  105                .ASCII  /NSE/
007601  040  101  124                .ASCII  / AT/
007604  040  101  104                .ASCII  / AD/
007607  104  122  105                .ASCII  /DRE/
007612  123  123  040                .ASCII  /SS /
007615  045  117  066                .ASCII  /%O6/
007620  000  000                     .ASCII  <00><00>
007622  045  101  052    P.AED:      .ASCII  /%A*/
007625  040  111  116                .ASCII  / IN/
007630  103  117  122                .ASCII  /COR/
007633  122  105  103                .ASCII  /REC/
007636  124  040  102                .ASCII  /T B/
007641  122  040  114                .ASCII  /R L/
007644  105  126  105                .ASCII  /EVE/
007647  114  040  106                .ASCII  /L F/
007652  117  122  040                .ASCII  /OR /
007655  104  122  111                .ASCII  /DRI/
007660  126  105  040                .ASCII  /VE /
007663  045  117  066                .ASCII  /%O6/
007666  000  000                     .ASCII  <00><00>
007670  045  101  052    P.AEE:      .ASCII  /%A*/
007673  040  123  124                .ASCII  / ST/
007676  105  120  040                .ASCII  /EP /
007701  045  104  061                .ASCII  /%D1/
```

```
007704  045  101  040            .ASCII  /■A /
007707  122  105  101            .ASCII  /REA/
007712  104  040  105            .ASCII  /D E/
007715  122  122  117            .ASCII  /RRO/
007720  122  000                 .ASCII  /R/<00>
007722  045  101  052  P.AEF:    .ASCII  /■A*/
007725  040  102  101            .ASCII  / BA/
007730  104  040  123            .ASCII  /D S/
007733  101  040  103            .ASCII  /A C/
007736  117  104  105            .ASCII  /ODE/
007741  040  106  122            .ASCII  / FR/
007744  117  115  040            .ASCII  /OM /
007747  104  122  111            .ASCII  /DRI/
007752  126  105  040            .ASCII  /VE /
007755  045  117  066            .ASCII  /■06/
007760  000  000                 .ASCII  <00><00>
007762  045  101  052  P.AEG:    .ASCII  /■A*/
007765  040  104  111            .ASCII  / DI/
007770  123  113  045            .ASCII  /SK■/
007773  104  062  045            .ASCII  /D2■/
007776  101  040  127            .ASCII  /A W/
010001  105  116  124            .ASCII  /ENT/
010004  040  117  106            .ASCII  / OF/
010007  106  114  111            .ASCII  /FLI/
010012  116  105  000            .ASCII  /NE/<00>
010015  000                      .ASCII  <00>
010016  045  101  052  P.AEH:    .ASCII  /■A*/
010021  040  104  122            .ASCII  / DR/
010024  111  126  105            .ASCII  /IVE/
010027  040  045  117            .ASCII  / ■O/
010032  066  045  101            .ASCII  /6■A/
010035  040  116  117            .ASCII  / NO/
010040  124  040  120            .ASCII  /T P/
010043  122  117  103            .ASCII  /ROC/
010046  105  123  123            .ASCII  /ESS/
010051  111  116  107            .ASCII  /ING/
010054  040  103  117            .ASCII  / CO/
010057  115  115  101            .ASCII  /MMA/
010062  116  104  040            .ASCII  /ND /
010065  120  101  103            .ASCII  /PAC/
010070  113  105  124            .ASCII  /KET/
010073  123  000  000            .ASCII  /S/<00><00>
010076  045  101  052  P.AEI:    .ASCII  /■A*/
010101  040  104  111            .ASCII  / DI/
010104  123  113  045            .ASCII  /SK■/
010107  104  062  045            .ASCII  /D2■/
010112  101  040  127            .ASCII  /A W/
010115  105  116  124            .ASCII  /ENT/
010120  040  124  117            .ASCII  / TO/
010123  040  124  110            .ASCII  / TH/
010126  105  040  042            .ASCII  /E "/
010131  101  126  101            .ASCII  /AVA/
010134  111  114  101            .ASCII  /ILA/
```

```
010137  102  114  105                .ASCII  /BLE/
010142  042  040  123                .ASCII  /" S/
010145  124  101  124                .ASCII  /TAT/
010150  105  000                     .ASCII  /E/<00>
010152  040  055  040     P.AEJ:     .ASCII  / - /
010155  125  116  122                .ASCII  /UNR/
010160  105  103  117                .ASCII  /ECO/
010163  107  116  111                .ASCII  /GNI/
010166  132  105  104                .ASCII  /ZED/
010171  040  115  105                .ASCII  / ME/
010174  123  123  101                .ASCII  /SSA/
010177  107  105  040                .ASCII  /GE /
010202  124  131  120                .ASCII  /TYP/
010205  105  000  000                .ASCII  /E/<00><00>
010210  040  055  040     P.AEK:     .ASCII  / - /
010213  125  116  122                .ASCII  /UNR/
010216  105  103  117                .ASCII  /ECO/
010221  107  116  111                .ASCII  /GNI/
010224  132  105  104                .ASCII  /ZED/
010227  040  103  117                .ASCII  / CO/
010232  116  116  105                .ASCII  /NNE/
010235  103  124  111                .ASCII  /CTI/
010240  117  116  040                .ASCII  /ON /
010243  111  104  000                .ASCII  /ID/<00>
010246  040  055  040     P.AEL:     .ASCII  / - /
010251  125  116  122                .ASCII  /UNR/
010254  105  103  117                .ASCII  /ECO/
010257  107  116  111                .ASCII  /GNI/
010262  132  105  104                .ASCII  /ZED/
010265  040  122  105                .ASCII  / RE/
010270  124  125  122                .ASCII  /TUR/
010273  116  040  115                .ASCII  /N M/
010276  105  123  123                .ASCII  /ESS/
010301  101  107  105                .ASCII  /AGE/
010304  000  000                     .ASCII  <00><00>
010306  040  055  040     P.AEM:     .ASCII  / - /
010311  125  116  122                .ASCII  /UNR/
010314  105  103  117                .ASCII  /ECO/
010317  107  116  111                .ASCII  /GNI/
010322  132  105  104                .ASCII  /ZED/
010325  040  122  105                .ASCII  / RE/
010330  124  125  122                .ASCII  /TUR/
010333  116  040  120                .ASCII  /N P/
010336  101  103  113                .ASCII  /ACK/
010341  105  124  000                .ASCII  /ET/<00>
010344  040  055  040     P.AEN:     .ASCII  /   /
010347  125  116  122                .ASCII  /UNR/
010352  105  103  117                .ASCII  /ECO/
010355  107  116  111                .ASCII  /GNI/
010360  132  105  104                .ASCII  /ZED/
010363  040  103  122                .ASCII  / CR/
010366  116  000                     .ASCII  /N/<00>
010370  040  055  040     P.AEO:     .ASCII  / - /
```

```
010373  125  116  122                .ASCII  /UNR/
010376  105  103  117                .ASCII  /ECO/
010401  107  116  111                .ASCII  /GNI/
010404  132  105  104                .ASCII  /ZED/
010407  040  117  120                .ASCII  / OP/
010412  103  117  104                .ASCII  /COD/
010415  105  000  000                .ASCII  /E/<00><00>
010420  040  055  040     P.AEP:     .ASCII  / - /
010423  115  123  103                .ASCII  /MSC/
010426  120  040  123                .ASCII  /P S/
010431  124  101  124                .ASCII  /TAT/
010434  125  123  040                .ASCII  /US /
010437  103  117  104                .ASCII  /COD/
010442  105  040  105                .ASCII  /E E/
010445  122  122  000                .ASCII  /RR/<00>
010450  040  055  040     P.AEQ:     .ASCII  / - /
010453  104  125  120                .ASCII  /DUP/
010456  040  123  124                .ASCII  / ST/
010461  101  124  125                .ASCII  /ATU/
010464  123  040  103                .ASCII  /S C/
010467  117  104  105                .ASCII  /ODE/
010472  040  105  122                .ASCII  / ER/
010475  122  000  000                .ASCII  /R/<00><00>
010500  040  055  040     P.AER:     .ASCII  / - /
010503  125  116  122                .ASCII  /UNR/
010506  105  103  117                .ASCII  /ECO/
010511  107  116  111                .ASCII  /GNI/
010514  132  105  104                .ASCII  /ZED/
010517  040  123  124                .ASCII  / ST/
010522  101  124  125                .ASCII  /ATU/
010525  123  040  103                .ASCII  /S C/
010530  117  104  105                .ASCII  /ODE/
010533  000                         .ASCII  <00>
010534  040  055  040     P.AES:     .ASCII  / - /
010537  114  102  116                .ASCII  /LBN/
010542  040  110  117                .ASCII  / HO/
010545  123  124  040                .ASCII  /ST /
010550  103  117  115                .ASCII  /COM/
010553  120  101  122                .ASCII  /PAR/
010556  105  040  105                .ASCII  /E E/
010561  122  122  000                .ASCII  /RR/<00>
010564  040  055  040     P.AET:     .ASCII  /  /
010567  104  102  116                .ASCII  /DBN/
010572  040  110  117                .ASCII  / HO/
010575  123  124  040                .ASCII  /ST /
010600  103  117  115                .ASCII  /COM/
010603  120  101  122                .ASCII  /PAR/
010606  105  040  105                .ASCII  /E E/
010611  122  122  000                .ASCII  /RR/<00>
010614  040  055  040     P.AEU:     .ASCII  / - /
010617  125  116  101                .ASCII  /UNA/
010622  102  114  105                .ASCII  /BLE/
010625  040  124  117                .ASCII  / TO/
```

```
010630  040     114     117             .ASCII  / LO/
010633  101     104     040             .ASCII  /AD /
010636  104     125     120             .ASCII  /DUP/
010641  040     115     105             .ASCII  / ME/
010644  104     111     101             .ASCII  /DIA/
010647  000                             .ASCII  <00>
010650  040     055     040     P.AEV:  .ASCII  / - /
010653  105     122     122             .ASCII  /ERR/
010656  040     111     116             .ASCII  / IN/
010661  040     104     125             .ASCII  / DU/
010664  120     040     120             .ASCII  /P P/
010667  113     124     040             .ASCII  /KT /
010672  127     110     105             .ASCII  /WHE/
010675  116     040     125             .ASCII  /N U/
010700  123     111     116             .ASCII  /SIN/
010703  107     040     105             .ASCII  /G C/
010706  124     114     122             .ASCII  /TLR/
010711  040     114     103             .ASCII  / LC/
010714  040     120     122             .ASCII  / PR/
010717  117     107     000             .ASCII  /OG/<00>
010722  045     101     052     P.AEW:  .ASCII  /%A*/
010725  040     104     111             .ASCII  / DI/
010730  123     113     045             .ASCII  /SK%/
010733  104     062     000             .ASCII  /D2/<00>
010736  045     101     111     P.AEY:  .ASCII  /%AI/
010741  116     126     101             .ASCII  /NVA/
010744  114     111     104             .ASCII  /LID/
010747  040     103     117             .ASCII  / CO/
010752  115     115     101             .ASCII  /MMA/
010755  116     104     000             .ASCII  /ND/<00>
010760  045     101     103     P.AEZ:  .ASCII  /%AC/
010763  117     115     115             .ASCII  /OMM/
010766  101     116     104             .ASCII  /AND/
010771  040     101     102             .ASCII  / AB/
010774  117     122     124             .ASCII  /ORT/
010777  105     104     000             .ASCII  /ED/<00>
011002  045     101     125     P.AFA:  .ASCII  /%AU/
011005  116     111     124             .ASCII  /NIT/
011010  040     117     106             .ASCII  / OF/
011013  106     114     111             .ASCII  /FLI/
011016  116     105     000             .ASCII  /NE/<00>
011021  000                             .ASCII  <00>
011022  045     101     124     P.AFB:  .ASCII  /%AT/
011025  122     101     116             .ASCII  /RAN/
011030  123     111     124             .ASCII  /SIT/
011033  111     117     116             .ASCII  /ION/
011036  040     124     117             .ASCII  / TO/
011041  040     101     126             .ASCII  / AV/
011044  101     111     114             .ASCII  /AIL/
011047  101     102     114             .ASCII  /ABL/
011052  105     040     123             .ASCII  /E S/
011055  124     101     124             .ASCII  /TAT/
011060  105     000                     .ASCII  /E/<00>
```

```
011062  045     101     115     P.AFC:  .ASCII  /■AM/
011065  105     104     111             .ASCII  /EDI/
011070  101     040     106             .ASCII  /A F/
011073  117     122     115             .ASCII  /ORM/
011076  101     124     040             .ASCII  /AT /
011101  105     122     122             .ASCII  /ERR/
011104  117     122     000             .ASCII  /OR/<00>
011107  000                             .ASCII  <00>
011110  045     101     127     P.AFD:  .ASCII  /■AW/
011113  122     111     124             .ASCII  /RIT/
011116  105     055     120             .ASCII  /E P/
011121  122     117     124             .ASCII  /ROT/
011124  105     103     124             .ASCII  /ECT/
011127  105     104     000             .ASCII  /ED/<00>
011132  045     101     104     P.AFE:  .ASCII  /■AD/
011135  105     126     111             .ASCII  /EVI/
011140  103     105     040             .ASCII  /CE /
011143  103     117     115             .ASCII  /COM/
011146  120     101     122             .ASCII  /PAR/
011151  105     040     105             .ASCII  /E E/
011154  122     122     117             .ASCII  /RRO/
011157  122     000     000             .ASCII  /R/<00><00>
011162  045     101     104     P.AFF:  .ASCII  /■AD/
011165  101     124     101             .ASCII  /ATA/
011170  040     105     122             .ASCII  / ER/
011173  122     117     122             .ASCII  /ROR/
011176  000     000                     .ASCII  <00><00>
011200  045     101     110     P.AFG:  .ASCII  /■AH/
011203  117     123     124             .ASCII  /OST/
011206  040     102     125             .ASCII  / BU/
011211  106     106     105             .ASCII  /FFE/
011214  122     040     101             .ASCII  /R A/
011217  103     103     105             .ASCII  /CCE/
011222  123     123     040             .ASCII  /SS /
011225  105     122     122             .ASCII  /ERR/
011230  117     122     000             .ASCII  /OR/<00>
011233  000                             .ASCII  <00>
011234  045     101     103     P.AFH:  .ASCII  /■AC/
011237  117     116     124             .ASCII  /ONT/
011242  122     117     114             .ASCII  /ROL/
011245  114     105     122             .ASCII  /LER/
011250  040     105     122             .ASCII  / ER/
011253  122     117     122             .ASCII  /ROR/
011256  000     000                     .ASCII  <00><00>
011260  045     101     104     P.AFI:  .ASCII  /■AD/
011263  122     111     126             .ASCII  /RIV/
011266  105     040     105             .ASCII  /E E/
011271  122     122     117             .ASCII  /RRO/
011274  122     000                     .ASCII  /R/<00>
011276  045     101     115     P.AFJ:  .ASCII  /■AM/
011301  105     123     123             .ASCII  /ESS/
011304  101     107     105             .ASCII  /AGE/
011307  040     106     122             .ASCII  / FR/
```

```
011312      117     115     040                 .ASCII  /OM /
011315      111     116     124                 .ASCII  /INT/
011320      105     122     116                 .ASCII  /ERN/
011323      101     114     040                 .ASCII  /AL /
011326      104     111     101                 .ASCII  /DIA/
011331      107     116     117                 .ASCII  /GNO/
011334      123     124     111                 .ASCII  /STI/
011337      103     123     000                 .ASCII  /CS/<00>
011342      045     101     110     P.AFK:      .ASCII  /%AH/
011345      117     123     124                 .ASCII  /OST/
011350      040     103     117                 .ASCII  / CO/
011353      115     120     101                 .ASCII  /MPA/
011356      122     105     040                 .ASCII  /RE /
011361      105     122     122                 .ASCII  /ERR/
011364      117     122     000                 .ASCII  /OR/<00>
011367      000                                 .ASCII  <00>
011370      045     101     103     P.AFL:      .ASCII  /%AC/
011373      117     115     115                 .ASCII  /OMM/
011376      101     116     104                 .ASCII  /AND/
011401      040     124     111                 .ASCII  / TI/
011404      115     105     117                 .ASCII  /MEO/
011407      125     124     000                 .ASCII  /UT/<00>
011412  010736'                     P.AEX:      .WORD   P.AEY
011414  010760'                                 .WORD   P.AEZ
011416  011002'                                 .WORD   P.AFA
011420  011022'                                 .WORD   P.AFB
011422  011062'                                 .WORD   P.AFC
011424  011110'                                 .WORD   P.AFD
011426  011132'                                 .WORD   P.AFE
011430  011162'                                 .WORD   P.AFF
011432  011200'                                 .WORD   P.AFG
011434  011234'                                 .WORD   P.AFH
011436  011260'                                 .WORD   P.AFI
011440  011276'                                 .WORD   P.AFJ
011442  011342'                                 .WORD   P.AFK
011444  011370'                                 .WORD   P.AFL
011446      045     101     105     P.AFM:      .ASCII  /%AE/
011451      122     122     117                 .ASCII  /RRO/
011454      122     040     114                 .ASCII  /R L/
011457      117     107     040                 .ASCII  /OG /
011462      115     105     123                 .ASCII  /MES/
011465      123     101     107                 .ASCII  /SAG/
011470      105     040     122                 .ASCII  /E R/
011473      105     103     105                 .ASCII  /ECE/
011476      111     126     105                 .ASCII  /IVE/
011501      104     072     045                 .ASCII  /D:%/
011504      116     000                         .ASCII  /N/<00>
011506      045     101     052     P.AFO:      .ASCII  /%A*/
011511      040     103     117                 .ASCII  / CO/
011514      116     124     122                 .ASCII  /NTR/
011517      117     114     114                 .ASCII  /OLL/
011522      105     122     040                 .ASCII  /ER /
011525      105     122     122                 .ASCII  /ERR/
```

```
011530  117  122  045                .ASCII  /OR*/
011533  116  000  000                .ASCII  /N/<00><00>
011536  045  101  052  P.AFP:        .ASCII  /*A*/
011541  040  110  117                .ASCII  / HO/
011544  123  124  040                .ASCII  /ST /
011547  115  105  115                .ASCII  /MEM/
011552  117  122  131                .ASCII  /ORY/
011555  040  101  103                .ASCII  / AC/
011560  103  105  123                .ASCII  /CES/
011563  123  040  105                .ASCII  /S E/
011566  122  122  117                .ASCII  /RRO/
011571  122  045  116                .ASCII  /R*N/
011574  000  000                     .ASCII  <00><00>
011576  045  101  052  P.AFQ:        .ASCII  /*A*/
011601  040  104  111                .ASCII  / DI/
011604  123  113  045                .ASCII  /SK*/
011607  104  062  045                .ASCII  /D2*/
011612  101  040  055                .ASCII  /A -/
011615  040  104  111                .ASCII  / DI/
011620  123  113  040                .ASCII  /SK /
011623  124  122  101                .ASCII  /TRA/
011626  116  123  106                .ASCII  /NSF/
011631  105  122  040                .ASCII  /ER /
011634  105  122  122                .ASCII  /ERR/
011637  117  122  045                .ASCII  /OR*/
011642  116  000                     .ASCII  /N/<00>
011644  045  101  052  P.AFR:        .ASCII  /*A*/
011647  040  104  111                .ASCII  / DI/
011652  123  113  045                .ASCII  /SK*/
011655  104  062  045                .ASCII  /D2*/
011660  101  040  055                .ASCII  /A -/
011663  040  042  123                .ASCII  / "S/
011666  124  101  116                .ASCII  /TAN/
011671  104  101  122                .ASCII  /DAR/
011674  104  040  104                .ASCII  /D D/
011677  111  123  113                .ASCII  /ISK/
011702  040  111  116                .ASCII  / IN/
011705  124  105  122                .ASCII  /TER/
011710  103  117  116                .ASCII  /CON/
011713  116  105  103                .ASCII  /NEC/
011716  124  042  040                .ASCII  /T" /
011721  105  122  122                .ASCII  /ERR/
011724  117  122  045                .ASCII  /OR*/
011727  116  000  000                .ASCII  /N/<00><00>
011732  045  101  052  P.AFS:        .ASCII  /*A*/
011735  040  104  111                .ASCII  / D1/
011740  123  113  045                .ASCII  /SK*/
011743  104  062  045                .ASCII  /D2*/
011746  101  040  055                .ASCII  /A -/
011751  040  042  123                .ASCII  / "S/
011754  115  101  114                .ASCII  /MAL/
011757  114  040  104                .ASCII  /L D/
011762  111  123  113                .ASCII  /ISK/
```

```
011765     042     040     105                  .ASCII  /" E/
011770     122     122     117                  .ASCII  /RRO/
011773     122     045     116                  .ASCII  /R%N/
011776     000     000                          .ASCII  <00><00>
012000  011506'                         P.AFN:  .WORD   P.AFO
012002  011536'                                 .WORD   P.AFP
012004  011576'                                 .WORD   P.AFQ
012006  011644'                                 .WORD   P.AFR
012010  011732'                                 .WORD   P.AFS
012012     045     116     045         P.AFT:  .ASCII  /%N%/
012015     101     052     040                 .ASCII  /A* /
012020     123     101     072                 .ASCII  /SA:/
012023     040     045     117                 .ASCII  / %O/
012026     066     000                         .ASCII  /6/<00>
012030     045     116     045         P.AFU:  .ASCII  /%N%/
012033     101     052     040                 .ASCII  /A* /
012036     123     124     101                 .ASCII  /STA/
012041     124     125     123                 .ASCII  /TUS/
012044     040     103     117                 .ASCII  / CO/
012047     104     105     072                 .ASCII  /DE:/
012052     040     045     117                 .ASCII  / %O/
012055     062     000     000                 .ASCII  /2/<00><00>
012060     045     117     064         P.AFV:  .ASCII  /%04/
012063     000                                 .ASCII  <00>
012064     045     116     045         P.AFW:  .ASCII  /%N%/
012067     101     052     040                 .ASCII  /A* /
012072     123     125     102                 .ASCII  /SUB/
012075     137     103     117                 .ASCII  /_CO/
012100     104     105     072                 .ASCII  /DE:/
012103     040     000     000                 .ASCII  / /<00><00>
012106     045     116     045         P.AFX:  .ASCII  /%N%/
012111     101     052     040                 .ASCII  /A* /
012114     103     117     115                 .ASCII  /COM/
012117     115     101     116                 .ASCII  /MAN/
012122     104     072     040                 .ASCII  /D: /
012125     000                                 .ASCII  <00>
012126     045     101     122         P.AFY:  .ASCII  /%AR/
012131     105     101     104                 .ASCII  /EAD/
012134     000     000                         .ASCII  <00><00>
012136     045     101     127         P.AFZ:  .ASCII  /%AW/
012141     122     111     124                 .ASCII  /RIT/
012144     105     000                         .ASCII  /E/<00>
012146     045     101     055         P.AGA:  .ASCII  /%A-/
012151     103     117     115                 .ASCII  /COM/
012154     120     101     122                 .ASCII  /PAR/
012157     105     000     000                 .ASCII  /E/<00><00>
012162     045     101     117         P.AGB:  .ASCII  /%AO/
012165     116     114     111                 .ASCII  /NLI/
012170     116     105     000                 .ASCII  /NE/<00>
012173     000                                 .ASCII  <00>
012174     045     101     101         P.AGC:  .ASCII  /%AA/
012177     103     103     105                 .ASCII  /CCE/
012202     123     123     000                 .ASCII  /SS/<00>
```

M6


```
012205  000                             .ASCII  <00>
012206  045     117     063     P.AGD:  .ASCII  /%03/
012211  000                             .ASCII  <00>
012212  045     116     045     P.AGE:  .ASCII  /%N%/
012215  101     052     040             .ASCII  /A* /
012220  102     101     104             .ASCII  /BAD/
012223  040     102     114             .ASCII  / BL/
012226  117     103     113             .ASCII  /OCK/
012231  040     050     110             .ASCII  / (H/
012234  157     163     164             .ASCII  /ost/
012237  040     162     145             .ASCII  / re/
012242  160     154     141             .ASCII  /pla/
012245  143     145     141             .ASCII  /cea/
012250  142     154     145             .ASCII  /ble/
012253  051     072     040             .ASCII  /): /
012256  045     104     065             .ASCII  /%D5/
012261  045     101     056             .ASCII  /%A./
012264  040     050     117             .ASCII  / (O/
012267  103     124     040             .ASCII  /CT /
012272  045     117     066             .ASCII  /%06/
012275  045     101     051             .ASCII  /%A)/
012300  000     000                     .ASCII  <00><00>
012302  045     116     045     P.AGF:  .ASCII  /%N%/
012305  101     052     040             .ASCII  /A* /
012310  061     163     164             .ASCII  /1st/
012313  040     102     101             .ASCII  / BA/
012316  104     040     102             .ASCII  /D B/
012321  114     117     103             .ASCII  /LOC/
012324  113     040     050             .ASCII  /K (/
012327  110     157     163             .ASCII  /Hos/
012332  164     040     162             .ASCII  /t r/
012335  145     160     154             .ASCII  /epl/
012340  141     143     145             .ASCII  /ace/
012343  141     142     154             .ASCII  /abl/
012346  145     051     072             .ASCII  /e):/
012351  040     045     104             .ASCII  / %D/
012354  065     045     101             .ASCII  /5%A/
012357  056     040     050             .ASCII  /. (/
012362  117     103     124             .ASCII  /OCT/
012365  040     045     117             .ASCII  / %O/
012370  066     045     101             .ASCII  /6%A/
012373  051     000     000             .ASCII  /)/<00><00>
012376  045     116     045     P.AGG:  .ASCII  /%N%/
012401  101     052     040             .ASCII  /A* /
012404  102     101     104             .ASCII  /BAD/
012407  040     102     114             .ASCII  / BL/
012412  117     103     113             .ASCII  /OCK/
012415  040     122     105             .ASCII  / RE/
012420  120     117     122             .ASCII  /POR/
012423  124     105     104             .ASCII  /TED/
012426  040     050     122             .ASCII  / (R/
012431  145     160     154             .ASCII  /epl/
012434  141     143     145             .ASCII  /ace/
```

N6


```
012457  144  051  072                .ASCII  /d):/
012462  040  045  104                .ASCII  / %D/
012465  045  101  056                .ASCII  /%A./
012470  040  050  117                .ASCII  / (O/
012473  103  124  040                .ASCII  /CT /
012476  045  117  066                .ASCII  /%O6/
012481  045  101  051                .ASCII  /%A)/
012464  000  000                     .ASCII  <00><00>
012466  045  116  045  P.AGH:        .ASCII  /%N%/
012471  101  052  040                .ASCII  /A* /
012474  114  102  116                .ASCII  /LBN/
012477  072  040  045                .ASCII  /: %/
012502  104  065  045                .ASCII  /D5%/
012505  101  056  040                .ASCII  /A. /
012510  050  117  103                .ASCII  /(OC/
012513  124  040  045                .ASCII  /T %/
012516  117  066  045                .ASCII  /O6%/
012521  101  051  000                .ASCII  /A)/<00>
012524  045  116  045  P.AGI:        .ASCII  /%N%/
012527  101  052  040                .ASCII  /A* /
012532  120  102  116                .ASCII  /PBN/
012535  072  040  045                .ASCII  /: %/
012540  104  065  045                .ASCII  /D5%/
012543  101  056  040                .ASCII  /A. /
012546  050  117  103                .ASCII  /(OC/
012551  124  040  045                .ASCII  /T %/
012554  117  066  045                .ASCII  /O6%/
012557  101  051  000                .ASCII  /A)/<00>
012562  045  116  045  P.AGJ:        .ASCII  /%N%/
012565  101  052  040                .ASCII  /A* /
012570  114  102  116                .ASCII  /LBN/
012573  072  040  050                .ASCII  /: (/
012576  122  105  101                .ASCII  /REA/
012601  104  051  040                .ASCII  /D) /
012604  045  104  065                .ASCII  /%D5/
012607  045  101  056                .ASCII  /%A./
012612  040  050  117                .ASCII  / (O/
012615  103  124  040                .ASCII  /CT /
012620  045  117  066                .ASCII  /%O6/
012623  045  101  051                .ASCII  /%A)/
012626  000  000                     .ASCII  <00><00>
012630  045  116  045  P.AGK:        .ASCII  /%N%/
012633  101  052  040                .ASCII  /A* /
012636  114  102  116                .ASCII  /LBN/
012641  072  040  050                .ASCII  /: (/
012644  127  122  111                .ASCII  /WRI/
012647  124  105  051                .ASCII  /TE)/
012652  040  045  104                .ASCII  / %D/
012655  065  045  101                .ASCII  /5%A/
012660  056  040  050                .ASCII  /. (/
012663  117  103  124                .ASCII  /OCT/
012666  040  045  117                .ASCII  / %O/
012671  066  045  101                .ASCII  /6%A/
```

```
012674  051  000                        .ASCII  /)/<00>
012676  045  116  045          P.AGL:    .ASCII  /-N-/
012701  101  052  040                    .ASCII  /A* /
012704  122  105  120                    .ASCII  /REP/
012707  114  101  103                    .ASCII  /LAC/
012712  105  115  105                    .ASCII  /EME/
012715  116  124  040                    .ASCII  /NT /
012720  102  114  117                    .ASCII  /BLO/
012723  103  113  040                    .ASCII  /CK /
012726  116  117  056                    .ASCII  /NO./
012731  040  045  104                    .ASCII  / -D/
012734  065  045  101                    .ASCII  /5-A/
012737  056  040  050                    .ASCII  /. (/
012742  117  103  124                    .ASCII  /OCT/
012745  040  045  117                    .ASCII  / -O/
012750  066  045  101                    .ASCII  /6-P/
012753  051  000  000                    .ASCII  /)/<00><00>
012756  045  116  045          P.AGM:    .ASCII  /-N-/
012761  101  052  040                    .ASCII  /A* /
012764  102  151  124                    .ASCII  /BIT/
012767  105  040  103                    .ASCII  /E C/
012772  117  125  116                    .ASCII  /OUN/
012775  124  040  111                    .ASCII  /T I/
013000  116  040  103                    .ASCII  /N C/
013003  117  115  115                    .ASCII  /OMM/
013006  101  116  104                    .ASCII  /AND/
013011  072  040  045                    .ASCII  /: -/
013014  104  065  045                    .ASCII  /D5-/
013017  101  056  000                    .ASCII  /A./<00>
013022  045  116  045          P.AGN:    .ASCII  /-N-/
013025  101  052  040                    .ASCII  /A* /
013030  102  151  124                    .ASCII  /BIT/
013033  105  040  103                    .ASCII  /E C/
013036  117  125  116                    .ASCII  /OUN/
013041  124  040  111                    .ASCII  /T I/
013044  116  040  122                    .ASCII  /N R/
013047  105  101  104                    .ASCII  /EAD/
013052  040  103  117                    .ASCII  / CO/
013055  115  115  101                    .ASCII  /MMA/
013060  116  104  072                    .ASCII  /ND:/
013063  040  045  104                    .ASCII  / -D/
013066  065  045  101                    .ASCII  /5-A/
013071  056  000  000                    .ASCII  /./<00><00>
013074  045  116  045          P.AGO:    .ASCII  /-N-/
013077  101  052  040                    .ASCII  /A* /
013102  102  151  124                    .ASCII  /BIT/
013105  105  040  103                    .ASCII  /E C/
013110  117  125  116                    .ASCII  /OUN/
013113  124  040  111                    .ASCII  /T I/
013116  116  040  122                    .ASCII  /N R/
013121  122  111  124                    .ASCII  /RIT/
013124  105  040  103                    .ASCII  /E C/
013127  117  115  115                    .ASCII  /OMM/
```

```
013136  101  116  104                .ASCII  /AND/
013143  072  040  045                .ASCII  /: %/
013146  104  065  045                .ASCII  /D5%/
013153  101  056  000                .ASCII  /A./<00>
013156  045  116  045     P.AGP:     .ASCII  /%N%/
013161  101  052  040                .ASCII  /A* /
013164  101  103  124                .ASCII  /ACT/
013167  125  101  114                .ASCII  /UAL/
013172  040  043  040                .ASCII  / # /
013175  117  106  040                .ASCII  /OF /
013200  102  131  124                .ASCII  /BYT/
013203  105  123  040                .ASCII  /ES /
013206  124  122  101                .ASCII  /TRA/
013211  116  123  106                .ASCII  /NSF/
013214  105  122  122                .ASCII  /ERR/
013217  105  104  072                .ASCII  /ED:/
013222  040  045  104                .ASCII  / %D/
013225  065  045  101                .ASCII  /5%A/
013230  056  000                     .ASCII  /./<00>
013232  045  116  045     P.AGQ:     .ASCII  /%N%/
013235  101  052  040                .ASCII  /A* /
013240  111  057  117                .ASCII  /I/<57>/O/
013243  040  102  125                .ASCII  / BU/
013246  106  106  105                .ASCII  /FFE/
013251  122  040  101                .ASCII  /R A/
013254  104  104  122                .ASCII  /DDR/
013257  105  123  123                .ASCII  /ESS/
013262  040  050  063                .ASCII  / (3/
013265  062  040  142                .ASCII  /2 b/
013270  151  164  163                .ASCII  /its/
013273  051  072  040                .ASCII  /): /
013276  045  117  066                .ASCII  /%06/
013303  045  101  040                .ASCII  /%A /
013306  045  117  066                .ASCII  /%06/
013311  000                         .ASCII  <00>
013312  045  116  045     P.AGR:     .ASCII  /%N%/
013317  101  052  040                .ASCII  /A* /
013322  111  057  117                .ASCII  /I/<57>/O/
013325  040  102  125                .ASCII  / BU/
013330  106  106  105                .ASCII  /FFE/
013333  122  040  101                .ASCII  /R A/
013336  104  104  122                .ASCII  /DDR/
013341  105  123  123                .ASCII  /ESS/
013344  040  106  117                .ASCII  / FO/
013347  122  040  122                .ASCII  /R R/
013352  105  101  104                .ASCII  /EAD/
013355  040  050  063                .ASCII  / (3/
013360  062  040  142                .ASCII  /2 b/
013363  151  164  163                .ASCII  /its/
013366  051  072  040                .ASCII  /): /
013371  045  117  066                .ASCII  /%06/
013374  045  101  040                .ASCII  /%A /
013377  045  117  066                .ASCII  /%06/
```

```
013366  000  000            .ASCII  <00><00>
013370  045  116  045  P.AGS: .ASCII  /%N%/
013373  101  052  040       .ASCII  /A* /
013376  111  057  117       .ASCII  /I/<57>/O/
013401  040  102  125       .ASCII  / BU/
013404  106  106  105       .ASCII  /FFE/
013407  122  040  101       .ASCII  /R A/
013412  104  104  122       .ASCII  /DDR/
013415  105  123  123       .ASCII  /ESS/
013420  040  106  117       .ASCII  / FO/
013423  122  040  127       .ASCII  /R W/
013426  122  111  124       .ASCII  /RIT/
013431  105  040  050       .ASCII  /E (/
013434  063  062  040       .ASCII  /32 /
013437  142  151  164       .ASCII  /bit/
013442  163  051  072       .ASCII  /s):/
013445  040  045  117       .ASCII  / %O/
013450  066  045  101       .ASCII  /6%A/
013453  040  045  117       .ASCII  / %O/
013456  066  000            .ASCII  /6/<00>
013460  045  116  045  P.AGT: .ASCII  /%N%/
013463  101  103  117       .ASCII  /ACO/
013466  116  124  105       .ASCII  /NTE/
013471  116  124  123       .ASCII  /NTS/
013474  040  117  106       .ASCII  / OF/
013477  040  103  117       .ASCII  / CO/
013502  115  115  101       .ASCII  /MMA/
013505  116  104  057       .ASCII  /ND/<57>
013510  122  105  123       .ASCII  /RES/
013513  120  117  116       .ASCII  /PON/
013516  123  105  040       .ASCII  /SE /
013521  120  101  103       .ASCII  /PAC/
013524  113  105  124       .ASCII  /KET/
013527  040  123  101       .ASCII  / SA/
013532  126  105  040       .ASCII  /VE /
013535  101  122  105       .ASCII  /ARE/
013540  101  072  045       .ASCII  /A:%/
013543  116  000  000       .ASCII  /N/<00><00>
013546  045  101  040  P.AGU: .ASCII  /%A /
013551  045  117  066       .ASCII  /%06/
013554  000  000            .ASCII  <00><00>
013556  045  116  045  P.AGV: .ASCII  /%N%/
013561  101  052  040       .ASCII  /A* /
013564  124  111  115       .ASCII  /TIM/
013567  105  072  040       .ASCII  /E: /
013572  045  132  062       .ASCII  /%Z2/
013575  045  101  072       .ASCII  /%A:/
013600  045  132  062       .ASCII  /%Z2/
013603  045  101  040       .ASCII  /%A /
013606  110  117  125       .ASCII  /HOU/
013611  122  123  045       .ASCII  /RS%/
013614  116  000            .ASCII  /N/<00>
013616  045  116  045  P.AGW: .ASCII  /%N%/
```

```
013621  101  040  052            .ASCII  /A */
013624  040  104  111            .ASCII  / DI/
013627  123  113  040            .ASCII  /SK /
013632  072  040  045            .ASCII  /: %/
013635  104  062  000            .ASCII  /D2/<00>
013640  045  116  045  P.AGX:     .ASCII  /%N%/
013643  101  104  102            .ASCII  /ADB/
013646  116  072  040            .ASCII  /N: /
013651  045  104  065            .ASCII  /%D5/
013654  045  101  056            .ASCII  /%A./
013657  040  050  117            .ASCII  / (O/
013662  103  124  040            .ASCII  /CT /
013665  045  117  066            .ASCII  /%O6/
013670  045  101  051            .ASCII  /%A)/
013673  000                      .ASCII  <00>
013674  045  116  045  P.AGY:     .ASCII  /%N%/
013677  101  102  131            .ASCII  /ABY/
013702  124  105  040            .ASCII  /TE /
013705  116  125  115            .ASCII  /NUM/
013710  102  105  122            .ASCII  /BER/
013713  072  040  045            .ASCII  /: %/
013716  104  063  000            .ASCII  /D3/<00>
013721  000                      .ASCII  <00>
013722  045  116  045  P.AGZ:     .ASCII  /%N%/
013725  101  122  101            .ASCII  /ARA/
013730  116  104  117            .ASCII  /NDO/
013733  115  040  127            .ASCII  /M W/
013736  122  111  124            .ASCII  /RIT/
013741  124  105  116            .ASCII  /TEN/
013744  040  127  117            .ASCII  / WO/
013747  122  104  040            .ASCII  /RD /
013752  072  045  102            .ASCII  /:%B/
013755  061  066  000            .ASCII  /16/<00>
013760  045  116  045  P.AHA:     .ASCII  /%N%/
013763  101  122  101            .ASCII  /ARA/
013766  116  104  117            .ASCII  /NDO/
013771  115  040  122            .ASCII  /M R/
013774  105  101  104            .ASCII  /EAD/
013777  040  127  117            .ASCII  / WO/
014002  122  104  040            .ASCII  /RD /
014005  142  151  156            .ASCII  /bin/
014010  072  045  102            .ASCII  /:%B/
014013  061  066  045            .ASCII  /16%/
014016  101  040  157            .ASCII  /A o/
014021  143  164  072            .ASCII  /ct:/
014024  045  117  066            .ASCII  /%O6/
014027  000                      .ASCII  <00>
014030  045  116  045  P.AHB:     .ASCII  /%N%/
014033  101  104  125            .ASCII  /ADU/
014036  120  114  111            .ASCII  /PLI/
014041  103  101  124            .ASCII  /CAT/
014044  105  040  125            .ASCII  /E U/
014047  116  111  124            .ASCII  /NIT/
```

```
014052  072     045     104             .ASCII  /:%D/
014055  062     045     101             .ASCII  /2%A/
014060  040     101     124             .ASCII  / AT/
014063  040     111     120             .ASCII  / IP/
014066  072     040     045             .ASCII  /: %/
014071  117     066     000             .ASCII  /06/<00>
014074  045     116     045     P.AHC:  .ASCII  /%N%/
014077  101     115     117             .ASCII  /AMO/
014102  122     105     040             .ASCII  /RE /
014105  124     110     101             .ASCII  /THA/
014110  116     040     045             .ASCII  /N %/
014113  104     061     045             .ASCII  /D1%/
014116  101     040     104             .ASCII  /A D/
014121  111     106     106             .ASCII  /IFF/
014124  105     122     105             .ASCII  /ERE/
014127  116     124     040             .ASCII  /NT /
014132  111     120     040             .ASCII  /IP /
014135  101     104     104             .ASCII  /ADD/
014140  122     105     123             .ASCII  /RES/
014143  123     105     123             .ASCII  /SES/
014146  000     000                     .ASCII  <00><00>
014150  045     101     123     P.AHD:  .ASCII  /%AS/
014153  120     111     116             .ASCII  /PIN/
014156  055     104     117             .ASCII  /-DO/
014161  127     116     040             .ASCII  /WN /
014164  111     107     116             .ASCII  /IGN/
014167  117     122     105             .ASCII  /ORE/
014172  104     000                     .ASCII  /D/<00>
014174  045     101     123     P.AHE:  .ASCII  /%AS/
014177  124     111     114             .ASCII  /TIL/
014202  114     040     103             .ASCII  /L C/
014205  117     116     116             .ASCII  /ONN/
014210  105     103     124             .ASCII  /ECT/
014213  105     104     000             .ASCII  /ED/<00>
014216  045     101     104     P.AHF:  .ASCII  /%AD/
014221  125     120     114             .ASCII  /UPL/
014224  111     103     101             .ASCII  /ICA/
014227  124     105     040             .ASCII  /TE /
014232  125     116     111             .ASCII  /UNI/
014235  124     040     116             .ASCII  /T N/
014240  125     115     102             .ASCII  /UMB/
014243  105     122     000             .ASCII  /ER/<00>
014246  045     101     101     P.AHG:  .ASCII  /%AA/
014251  114     122     105             .ASCII  /LRE/
014254  101     104     131             .ASCII  /AD1/
014257  040     117     116             .ASCII  / ON/
014262  114     111     116             .ASCII  /LIN/
014265  105     000     000             .ASCII  /E/<00><00>
014270  045     101     123     P.AHH:  .ASCII  /%AS/
014273  124     111     114             .ASCII  /TIL/
014276  114     040     117             .ASCII  /L O/
014301  116     114     111             .ASCII  /NLI/
014304  116     105     000             .ASCII  /NE/<00>
```

```
014307  000                             .ASCII  <00>
014310  045     101     125     P.AHI:  .ASCII  /%AU/
014313  116     111     124             .ASCII  /NIT/
014316  040     125     116             .ASCII  / UN/
014321  113     116     117             .ASCII  /KNO/
014324  127     116     040             .ASCII  /WN /
014327  117     122     040             .ASCII  /OR /
014332  117     116     114             .ASCII  /ONL/
014335  111     116     105             .ASCII  /INE/
014340  040     124     117             .ASCII  / TO/
014343  040     101     116             .ASCII  / AN/
014346  117     124     110             .ASCII  /OTH/
014351  105     122     040             .ASCII  /ER /
014354  103     117     116             .ASCII  /CON/
014357  124     122     117             .ASCII  /TRO/
014362  114     114     105             .ASCII  /LLE/
014365  122     000     000             .ASCII  /R/<00><00>
014370  045     101     116     P.AHJ:  .ASCII  /%AN/
014373  117     040     126             .ASCII  /O V/
014376  117     114     125             .ASCII  /OLU/
014401  115     105     040             .ASCII  /ME /
014404  115     117     125             .ASCII  /MOU/
014407  116     124     105             .ASCII  /NTE/
014412  104     040     117             .ASCII  /D O/
014415  122     040     104             .ASCII  /R D/
014420  122     111     126             .ASCII  /RIV/
014423  105     040     104             .ASCII  /E D/
014426  111     123     101             .ASCII  /ISA/
014431  102     114     105             .ASCII  /BLE/
014434  104     040     102             .ASCII  /D B/
014437  131     040     123             .ASCII  /Y S/
014442  127     111     124             .ASCII  /WIT/
014445  103     110     000             .ASCII  /CH/<00>
014450  045     101     125     P.AHK:  .ASCII  /%AU/
014453  116     111     124             .ASCII  /NIT/
014456  040     111     116             .ASCII  / IN/
014461  117     120     105             .ASCII  /OPE/
014464  122     101     124             .ASCII  /RAT/
014467  111     126     105             .ASCII  /IVE/
014472  040     050     122             .ASCII  / (R/
014475  104     065     061             .ASCII  /D51/
014500  057     065     062             .ASCII  <57>/52/
014503  040     167     162             .ASCII  / wr/
014506  151     164     145             .ASCII  /ite/
014511  040     146     141             .ASCII  / fa/
014514  165     154     164             .ASCII  /ult/
014517  051     000     000             .ASCII  /)/<00><00>
014522  045     101     125     P.AHL:  .ASCII  /%AU/
014525  116     111     124             .ASCII  /NIT/
014530  040     104     111             .ASCII  / DI/
014533  123     101     102             .ASCII  /SAB/
014536  114     105     104             .ASCII  /LED/
014541  040     102     131             .ASCII  / BY/
```

```
014544  040     106     111             .ASCII  / FI/
014547  105     114     104             .ASCII  /ELD/
014552  040     123     105             .ASCII  / SE/
014555  122     126     111             .ASCII  /RVI/
014560  103     105     040             .ASCII  /CE /
014563  117     122     040             .ASCII  /OR /
014566  111     116     124             .ASCII  /INT/
014571  105     122     116             .ASCII  /ERN/
014574  101     114     040             .ASCII  /AL /
014577  104     111     101             .ASCII  /DIA/
014602  107     116     117             .ASCII  /GNO/
014605  123     124     111             .ASCII  /STI/
014610  103     123     000             .ASCII  /CS/<00>
014613  000                             .ASCII  <00>
014614  045     101     042     P.AHM:  .ASCII  /%A"/
014617  106     117     122             .ASCII  /FOR/
014622  103     105     104             .ASCII  /CED/
014625  040     105     122             .ASCII  / ER/
014630  122     117     122             .ASCII  /ROR/
014633  042     040     104             .ASCII  /" D/
014636  105     124     105             .ASCII  /ETE/
014641  103     124     105             .ASCII  /CTE/
014644  104     040     127             .ASCII  /D W/
014647  110     111     114             .ASCII  /HIL/
014652  105     040     101             .ASCII  /E A/
014655  103     103     105             .ASCII  /CCE/
014660  123     123     111             .ASCII  /SSI/
014663  116     107     040             .ASCII  /NG /
014666  106     103     124             .ASCII  /FCT/
014671  040     117     122             .ASCII  / OR/
014674  040     122     103             .ASCII  / RC/
014677  124     000     000             .ASCII  /T/<00><00>
014702  045     101     123     P.AHN:  .ASCII  /%AS/
014705  105     103     124             .ASCII  /ECT/
014710  117     122     040             .ASCII  /OR /
014713  110     101     104             .ASCII  /HAD/
014716  040     102     105             .ASCII  / BE/
014721  105     116     040             .ASCII  /EN /
014724  127     122     111             .ASCII  /WRI/
014727  124     124     105             .ASCII  /TTE/
014732  116     040     127             .ASCII  /N W/
014735  111     124     110             .ASCII  /ITH/
014740  040     042     106             .ASCII  / "F/
014743  117     122     103             .ASCII  /ORC/
014746  105     104     040             .ASCII  /ED /
014751  105     122     122             .ASCII  /ERR/
014754  117     122     042             .ASCII  /OR"/
014757  040     115     117             .ASCII  / MO/
014762  104     111     106             .ASCII  /DIF/
014765  111     105     122             .ASCII  /IER/
014770  000     000                     .ASCII  <00><00>
014772  045     101     106     P.AHO:  .ASCII  /%AF/
014775  103     124     040             .ASCII  /CT /
```

```
015000  117  122  040          .ASCII  /OR /
015003  122  103  124          .ASCII  /RCT/
015006  040  125  116          .ASCII  / UN/
015011  122  105  101          .ASCII  /REA/
015014  104  101  102          .ASCII  /DAB/
015017  114  105  040          .ASCII  /LE /
015022  055  040  111          .ASCII  /- I/
015025  116  126  101          .ASCII  /NVA/
015030  114  111  104          .ASCII  /LID/
015033  040  123  105          .ASCII  / SE/
015036  103  124  117          .ASCII  /CTO/
015041  122  040  110          .ASCII  /R H/
015044  105  101  104          .ASCII  /EAD/
015047  105  122  000          .ASCII  /ER/<00>
015052  045  101  110  P.AHP:   .ASCII  /%AH/
015055  105  101  104          .ASCII  /EAD/
015060  105  122  040          .ASCII  /ER /
015063  103  117  115          .ASCII  /COM/
015066  120  101  122          .ASCII  /PAR/
015071  105  040  105          .ASCII  /E E/
015074  122  122  117          .ASCII  /RRO/
015077  122  040  050          .ASCII  /R (/
015102  126  141  154          .ASCII  /Val/
015105  151  144  040          .ASCII  /id /
015110  150  145  141          .ASCII  /hea/
015113  144  145  162          .ASCII  /der/
015116  040  156  157          .ASCII  / no/
015121  164  040  146          .ASCII  /t f/
015124  157  165  156          .ASCII  /oun/
015127  144  051  000          .ASCII  /d)/<00>
015132  045  101  106  P.AHQ:   .ASCII  /%AF/
015135  103  124  040          .ASCII  /CT /
015140  117  122  040          .ASCII  /OR /
015143  122  103  124          .ASCII  /RCT/
015146  040  125  116          .ASCII  / UN/
015151  122  105  101          .ASCII  /REA/
015154  104  101  102          .ASCII  /DAB/
015157  114  105  040          .ASCII  /LE /
015162  055  040  104          .ASCII  /- D/
015165  101  124  101          .ASCII  /ATA/
015170  040  123  131          .ASCII  / SY/
015173  116  103  040          .ASCII  /NC /
015176  124  111  115          .ASCII  /TIM/
015201  105  117  125          .ASCII  /EOU/
015204  124  000                .ASCII  /T.<00>
015206  045  101  104  P.AHR:   .ASCII  /%AD/
015211  101  124  101          .ASCII  /ATA/
015214  040  123  131          .ASCII  / SY/
015217  116  103  040          .ASCII  /NC /
015222  116  117  124          .ASCII  /NOT/
015225  040  106  117          .ASCII  / FO/
015230  125  116  104          .ASCII  /UND/
015233  040  050  104          .ASCII  / (D/
```

```
015236  141  164  141          .ASCII  /ata/
015241  040  163  171          .ASCII  / sy/
015244  156  143  040          .ASCII  /nc /
015247  164  151  155          .ASCII  /tim/
015252  145  157  165          .ASCII  /eou/
015255  164  051  000          .ASCII  /t)/<00>
015260  045  101  106  P.AHS:   .ASCII  /%AF/
015263  103  124  040          .ASCII  /CT /
015266  117  122  040          .ASCII  /OR /
015271  122  103  124          .ASCII  /RCT/
015274  040  125  116          .ASCII  / UN/
015277  122  105  101          .ASCII  /REA/
015302  104  101  102          .ASCII  /DAB/
015305  114  105  040          .ASCII  /LE /
015310  055  040  125          .ASCII  /- U/
015313  116  103  117          .ASCII  /NCO/
015316  122  122  105          .ASCII  /RRE/
015321  103  124  101          .ASCII  /CTA/
015324  102  114  105          .ASCII  /BLE/
015327  040  105  103          .ASCII  / EC/
015332  103  040  105          .ASCII  /C E/
015335  122  122  117          .ASCII  /RRO/
015340  122  000               .ASCII  /R/<00>
015342  045  101  125  P.AHT:   .ASCII  /%AU/
015345  116  103  117          .ASCII  /NCO/
015350  122  122  105          .ASCII  /RRE/
015353  103  124  101          .ASCII  /CTA/
015356  102  114  105          .ASCII  /BLE/
015361  040  105  103          .ASCII  / EC/
015364  103  040  105          .ASCII  /C E/
015367  122  122  117          .ASCII  /RRO/
015372  122  000               .ASCII  /R/<00>
015374  045  101  122  P.AHU:   .ASCII  /%AR/
015377  103  124  040          .ASCII  /CT /
015402  103  117  122          .ASCII  /COR/
015405  122  125  120          .ASCII  /RUP/
015410  124  105  104          .ASCII  /TED/
015413  000                    .ASCII  <00>
015414  045  101  116  P.AHV:   .ASCII  /%AN/
015417  117  040  122          .ASCII  /O R/
015422  105  120  114          .ASCII  /EPL/
015425  101  103  105          .ASCII  /ACE/
015430  115  105  116          .ASCII  /MEN/
015433  124  040  102          .ASCII  /T B/
015436  114  117  103          .ASCII  /LOC/
015441  113  040  101          .ASCII  /K A/
015444  126  101  111          .ASCII  /VAI/
015447  114  101  102          .ASCII  /LAB
015452  114  105  040          .ASCII  /LE /
015455  050  122  103          .ASCII  /(RC/
015460  124  040  146          .ASCII  /T f/
015463  165  154  154          .ASCII  /ull/
015466  051  000               .ASCII  /)/<00>
```

```
015430  045  101  104  P.AHW:  .ASCII  /%AD/
015433  111  123  113         .ASCII  /ISK/
015436  040  116  117         .ASCII  / NO/
015501  124  040  106         .ASCII  /T F/
015504  117  122  115         .ASCII  /ORM/
015507  101  124  124         .ASCII  /ATT/
015512  105  104  040         .ASCII  /ED /
015515  127  111  124         .ASCII  /WIT/
015520  110  040  065         .ASCII  /H 5/
015523  061  062  040         .ASCII  /12 /
015526  102  131  124         .ASCII  /BYT/
015531  105  040  123         .ASCII  /E S/
015534  105  103  124         .ASCII  /ECT/
015537  117  122  123         .ASCII  /ORS/
015542  000  000              .ASCII  <00><00>
015544  045  101  104  P.AHX:  .ASCII  /%AD/
015547  111  123  113         .ASCII  /ISK/
015552  040  116  117         .ASCII  / NO/
015555  124  040  106         .ASCII  /T F/
015560  117  122  115         .ASCII  /ORM/
015563  101  124  124         .ASCII  /ATT/
015566  105  104  040         .ASCII  /ED /
015571  117  122  040         .ASCII  /OR /
015574  106  103  124         .ASCII  /FCT/
015577  040  103  117         .ASCII  / CO/
015602  122  122  125         .ASCII  /RRU/
015605  120  124  105         .ASCII  /PTE/
015610  104  000              .ASCII  /D/<00>
015612  045  101  117  P.AHY:  .ASCII  /%AO/
015615  116  105  040         .ASCII  /NE /
015620  123  131  115         .ASCII  /SYM/
015623  102  117  114         .ASCII  /BOL/
015626  040  105  103         .ASCII  / EC/
015631  103  040  105         .ASCII  /C E/
015634  122  122  117         .ASCII  /RRO/
015637  122  000  000         .ASCII  /R/<00><00>
015642  045  101  124  P.AHZ:  .ASCII  /%AT/
015645  127  117  040         .ASCII  /WO /
015650  123  131  115         .ASCII  /SYM/
015653  102  117  114         .ASCII  /BOL/
015656  040  105  103         .ASCII  / EC/
015661  103  040  105         .ASCII  /C E/
015664  122  122  117         .ASCII  /RRO/
015667  122  000  000         .ASCII  /R/<00><00>
015672  045  101  124  P.AIA:  .ASCII  /%AT/
015675  110  122  105         .ASCII  /HRE/
015700  105  040  123         .ASCII  /E S/
015703  131  115  102         .ASCII  /YMB/
015706  117  114  040         .ASCII  /OL /
015711  105  103  103         .ASCII  /ECC/
015714  040  105  122         .ASCII  / ER/
015717  120  117  122         .ASCII  /ROR/
015722  000  000              .ASCII  <00><00>
```

```
015724  045  101  106  P.AIB:  .ASCII  /*AF/
015727  117  125  122          .ASCII  /OUR/
015732  040  123  131          .ASCII  / SY/
015735  115  102  117          .ASCII  /MBO/
015740  114  040  105          .ASCII  /L E/
015743  103  103  040          .ASCII  /CC /
015746  105  122  122          .ASCII  /ERR/
015751  117  122  000          .ASCII  /OR/<00>
015754  045  101  106  P.AIC:  .ASCII  /*AF/
015757  111  126  105          .ASCII  /IVE/
015762  040  123  131          .ASCII  / SY/
015765  115  102  117          .ASCII  /MBO/
015770  114  040  105          .ASCII  /L E/
015773  103  103  040          .ASCII  /CC /
015776  105  122  122          .ASCII  /ERR/
016001  117  122  000          .ASCII  /OR/<00>
016004  045  101  123  P.AID:  .ASCII  /*AS/
016007  111  130  040          .ASCII  /IX /
016012  123  131  115          .ASCII  /SYM/
016015  102  117  114          .ASCII  /BOL/
016020  040  105  103          .ASCII  / EC/
016023  103  040  105          .ASCII  /C E/
016026  122  122  117          .ASCII  /RRO/
016031  122  000  000          .ASCII  /R/<00><00>
016034  045  101  123  P.AIE:  .ASCII  /*AS/
016037  105  126  105          .ASCII  /EVE/
016042  116  040  123          .ASCII  /N S/
016045  131  115  102          .ASCII  /YMB/
016050  117  114  040          .ASCII  /OL /
016053  105  103  103          .ASCII  /ECC/
016056  040  105  122          .ASCII  / ER/
016061  122  117  122          .ASCII  /ROR/
016064  000  000               .ASCII  <00><00>
016066  045  101  105  P.AIF:  .ASCII  /*AE/
016071  111  107  110          .ASCII  /IGH/
016074  124  040  123          .ASCII  /T S/
016077  131  115  102          .ASCII  /YMB/
016102  117  114  040          .ASCII  /OL /
016105  105  103  103          .ASCII  /ECC/
016110  040  105  122          .ASCII  / ER/
016113  122  117  122          .ASCII  /ROR/
016116  000  000               .ASCII  <00><00>
016120  045  101  103  P.AIG:  .ASCII  /*AC/
016123  117  122  122          .ASCII  /ORR/
016126  105  103  124          .ASCII  /ECT/
016131  101  102  114          .ASCII  /ABL/
016134  105  040  105          .ASCII  /E E/
016137  122  122  117          .ASCII  /RRO/
016142  122  040  111          .ASCII  /R I/
016145  116  040  105          .ASCII  /N E/
016150  103  103  040          .ASCII  /CC /
016153  106  111  105          .ASCII  /FIE/
016156  114  104  000          .ASCII  /LD/<00>
```

M7


```
016161  000                             .ASCII  <00>
016162  045     101     125     P.AIH:  .ASCII  /%AU/
016165  116     111     124             .ASCII  /NIT/
016170  040     123     117             .ASCII  / SO/
016173  106     124     127             .ASCII  /FTW/
016176  101     122     105             .ASCII  /ARE/
016201  040     127     122             .ASCII  / WR/
016204  111     124     105             .ASCII  /ITE/
016207  040     120     122             .ASCII  / PR/
016212  117     124     105             .ASCII  /OTE/
016215  103     124     105             .ASCII  /CTE/
016220  104     000                     .ASCII  /D/<00>
016222  045     101     125     P.AII:  .ASCII  /%AU/
016225  116     111     124             .ASCII  /NIT/
016230  040     110     101             .ASCII  / HA/
016233  122     104     127             .ASCII  /RDW/
016236  101     122     105             .ASCII  /ARE/
016241  040     127     122             .ASCII  / WR/
016244  111     124     105             .ASCII  /ITE/
016247  040     120     122             .ASCII  / PR/
016252  117     124     105             .ASCII  /OTE/
016255  103     124     105             .ASCII  /CTE/
016260  104     000                     .ASCII  /D/<00>
016262  045     101     117     P.AIJ:  .ASCII  /%AO/
016265  104     104     040             .ASCII  /DD /
016270  124     122     101             .ASCII  /TRA/
016273  116     123     106             .ASCII  /NSF/
016276  105     122     040             .ASCII  /ER /
016301  101     104     104             .ASCII  /ADD/
016304  122     105     123             .ASCII  /RES/
016307  123     000     000             .ASCII  /S/<00><00>
016312  045     101     117     P.AIK:  .ASCII  /%AO/
016315  104     104     040             .ASCII  /DD /
016320  102     131     124             .ASCII  /BYT/
016323  105     040     103             .ASCII  /E C/
016326  117     125     116             .ASCII  /OUN/
016331  124     000     000             .ASCII  /T/<00><00>
016334  045     101     116     P.AIL:  .ASCII  /%AN/
016337  117     116     055             .ASCII  /ON-/
016342  105     130     111             .ASCII  /EXI/
016345  123     124     105             .ASCII  /STE/
016350  116     124     040             .ASCII  /NT /
016353  110     117     123             .ASCII  /HOS/
016356  124     040     115             .ASCII  /T M/
016361  105     115     117             .ASCII  /EMO/
016364  122     131     000             .ASCII  /RY/<00>
016367  000                             .ASCII  <00>
016370  045     101     110     P.AIM:  .ASCII  /%AH/
016373  117     123     124             .ASCII  /OST/
016376  040     115     105             .ASCII  / ME/
016401  115     117     122             .ASCII  /MOR/
016404  131     040     120             .ASCII  /Y P/
016407  101     122     111             .ASCII  /ARI/
```

```
016412  124     131     040             .ASCII  /TY /
016415  105     122     122             .ASCII  /ERR/
016420  117     122     000             .ASCII  /OR/<00>
016423  000                             .ASCII  <00>
016424  045     101     103     P.AIN:  .ASCII  /%AC/
016427  117     115     115             .ASCII  /OMM/
016432  101     116     104             .ASCII  /AND/
016435  040     124     111             .ASCII  / TI/
016440  115     117     125             .ASCII  /MOU/
016443  124     040     117             .ASCII  /T O/
016446  122     040     122             .ASCII  /R R/
016451  105     124     122             .ASCII  /ETR/
016454  131     040     114             .ASCII  /Y L/
016457  111     115     111             .ASCII  /IMI/
016462  124     040     105             .ASCII  /T E/
016465  130     103     105             .ASCII  /XCE/
016470  105     104     105             .ASCII  /EDE/
016473  104     000     000             .ASCII  /D/<00><00>
016476  045     101     123     P.AIO:  .ASCII  /%AS/
016501  105     122     111             .ASCII  /ERI/
016504  101     114     111             .ASCII  /ALI/
016507  132     105     122             .ASCII  /ZER/
016512  057     104     105             .ASCII  <57>/DE/
016515  123     105     122             .ASCII  /SER/
016520  111     101     114             .ASCII  /IAL/
016523  111     132     105             .ASCII  /IZE/
016526  122     040     117             .ASCII  /R O/
016531  126     105     122             .ASCII  /VER/
016534  122     125     116             .ASCII  /RUN/
016537  040     117     122             .ASCII  / OR/
016542  040     125     116             .ASCII  / UN/
016545  104     105     122             .ASCII  /DER/
016550  122     125     116             .ASCII  /RUN/
016553  000                             .ASCII  <00>
016554  045     101     042     P.AIP:  .ASCII  /%A"/
016557  105     122     122             .ASCII  /ERR/
016562  117     122     040             .ASCII  /OR /
016565  104     105     124             .ASCII  /DET/
016570  105     103     124             .ASCII  /ECT/
016573  111     117     116             .ASCII  /ION/
016576  040     103     117             .ASCII  / CO/
016601  104     105     042             .ASCII  /DE"/
016604  040     105     122             .ASCII  / ER/
016607  122     117     122             .ASCII  /ROR/
016612  000     000                     .ASCII  <00><00>
016614  045     101     111     P.AIQ:  .ASCII  /%AI/
016617  116     103     117             .ASCII  /NCO/
016622  116     123     111             .ASCII  /NSI/
016625  123     124     105             .ASCII  /STE/
016630  116     124     040             .ASCII  /NT /
016633  111     116     124             .ASCII  /INT/
016636  105     122     116             .ASCII  /ERN/
016641  101     114     040             .ASCII  /AL /
```

```
016644  104  101  124          .ASCII  /DAT/
016647  101  040  125          .ASCII  /A S/
016652  124  122  125          .ASCII  /TRU/
016655  103  124  125          .ASCII  /CTU/
016660  122  105  000          .ASCII  /RE/<00>
016663  000                    .ASCII  <00>
016664  045  101  104  P.AIR:   .ASCII  /%AD/
016667  122  111  126          .ASCII  /RIV/
016672  105  040  103          .ASCII  /E C/
016675  117  115  115          .ASCII  /OMM/
016700  101  116  104          .ASCII  /AND/
016703  040  124  111          .ASCII  / TI/
016706  115  105  117          .ASCII  /ME0/
016711  125  124  040          .ASCII  /UT /
016714  050  116  157          .ASCII  /(No/
016717  040  162  145          .ASCII  / re/
016722  163  160  157          .ASCII  /spo/
016725  156  163  145          .ASCII  /nse/
016730  040  157  162          .ASCII  / or/
016733  040  163  145          .ASCII  / se/
016736  145  153  040          .ASCII  /ek /
016741  151  156  143          .ASCII  /inc/
016744  157  155  160          .ASCII  /omp/
016747  154  145  164          .ASCII  /let/
016752  145  051  000          .ASCII  /e)/<00>
016755  000                    .ASCII  <00>
016756  045  101  103  P.AIS:   .ASCII  /%AC/
016761  117  116  124          .ASCII  /ONT/
016764  122  117  114          .ASCII  /ROL/
016767  114  105  122          .ASCII  /LER/
016772  040  104  105          .ASCII  / DE/
016775  124  105  103          .ASCII  /TEC/
017000  124  105  104          .ASCII  /TED/
017003  040  124  122          .ASCII  / TR/
017006  101  116  123          .ASCII  /ANS/
017011  115  111  123          .ASCII  /MIS/
017014  123  111  117          .ASCII  /SIO/
017017  116  040  117          .ASCII  /N O/
017022  122  040  120          .ASCII  /R P/
017025  122  117  124          .ASCII  /ROT/
017030  117  103  117          .ASCII  /OCO/
017033  114  040  105          .ASCII  /L E/
017036  122  122  117          .ASCII  /RRO/
017041  122  000  000          .ASCII  /R/<00><00>
017044  045  101  120  P.AII:   .ASCII  /%AP/
017047  117  123  111          .ASCII  /OSI/
017052  124  111  117          .ASCII  /TIO/
017055  116  040  105          .ASCII  /N E/
017060  122  122  117          .ASCII  /RRO/
017063  122  040  050          .ASCII  /R (
017066  115  151  163          .ASCII  /Mis/
017071  055  145  170          .ASCII  /-ex/
017074  145  155  051          .ASCII  /em)/
```

```
017077  000
017100  045  101  114        P.AIU:  .ASCII  <00>
017103  117  123  124                .ASCII  /*AL/
017106  040  122  105                .ASCII  /OST/
017111  101  104  057                .ASCII  / RE/
017114  122  122  111                .ASCII  /AD/<57>
017117  124  105  040                .ASCII  /WRI/
017122  122  105  101                .ASCII  /TE /
017125  104  151  040                .ASCII  /REA/
017130  104  125  122                .ASCII  /D1 /
017133  111  116  107                .ASCII  /DUR/
017136  057  103  105                .ASCII  /ING/
017141  124  127  105                .ASCII  <57>/BE/
017144  105  116  040                .ASCII  /TWE/
017147  124  122  101                .ASCII  /EN /
017152  116  123  106                .ASCII  /TRA/
017155  105  122  123                .ASCII  /NSF/
017160  000  000                    .ASCII  /ERS/
017162  045  101  104                .ASCII  <00><00>
017165  122  111  126        P.AIV:  .ASCII  /*AD/
017170  105  040  103                .ASCII  /RIV/
017173  114  117  103                .ASCII  /E C/
017176  113  040  104                .ASCII  /LOC/
017201  122  117  120                .ASCII  /K D/
017204  117  125  124                .ASCII  /ROP/
017207  000                        .ASCII  /OUT/
017210  045  101  114        P.AIW:  .ASCII  <00>
017213  117  123  124                .ASCII  /*AL/
017216  040  122  105                .ASCII  /OST/
017221  103  105  111                .ASCII  / RE/
017224  126  105  122                .ASCII  /CEI/
017227  040  122  105                .ASCII  /VER/
017232  101  104  151                .ASCII  / RE/
017235  040  102  105                .ASCII  /ADI/
017240  124  127  105                .ASCII  / BE/
017243  105  116  040                .ASCII  /TWE/
017246  123  105  103                .ASCII  /EN /
017251  124  117  122                .ASCII  /SEC/
017254  123  000                    .ASCII  /TOR/
017256  045  101  104        P.AIX:  .ASCII  /S/<00>
017261  122  111  126                .ASCII  /*AD/
017264  105  040  104                .ASCII  /RIV/
017267  105  124  105                .ASCII  /E D/
017272  103  124  105                .ASCII  /ETE/
017275  104  040  105                .ASCII  /CTE/
017300  122  122  117                .ASCII  /D E/
017303  122  000  000                .ASCII  /RRO/
017306  045  101  103        P.AIY:  .ASCII  /R/<00><00>
017311  117  116  124                .ASCII  /*AC/
017314  122  117  114                .ASCII  /ONT/
017317  114  105  122                .ASCII  /ROL/
017322  040  104  105                .ASCII  /LER/
017325  124  105  103                .ASCII  / DE/
                                    .ASCII  /TEC/
```

```
017330  124  105  104            .ASCII  /TED/
017333  040  120  125            .ASCII  / PU/
017336  114  123  105            .ASCII  /LSE/
017341  040  117  122            .ASCII  / OR/
017344  040  123  124            .ASCII  / ST/
017347  101  124  105            .ASCII  /ATE/
017352  040  120  101            .ASCII  / PA/
017355  122  111  124            .ASCII  /RIT/
017360  131  040  105            .ASCII  /Y E/
017363  122  122  117            .ASCII  /RRO/
017366  122  000                 .ASCII  /R/<00>
017370  045  101  103  P.AJA:    .ASCII  /%AC/
017373  117  116  124            .ASCII  /ONT/
017376  122  117  114            .ASCII  /ROL/
017401  114  105  122            .ASCII  /LER/
017404  040  124  111            .ASCII  / TI/
017407  115  105  117            .ASCII  /MEO/
017412  125  124  000            .ASCII  /UT/<00>
017415  000                      .ASCII  <00>
017416  045  101  105  P.AJB:    .ASCII  /%AE/
017421  116  126  105            .ASCII  /NVE/
017424  114  117  120            .ASCII  /LOP/
017427  105  057  120            .ASCII  /E/<57>/P/
017432  101  103  113            .ASCII  /ACK/
017435  105  124  040            .ASCII  /ET /
017440  122  105  101            .ASCII  /RFA/
017443  104  040  105            .ASCII  /D E/
017446  122  122  117            .ASCII  /RRO/
017451  122  040  050            .ASCII  /R (/
017454  120  141  162            .ASCII  /Par/
017457  151  164  171            .ASCII  /ity/
017462  040  157  162            .ASCII  / or/
017465  040  164  151            .ASCII  / ti/
017470  155  145  157            .ASCII  /meo/
017473  165  164  051            .ASCII  /ut)/
017476  000  000                 .ASCII  <00><00>
017500  045  101  105  P.AJC:    .ASCII  /%AE/
017503  116  126  105            .ASCII  /NVE/
017506  114  117  120            .ASCII  /LOP/
017511  105  057  120            .ASCII  /E/<57>/P/
017514  101  103  113            .ASCII  /ACK/
017517  105  124  040            .ASCII  /ET /
017522  127  122  111            .ASCII  /WRI/
017525  124  105  040            .ASCII  /TE /
017530  105  122  122            .ASCII  /ERR/
017533  117  122  040            .ASCII  /OR /
017536  050  120  141            .ASCII  /(Pa/
017541  162  151  164            .ASCII  /rit/
017544  171  040  157            .ASCII  /y o/
017547  162  040  164            .ASCII  /r t/
017552  151  155  145            .ASCII  /ime/
017557  157  165  164            .ASCII  /out/
017560  051  000                 .ASCII  /)/<00>
```

```
017562  045  101  103  P.AJD:  .ASCII  /*AC/
017565  117  116  124         .ASCII  /ONT/
017570  122  117  114         .ASCII  /ROL/
017573  114  105  122         .ASCII  /LER/
017576  040  122  117         .ASCII  / RO/
017601  115  040  101         .ASCII  /M A/
017604  116  104  040         .ASCII  /ND /
017607  122  101  115         .ASCII  /RAM/
017612  040  120  101         .ASCII  / PA/
017615  122  111  124         .ASCII  /RIT/
017620  131  040  105         .ASCII  /Y E/
017623  122  122  117         .ASCII  /RRO/
017626  122  000              .ASCII  /R/<00>
017630  045  101  103  P.AJE:  .ASCII  /*AC/
017633  117  116  124         .ASCII  /ONT/
017636  122  117  114         .ASCII  /ROL/
017641  114  105  122         .ASCII  /LER/
017644  040  122  101         .ASCII  / RA/
017647  115  040  120         .ASCII  /M P/
017652  101  122  111         .ASCII  /ARI/
017655  124  131  040         .ASCII  /TY /
017660  105  122  122         .ASCII  /ERR/
017663  117  122  000         .ASCII  /OR/<00>
017666  045  101  103  P.AJF:  .ASCII  /*AC/
017671  117  116  124         .ASCII  /ONT/
017674  122  117  114         .ASCII  /ROL/
017677  114  105  122         .ASCII  /LER/
017702  040  122  117         .ASCII  / RO/
017705  115  040  120         .ASCII  /M P/
017710  101  122  111         .ASCII  /ARI/
017713  124  131  040         .ASCII  /TY /
017716  105  122  122         .ASCII  /ERR/
017721  117  122  000         .ASCII  /OR/<00>
017724  045  101  122  P.AJG:  .ASCII  /*AR/
017727  111  116  107         .ASCII  /ING/
017732  040  122  105         .ASCII  / RE/
017735  101  104  040         .ASCII  /AD /
017740  105  122  122         .ASCII  /ERR/
017743  117  122  040         .ASCII  /OR /
017746  050  120  141         .ASCII  /(Pa/
017751  162  151  164         .ASCII  /rit/
017754  171  040  157         .ASCII  /y o/
017757  162  040  164         .ASCII  /r t/
017762  151  155  145         .ASCII  /ime/
017765  157  165  164         .ASCII  /out/
017770  051  000              .ASCII  /)/<00>
017772  045  101  122  P.AJH:  .ASCII  /*AR/
017775  111  116  107         .ASCII  /ING/
020000  040  127  122         .ASCII  / WR/
020003  111  124  105         .ASCII  /IT /
020006  040  105  122         .ASCII  / ER/
020011  122  117  122         .ASCII  /ROR/
020014  040  050  120         .ASCII  / (P/
```

```
020017  141  162  151          .ASCII  /ari/
020022  164  171  040          .ASCII  /ty /
020025  157  162  040          .ASCII  /or /
020030  164  151  155          .ASCII  /tim/
020033  145  157  165          .ASCII  /eou/
020036  164  051  000          .ASCII  /t)/<00>
020041  000                    .ASCII  <00>
020042  111  116  124  P.AJI:  .ASCII  /INT/
020045  105  122  122          .ASCII  /ERR/
020050  125  120  124          .ASCII  /UPT/
020053  040  115  101          .ASCII  / MA/
020056  123  124  105          .ASCII  /STE/
020061  122  040  106          .ASCII  /R F/
020064  101  111  114          .ASCII  /AIL/
020067  125  122  105          .ASCII  /URE/
020072  000  000                .ASCII  <00><00>
020074  045  101  110  P.AJJ:  .ASCII  /%AH/
020077  117  123  124          .ASCII  /OST/
020102  040  101  103          .ASCII  / AC/
020105  103  105  123          .ASCII  /CES/
020110  123  040  124          .ASCII  /S T/
020113  111  115  105          .ASCII  /IME/
020116  117  125  124          .ASCII  /OUT/
020121  040  050  110          .ASCII  / (H/
020124  151  147  150          .ASCII  /igh/
020127  145  162  040          .ASCII  /er /
020132  154  145  166          .ASCII  /lev/
020135  145  154  040          .ASCII  /el /
020140  160  162  157          .ASCII  /pro/
020143  164  157  143          .ASCII  /toc/
020146  157  154  040          .ASCII  /ol /
020151  144  145  160          .ASCII  /dep/
020154  145  156  144          .ASCII  /end/
020157  145  156  164          .ASCII  /ent/
020162  051  000                .ASCII  /)/<00>
020164  045  101  103  P.AJK:  .ASCII  /%AC/
020167  122  105  104          .ASCII  /RED/
020172  111  124  040          .ASCII  /IT /
020175  114  111  115          .ASCII  /LIM/
020200  111  124  040          .ASCII  /IT /
020203  105  130  103          .ASCII  /EXC/
020206  105  105  104          .ASCII  /EED/
020211  105  104  000          .ASCII  /ED/<00>
020214  045  101  121  P.AJL:  .ASCII  /%AQ/
020217  055  102  125          .ASCII  / BU/
020222  123  040  115          .ASCII  /S M/
020225  101  123  124          .ASCII  /AST/
020230  105  122  040          .ASCII  /ER /
020233  105  122  122          .ASCII  /ERR/
020236  117  122  000          .ASCII  /OR/<00>
020241  000                    .ASCII  <00>
020242  045  101  103  P.AJM:  .ASCII  /%AC/
020245  117  116  124          .ASCII  /ONT/
```

```
020250  122     117     114             .ASCII  /ROL/
020253  114     105     122             .ASCII  /LER/
020256  040     106     101             .ASCII  / FA/
020261  124     101     114             .ASCII  /TAL/
020264  040     105     122             .ASCII  / ER/
020267  122     117     122             .ASCII  /ROR/
020272  000     000                     .ASCII  <00><00>
020274  045     101     111     P.AJN:  .ASCII  /%AI/
020277  116     123     124             .ASCII  /NST/
020302  122     125     103             .ASCII  /RUC/
020305  124     111     117             .ASCII  /TIO/
020310  116     040     114             .ASCII  /N L/
020313  117     117     120             .ASCII  /OOP/
020316  040     124     111             .ASCII  / TI/
020321  115     105     117             .ASCII  /MEO/
020324  125     124     000             .ASCII  /UT/<00>
020327  000                             .ASCII  <00>
020330  045     101     111     P.AJO:  .ASCII  /%AI/
020333  114     114     105             .ASCII  /LLE/
020336  107     101     114             .ASCII  /GAL/
020341  040     126     111             .ASCII  / VI/
020344  122     124     125             .ASCII  /RTU/
020347  101     114     040             .ASCII  /AL /
020352  103     111     122             .ASCII  /CIR/
020355  103     125     111             .ASCII  /CUI/
020360  124     040     111             .ASCII  /T I/
020363  104     000     000             .ASCII  /D/<00><00>
020366  045     101     111     P.AJP:  .ASCII  /%AI/
020371  116     124     105             .ASCII  /NTE/
020374  122     122     125             .ASCII  /RRU/
020377  120     124     040             .ASCII  /PT /
020402  126     105     103             .ASCII  /VEC/
020405  124     117     122             .ASCII  /TOR/
020410  040     111     114             .ASCII  / IL/
020413  114     105     107             .ASCII  /LEG/
020416  101     114     000             .ASCII  /AL/<00>
020421  000                             .ASCII  <00>
020422  045     101     115     P.AJQ:  .ASCII  /%AM/
020425  101     111     116             .ASCII  /AIN/
020430  124     105     116             .ASCII  /TEN/
020433  101     116     103             .ASCII  /ANC/
020436  105     040     122             .ASCII  /E R/
020441  105     101     104             .ASCII  /EAD/
020444  057     127     122             .ASCII  <57>/WR/
020447  111     124     105             .ASCII  /ITE/
020452  040     111     116             .ASCII  / IN/
020455  126     101     114             .ASCII  /VAL/
020460  111     104     040             .ASCII  /ID /
020463  122     105     107             .ASCII  /REG/
020466  111     117     116             .ASCII  /ION/
020471  040     111     104             .ASCII  / ID/
020474  105     116     124             .ASCII  /ENT/
020477  111     106     111             .ASCII  /IFI/
```

```
020502  105  122  000                    .ASCII  /ER/<00>
020505  000                             .ASCII  <00>
020506  045  101  115          P.AJR:    .ASCII  /%AM/
020511  101  111  116                    .ASCII  /AIN/
020514  124  105  116                    .ASCII  /TEN/
020517  101  116  103                    .ASCII  /ANC/
020522  105  040  127                    .ASCII  /E W/
020525  122  111  124                    .ASCII  /RIT/
020530  105  040  114                    .ASCII  /E L/
020533  117  101  104                    .ASCII  /OAD/
020536  040  124  117                    .ASCII  / TO/
020541  040  116  117                    .ASCII  / NO/
020544  116  055  114                    .ASCII  /N-L/
020547  117  101  104                    .ASCII  /OAD/
020552  101  102  114                    .ASCII  /ABL/
020555  105  040  103                    .ASCII  /E C/
020560  117  116  124                    .ASCII  /ONT/
020563  122  117  114                    .ASCII  /ROL/
020566  114  105  122                    .ASCII  /LER/
020571  000                             .ASCII  <00>
020572  045  101  103          P.AJS:    .ASCII  /%AC/
020575  117  116  124                    .ASCII  /ONT/
020600  122  117  114                    .ASCII  /ROL/
020603  114  105  122                    .ASCII  /LER/
020606  040  122  101                    .ASCII  / RA/
020611  115  040  105                    .ASCII  /M E/
020614  122  122  117                    .ASCII  /RRO/
020617  122  040  050                    .ASCII  /R (/
020622  116  157  156                    .ASCII  /Non/
020625  055  160  141                    .ASCII  /-pa/
020630  162  151  164                    .ASCII  /rit/
020633  171  051  000                    .ASCII  /y)/<00>
020636  045  101  111          P.AJT:    .ASCII  /%AI/
020641  116  111  124                    .ASCII  /NIT/
020644  040  123  105                    .ASCII  / SE/
020647  121  125  105                    .ASCII  /QUE/
020652  116  103  105                    .ASCII  /NCE/
020655  040  105  122                    .ASCII  / ER/
020660  122  117  122                    .ASCII  /ROR/
020663  000                             .ASCII  <00>
020664  045  101  110          P.AJU:    .ASCII  /%AH/
020667  111  107  110                    .ASCII  /IGH/
020672  105  122  040                    .ASCII  /ER /
020675  114  105  126                    .ASCII  /LEV/
020700  105  114  040                    .ASCII  /EL /
020703  120  122  117                    .ASCII  /PRO/
020706  124  117  103                    .ASCII  /TOC/
020711  117  114  040                    .ASCII  /OL /
020714  111  116  103                    .ASCII  /INC/
020717  117  115  120                    .ASCII  /OMP/
020722  101  124  111                    .ASCII  /ATI/
020725  102  111  114                    .ASCII  /BIL/
020730  111  124  131                    .ASCII  /ITY/
```

```
020733  040     105     122                     .ASCII  / ER/
020736  122     117     122                     .ASCII  /ROR/
020741  000                                     .ASCII  <00>
020742  045     101     120     P.AJV:          .ASCII  /%AP/
020745  105     122     107                     .ASCII  /URG/
020750  105     057     120                     .ASCII  /E/<57>/P/
020753  117     114     114                     .ASCII  /OLL/
020756  040     110     101                     .ASCII  / HA/
020761  122     104     127                     .ASCII  /RDW/
020764  101     122     105                     .ASCII  /ARE/
020767  040     106     101                     .ASCII  / FA/
020772  111     114     125                     .ASCII  /ILU/
020775  122     105     000                     .ASCII  /RE/<00>
021000  045     101     115     P.AJW:          .ASCII  /%AM/
021003  101     120     120                     .ASCII  /APP/
021006  111     116     107                     .ASCII  /ING/
021011  040     122     105                     .ASCII  / RE/
021014  107     111     123                     .ASCII  /GIS/
021017  124     105     122                     .ASCII  /TER/
021022  040     122     105                     .ASCII  / RE/
021025  101     104     040                     .ASCII  /AD /
021030  106     101     111                     .ASCII  /FAI/
021033  114     125     122                     .ASCII  /LUR/
021036  105     040     050                     .ASCII  /E (/
021041  120     141     162                     .ASCII  /Par/
021044  151     164     171                     .ASCII  /ity/
021047  040     157     162                     .ASCII  / or/
021052  040     164     151                     .ASCII  / ti/
021055  155     145     157                     .ASCII  /meo/
021060  165     164     051                     .ASCII  /ut)/
021063  000                                     .ASCII  <00>
021064  017370'                 P.AI7:          .WORD   P.AJA
021066  017416                                  .WORD   P.AJB
021070  017500'                                 .WORD   P.AJC
021072  017562'                                 .WORD   P.AJD
021074  017630'                                 .WORD   P.AJE
021076  017666'                                 .WORD   P.AJF
021100  017724'                                 .WORD   P.AJG
021102  017772'                                 .WORD   P.AJH
021104  020042                                  .WORD   P.AJI
021106  020074'                                 .WORD   P.AJJ
021110  020164'                                 .WORD   P.AJK
021112  020214                                  .WORD   P.AJL
021114  020242'                                 .WORD   P.AJM
021116  020274'                                 .WORD   P.AJN
021120  020330'                                 .WORD   P.AJO
021122  020366'                                 .WORD   P.AJP
021124  020422'                                 .WORD   P.AJQ
021126  020506                                  .WORD   P.AJR
021130  020572'                                 .WORD   P.AJS
021132  020636'                                 .WORD   P.AJT
021134  020664                                  .WORD   P.AJU
021136  020742'                                 .WORD   P.AJV
```

```
021140  021000                          .WORD   P.AJW
021143      045     101     124 P.AJY:  .ASCII  /*AT/
021145      061     061     040         .ASCII  /11 /
021150      103     120     125         .ASCII  /CPU/
021153      040     106     101         .ASCII  / FA/
021156      111     114     125         .ASCII  /ILU/
021161      122     105     000         .ASCII  /RE/<00>
021164      045     101     116 P.AJZ:  .ASCII  /*AN/
021167      117     116     055         .ASCII  /ON-/
021172      120     101     122         .ASCII  /PAR/
021175      111     124     131         .ASCII  /ITY/
021200      040     122     101         .ASCII  / RA/
021203      115     040     105         .ASCII  /M E/
021206      122     122     117         .ASCII  /RRO/
021211      122     000     000         .ASCII  /R/<00><00>
021214      045     101     123 P.AKA:  .ASCII  /*AS/
021217      124     101     124         .ASCII  /TAT/
021222      105     040     115         .ASCII  /E M/
021225      101     103     110         .ASCII  /ACH/
021230      111     116     105         .ASCII  /INE/
021233      040     106     101         .ASCII  / FA/
021236      111     114     125         .ASCII  /ILU/
021241      122     105     040         .ASCII  /RE /
021244      055     040     124         .ASCII  /- T/
021247      061     061     040         .ASCII  /11 /
021252      101     104     104         .ASCII  /ADD/
021255      122     105     123         .ASCII  /RES/
021260      123     040     122         .ASCII  /S R/
021263      105     107     111         .ASCII  /EGI/
021266      123     124     105         .ASCII  /STE/
021271      122     000     000         .ASCII  /R/<00><00>
021274      045     101     123 P.AKB:  .ASCII  /*AS/
021277      124     101     124         .ASCII  /TAT/
021302      105     040     115         .ASCII  /E M/
021305      101     103     110         .ASCII  /ACH/
021310      111     116     105         .ASCII  /INE/
021313      040     106     101         .ASCII  / FA/
021316      111     114     125         .ASCII  /ILU/
021321      122     105     040         .ASCII  /RE /
021324      055     040     121         .ASCII  /- Q/
021327      055     102     125         .ASCII  / BU/
021332      123     040     101         .ASCII  /S A/
021335      104     104     122         .ASCII  /DDR/
021340      105     123     123         .ASCII  /ESS/
021343      040     122     105         .ASCII  / RE/
021346      107     111     123         .ASCII  /GIS/
021351      124     105     122         .ASCII  /TER/
021354      000     000                 .ASCII  <00><00>
021356      045     101     123 P.AKC:  .ASCII  /*AS/
021361      124     101     124         .ASCII  /TAT/
021364      105     040     115         .ASCII  /E M/
021367      101     103     110         .ASCII  /ACH/
021372      111     116     105         .ASCII  /INE/
```

```
021375      040     106     101                 .ASCII  / FA/
021400      111     114     125                 .ASCII  /ILU/
021403      122     105     040                 .ASCII  /RE /
021406      055     040     103                 .ASCII  /- C/
021411      122     103     040                 .ASCII  /RC /
021414      122     105     107                 .ASCII  /REG/
021417      111     123     124                 .ASCII  /IST/
021422      105     122     000                 .ASCII  /ER/<00>
021425      000                                 .ASCII  <00>
021426      045     101     123     P.AKD:      .ASCII  /%AS/
021431      124     101     124                 .ASCII  /TAT/
021434      105     040     115                 .ASCII  /E M/
021437      101     103     110                 .ASCII  /ACH/
021442      111     116     105                 .ASCII  /INE/
021445      040     106     101                 .ASCII  / FA/
021450      111     114     125                 .ASCII  /ILU/
021453      122     105     040                 .ASCII  /RE /
021456      055     040     123                 .ASCII  /- S/
021461      105     122     111                 .ASCII  /ERI/
021464      101     114     111                 .ASCII  /ALI/
021467      132     105     122                 .ASCII  /ZER/
021472      057     104     105                 .ASCII  <57>/DE/
021475      123     105     122                 .ASCII  /SER/
021500      111     101     114                 .ASCII  /IAL/
021503      111     132     105                 .ASCII  /IZE/
021506      122     040     122                 .ASCII  /R R/
021511      105     107     111                 .ASCII  /EGI/
021514      123     124     105                 .ASCII  /STE/
021517      122     000     000                 .ASCII  /R/<00><00>
021522      045     101     123     P.AKE:      .ASCII  /%AS/
021525      124     101     124                 .ASCII  /TAT/
021530      105     040     115                 .ASCII  /E M/
021533      101     103     110                 .ASCII  /ACH/
021536      111     116     105                 .ASCII  /INE/
021541      040     106     101                 .ASCII  / FA/
021544      111     114     125                 .ASCII  /ILU/
021547      122     105     040                 .ASCII  /RE /
021552      055     040     127                 .ASCII  /- W/
021555      122     117     116                 .ASCII  /RON/
021560      107     040     110                 .ASCII  /G H/
021563      101     122     104                 .ASCII  /ARD/
021566      127     101     122                 .ASCII  /WAR/
021571      105     040     126                 .ASCII  /E V/
021574      105     122     123                 .ASCII  /ERS/
021577      111     117     116                 .ASCII  /ION/
021602      000     000                         .ASCII  <00><00>
021604  021142'                     P.AJX:      .WORD   P.AJY
021606  021164'                                 .WORD   P.AJZ
021610  021214'                                 .WORD   P.AKA
021612  021274'                                 .WORD   P.AKB
021614  021356'                                 .WORD   P.AKC
021616  021426'                                 .WORD   P.AKD
021620  021522'                                 .WORD   P.AKE
```

```
021622  045  116  045  P.AKF:  .ASCII  /%N%/
021625  101  132  122         .ASCII  /AZR/
021630  121  101  040         .ASCII  /QA /
021633  104  105  126         .ASCII  /DEV/
021636  040  106  124         .ASCII  / FT/
021641  114  040  040         .ASCII  /L  /
021644  045  132  065         .ASCII  /%Z5/
021647  045  101  040         .ASCII  /%A /
021652  117  116  040         .ASCII  /ON /
021655  125  116  111         .ASCII  /UNI/
021660  124  040  045         .ASCII  /T %/
021663  132  062  045         .ASCII  /Z2%/
021666  101  040  124         .ASCII  /A T/
021671  123  124  040         .ASCII  /ST /
021674  060  060  061         .ASCII  /001/
021677  040  123  125         .ASCII  / SU/
021702  102  040  060         .ASCII  /B 0/
021705  060  060  040         .ASCII  /00 /
021710  120  103  072         .ASCII  /PC:/
021713  040  045  117         .ASCII  / %O/
021716  066  000               .ASCII  /6/<00>
021720  045  116  045  P.AKG:  .ASCII  /%N%/
021723  101  132  122         .ASCII  /AZR/
021726  121  101  040         .ASCII  /QA /
021731  110  122  104         .ASCII  /HRD/
021734  040  105  122         .ASCII  / ER/
021737  122  040  040         .ASCII  /R  /
021742  045  132  065         .ASCII  /%Z5/
021745  045  101  040         .ASCII  /%A /
021750  117  116  040         .ASCII  /ON /
021753  125  116  111         .ASCII  /UNI/
021756  124  040  045         .ASCII  /T %/
021761  132  062  045         .ASCII  /Z2%/
021764  101  040  124         .ASCII  /A T/
021767  123  124  040         .ASCII  /ST /
021772  060  060  061         .ASCII  /001/
021775  040  123  125         .ASCII  / SU/
022000  102  040  060         .ASCII  /B 0/
022003  060  060  040         .ASCII  /00 /
022006  120  103  072         .ASCII  /PC:/
022011  040  045  117         .ASCII  / %O/
022014  066  000               .ASCII  /6/<00>
022016  045  116  045  P.AKH:  .ASCII  /%N%/
022021  101  132  122         .ASCII  /AZR/
022024  121  101  040         .ASCII  /QA /
022027  123  106  124         .ASCII  /SFT/
022032  040  105  122         .ASCII  / ER/
022035  122  040  040         .ASCII  /R  /
022040  045  132  065         .ASCII  /%Z5/
022043  045  101  040         .ASCII  /%A /
022046  117  116  040         .ASCII  /ON /
022051  125  116  111         .ASCII  /UNI/
022054  124  040  045         .ASCII  /T %/
```

```
022057    132     062     045                    .ASCII  /Z2✱/
022062    101     040     124                    .ASCII  /A T/
022065    123     124     040                    .ASCII  /ST /
022070    060     060     061                    .ASCII  /001/
022073    040     123     125                    .ASCII  / SU/
022076    102     040     060                    .ASCII  /B 0/
022101    060     060     040                    .ASCII  /00 /
022104    120     103     072                    .ASCII  /PC:/
022107    040     045     117                    .ASCII  / ✱O/
022112    066     045     116                    .ASCII  /6✱N/
022115    000                                    .ASCII  <00>
022116    045     116     045     P.AKI:         .ASCII  /✱N✱/
022121    101     111     057                    .ASCII  /AI/<57>
022124    117     040     122                    .ASCII  /O R/
022127    105     121     125                    .ASCII  /EQU/
022132    105     123     124                    .ASCII  /EST/
022135    040     106     101                    .ASCII  / FA/
022140    111     114     105                    .ASCII  /ILE/
022143    104     045     116                    .ASCII  /D✱N/
022146    000     000                            .ASCII  <00><00>
022150    045     123     064     P.AKJ:         .ASCII  /✱S4/
022153    000                                    .ASCII  <00>
022154    045     116     000     P.AKK:         .ASCII  /✱N/<00>
022157    000                                    .ASCII  <00>
022160    045     101     040     P.AKL:         .ASCII  /✱A /
022163    055     040     000                    .ASCII  /- /<00>
022166    045     101     052     P.AKM:         .ASCII  /✱A*/
022171    040     000     000                    .ASCII  / /<00><00>
022174  000000C                   L$HWLEN::
                                                .WORD   <<L$NDHW-L$HWLEN>/2>
022176  172150    HWPT.IP.ADDR::
                                                .WORD   -5630
022200  000154    HWPT.VECTOR::
                                                .WORD   154
022202  000004    HWPT.BR.LEVEL::
                                                .WORD   4
022204  100340    HWPT.DISK::
                                                .WORD   -77440
022206  000000    HWPTS0.LBN::
                                                .WORD   0
022210  000000    HWPTS1.LBN::
                                                .WORD   0
022212  177777    HWPTE0.LBN::
                                                .WORD   -1
022214  000000    HWPTE1.LBN::
                                                .WORD   0
022216            L$NDHW::.BLKW   1
022220  000000C   L$SWLEN::
                                                .WORD   <<L$NDSW-L$SWLEN>/2>
022222  000040    SWP.ERROR::
                                                .WORD   40
022224  000000    SWP.XFER::
                                                .WORD   0
```

```
022226  054046                  SWP.FLAGS::
                                        .WORD   54046
022230  000000                  SWP.DPAT::
                                        .WORD   0
022232  000143                  SWP.RAT::
                                        .WORD   143
022234  000000                  SWP.TIME::
                                        .WORD   0
022236  000015                  DUPROUND::
                                        .WORD   13
022240  000020                  SWP.UCNT::
                                        .WORD   20
022242                          SWP.UDPAT::
                                        .BLKW   20
022302                          L$NDSW::.BLKW   1
022304  000000                  L$PROT::.WORD   0
022306  177777                          .WORD   -1
022310  000006                          .WORD   6


000000                                  .PSECT  $FFF$,  D  ,  GBL
000000                          CST::   .BLKW   53
000126                          CST.ADDR::
                                        .BLKW   1
000130                          DCT::   .BLKW   11
000152                          DCT.ADDR::
                                        .BLKW   1
000154                          RDRX.ADDR::
                                        .BLKW   1
000156                          IRDRX.ADDR::
                                        .BLKW   1
000160                          BST::   .BLKW   10
000200                          TALLY:: .BLKW   154
000530                          T.ADDR::.BLKW   1
000532                          DUPPKT::.BLKW   401
001534                          TRK.SGN::
001534  001                             .BYTE   1
001535  001                             .BYTE   1
001536  001                             .BYTE   1
001537  001                             .BYTE   1
001540  000020                  RDM.CNT::
                                        .WORD   20
001542                          RANDOM::.BLKW   20
001602                          C.ERR.TBL::
                                        .BLKW   1
001604                          MSCP.PKT::
                                        .BLKW   644
003314                          IPKT.ADDR::
                                        .BLKW   1
003316                          PKT.USE::
                                        .BLKW   6
003332                          RETPKT::.BLKW   260
```

```
                        RP.USE::.BLKW   4
                        RP.INDX::
                                .BLKW   1
                        RP.ADDR::
                                .BLKW   1
                        FLOG.PKT::
                                .BLKW   655
                        BUFF.ADDR::
                                .BLKW   10
                        BUFF.OWN::
                                .BLKW   4
                        IODQ::  .BLKW   4
                        IODQ.IN::
                                .BLKW   1
                        IODQ.OUT::
                                .BLKW   1
                        ENTRY.REASON::
                                .BLKB   1
                        FOP.FLAG::
                                .BLKB   1
                        DUP.FLAGS::
                                .BLKW   1
                        CCTLR::  .BLKW  1
                        CDISK::  .BLKW  1
                        CUOFF::  .BLKW  1
                        CTLR.CNT::
                                .BLKW   1
                        DUR::    .BLKW  1
                        QIO::    .BLKB  1
                                .EVEN
                        FREE.MEM.ADDR::
                                .BLKW   1
                        BYTS.PER.QIO::
                                .BLKW   1
                        ST.CODE::
                                .BLKW   1
                        SB.CODE::
                                .BLKW   1
                        STEP::   .BLKW  1
                        OF.RC::  .BLKW  1
                        SA.REG::.BLKW   1
                        CMD.TIME::
                                .BLKW   1
                        NEX::    .BLKW  1
                        CRN.LOW::
                                .BLKW   1
                        CRN.HIGH::
                                .BLKW   1
                        CREDIT.BAL::
                                .BLKW   1
                        NEXT.PKT.CODE::
                                .BLKB   1
                        HOURS::  .BLKB  1
```

```
005260                  MINUTES::
                                .BLKB   1
                                .EVEN
005262                  CLK.TICKS::
                                .BLKW   1
005264                  CLK.PRESENT::
                                .BLKB   1
005265                  HOF.FLAG::
                                .BLKB   1
005266                  S.PATTERN::
                                .BLKW   1
005270                  S.DUPPKT::
                                .BLKW   1
005272                  P.INDEX::
                                .BLKW   1
005274                  FORCED.ERROR::
                                .BLKB   1
                                .EVEN
005276                  FER.LBN::
                                .BLKW   1
006000                  FER.BC::.BLKW   1
006002                  INIT.OCCURED::
006002  000                     .BYTE   0
006003                  ADDR.VECT.OK::
006003  000                     .BYTE   0

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

100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                  BIT9=   1000
000400                  BIT8=   400
000200                  BIT7=   200
000100                  BIT6=   100
000040                  BIT5=   40
```

```
000020          BIT4==          20
000010          BIT3==          10
000004          BIT2==          4
000002          BIT1==          2
000001          BIT0==          1
000040          EF.START==      40
000037          EF.RESTART==    37
000036          EF.CONTINUE==   36
000035          EF.NEW==        35
000034          EF.PWR==        34
000340          PRIO7==         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
000126'         L$ERRTBL==      ERRTYP
022222'         L$SW==          L$SWLEN*2
022176'         L$HW==          L$HWLEN*2
000011'         L$DEPO==        L$REV*1
000136          HWQ1==          P.AAA
000152'         HWQ2==          P.AAB
000162'         HWQ3==          P.AAC
000174'         HWQ4==          P.AAD
000212'         HWQ5==          P.AAE
000262'         HWQ6A==         P.AAF
000334'         HWQ6B==         P.AAG
000410'         HWQ7A==         P.AAH
000460'         HWQ7B==         P.AAI
000530'         HWQ8==          P.AAJ
000602'         HWQ9==          P.AAK
000702'         HWQ10==         P.AAL
000732'         HWQ11==         P.AAM
000764'         SWQ1==          P.AAN
001006'         SWQ2==          P.AAO
001070'         SWQ4==          P.AAP
001112'         SWQ7==          P.AAQ
```

```
001164'  SWQ9**     P.AAR
001240'  SWQ10**    P.AAS
001304'  SWQ11**    P.AAT
001336'  SWQ12**    P.AAU
001434'  SWQ13**    P.AAV
001512'  SWQ14**    P.AAW
001564'  SWQ15**    P.AAX
001634'  SWQ17**    P.AAY
001732'  SWQ19**    P.AAZ
002022'  SWQ20**    P.ABA
002120'  SWQ21**    P.ABB
002222'  SWQ22**    P.ABC
002300'  SWQ23**    P.ABD
002372'  SWQ24**    P.ABE
002470'  SWQ25**    P.ABF
002570'  SWM1**     P.ABG
002660'  NULL**     P.ABH
002662'  DBM5**     P.ABI
002710'  DBM12**    P.ABJ
002764'  DBM15**    P.ABK
003014'  DBM18**    P.ABL
003066'  DBM19**    P.ABM
003152'  DBM20**    P.ABN
003226'  DBM21**    P.ABO
003310'  DBM22**    P.ABP
003354'  DBM23**    P.ABQ
003412'  DBM25**    P.ABR
003460'  DBM26**    P.ABS
003512'  DBM27**    P.ABT
003564'  DBM28A**   P.ABU
003624'  DBM28B**   P.ABV
003664'  DBM29**    P.ABW
003732'  DBM32**    P.ABX
004006'  DBM101**   P.ABY
004034'  DBM104**   P.ABZ
004076'  DBM105**   P.ACA
004134'  DBM107**   P.ACB
004172'  DBM108**   P.ACC
004262'  DBM109**   P.ACD
004342'  DBM111**   P.ACE
004442'  DBM112**   P.ACF
004544'  DBM120**   P.ACG
004636'  DBM121**   P.ACH
004726'  DU.MSG**   P.ACI
005446'  DU.RSN**   P.ACJ
005474'  MSG.01**   P.ACV
005526'  MSG.02**   P.ACW
005562'  MSG.03**   P.ACX
005614'  RPT1**     P.ACY
005700'  RPT2**     P.ACZ
005744'  RPT3**     P.ADA
006030'  RPT4**     P.ADB
006074'  RPT5**     P.ADC
```

```
006162'  RPT6**  P.ADD
006226'  RPT7**  P.ADE
006244'  RPT8**  P.ADF
006272'  RPT9**  P.ADG
006324'  RPT10**  P.ADH
006412'  RPT11**  P.ADI
006460'  EGS.01**  P.ADJ
006500'  EGS.02**  P.ADK
006572'  EGD.10**  P.ADL
006632'  EGD.11**  P.ADM
006656'  EGD.12**  P.ADN
006704'  EGD.13**  P.ADO
006732'  EGD.14**  P.ADP
006762'  EGD.15**  P.ADQ
007000'  EGD.16**  P.ADR
007030'  EGD.17**  P.ADS
007046'  EGD.18**  P.ADT
007066'  EGD.19**  P.ADU
007126'  EGD.20**  P.ADV
007214'  EGD.21**  P.ADW
007326'  EGD.22**  P.ADX
007366'  EGD.23**  P.ADY
007430'  EGD.24**  P.ADZ
007474'  EGH.30**  P.AEA
007520'  EBS.01**  P.AEB
007562'  EBD.10**  P.AEC
007622'  EBD.12**  P.AED
007670'  EBD.13**  P.AEE
007722'  EBD.14**  P.AEF
007762'  EBD.18**  P.AEG
010016'  EBD.19**  P.AEH
010076'  EBD.24**  P.AEI
010152'  EH.0**  P.AEJ
010210'  EH.1**  P.AEK
010246'  EH.2**  P.AEL
010306'  EH.3**  P.AEM
010344'  EH.4**  P.AEN
010370'  EH.5**  P.AEO
010420'  EH.6**  P.AEP
010450'  EH.7**  P.AEQ
010500'  EH.8**  P.AER
010534'  EH.9**  P.AES
010564'  EH.10**  P.AET
010614'  EH.12**  P.AEU
010650'  EH.13**  P.AEV
010722'  ERR.00**  P.AEW
011412'  ERR.C0D**  P.AEX
011446'  ELG.00**  P.AFM
012000'  ELG.FMI**  P.AFN
012012'  EX.SA**  P.AFT
012030'  EX.SC**  P.AFU
012060'  EX.SBO**  P.AFV
012064'  EX.SB**  P.AFW
```

```
012106'                 EX.CMD**        P.AFX
012126'                 EX.RD**         P.AFY
012136'                 EX.WRT**        P.AFZ
012146'                 EX.CMP**        P.AGA
012162'                 EX.ONL**        P.AGB
012174'                 EX.ACC**        P.AGC
012206'                 EX.OP**         P.AGD
012212'                 EX.BB**         P.AGE
012302'                 EX.BB1**        P.AGF
012376'                 EX.BBU**        P.AGG
012466'                 EX.LBN**        P.AGH
012524'                 EX.PBN**        P.AGI
012562'                 EX.LBR**        P.AGJ
012630'                 EX.LBW**        P.AGK
012676'                 EX.RBN**        P.AGL
012756'                 EX.CBC**        P.AGM
013022'                 EX.CBR**        P.AGN
013074'                 EX.CBW**        P.AGO
013146'                 EX.BC**         P.AGP
013222'                 EX.BD**         P.AGQ
013300'                 EX.BDR**        P.AGR
013370'                 EX.BDW**        P.AGS
013460'                 EX.RP**         P.AGT
013546'                 EX.WRD**        P.AGU
013556'                 EX.TIM**        P.AGV
013616'                 XX13**          P.AGW
013640'                 XX23**          P.AGX
013674'                 XX32**          P.AGY
013722'                 XX33**          P.AGZ
013750'                 XX34**          P.AHA
014030'                 CER.01**        P.AHB
014074'                 CER.02**        P.AHC
014150'                 SC.SDI**        P.AHD
014174'                 SC.CON**        P.AHE
014216'                 SC.DUP**        P.AHF
014246'                 SC.ONL**        P.AHG
014270'                 SC.SON**        P.AHH
014310'                 SC.UNK**        P.AHI
014370'                 SC.VOL**        P.AHJ
014450'                 SC.IOP**        P.AHK
014522'                 SC.DIS**        P.AHL
014614'                 SC.FER**        P.AHM
014702'                 SC.FE2**        P.AHN
014772'                 SC.ISH**        P.AHO
015052'                 SC.IS2**        P.AHP
015132'                 SC.DST**        P.AHQ
015206'                 SC.DS2**        P.AHR
015260'                 SC.ECC**        P.AHS
015342'                 SC.ECD**        P.AHT
015374'                 SC.RCT**        P.AHU
015414'                 SC.FUL**        P.AHV
015470'                 SC.STO**        P.AHW
015544'                 SC.FCT**        P.AHX
```

```
015612'                 SC.EC1**        P.AHY
015642'                 SC.EC2**        P.AHZ
015672'                 SC.EC3**        P.AIA
015724'                 SC.EC4**        P.AIB
015754'                 SC.EC5**        P.AIC
016004'                 SC.EC6**        P.AID
016034'                 SC.EC7**        P.AIE
016066'                 SC.EC8**        P.AIF
016120'                 SC.EC9**        P.AIG
016162'                 SC.SWP**        P.AIH
016222'                 SC.HWP**        P.AII
016262'                 SC.ODA**        P.AIJ
016312'                 SC.ODB**        P.AIK
016334                  SC.NXM**        P.AIL
016370'                 SC.PAR**        P.AIM
016424                  SC.CTO**        P.AIN
016476'                 SC.SDS**        P.AIO
016554'                 SC.EDC**        P.AIP
016614'                 SC.IDS**        P.AIQ
016664'                 SC.SRT**        P.AIR
016756'                 SC.SRI**        P.AIS
017044'                 SC.POE**        P.AIT
017100'                 SC.RDY**        P.AIU
017162'                 SC.CLK**        P.AIV
017210'                 SC.RSP**        P.AIW
017256'                 SC.SUR**        P.AIX
017306'                 SC.PSP**        P.AIY
021064'                 CNTR.ERR**      P.AIZ
021604'                 RDRX.ERR**      P.AJX
021622'                 DF.MSG**        P.AKF
021720'                 HRD.MSG**       P.AKG
022016'                 SFT.MSG**       P.AKH
022116'                 HRD.SUB**       P.AKI
022150'                 SPACE4**        P.AKJ
022154'                 CRLF**          P.AKK
022160'                 DASH**          P.AKL
022166'                 ASTERISK**      P.AKM
022176'                 DFPTBL**        L$HWLEN*2
022222'                 SFPTBL**        L$SWLEN*2


;                               PSECT SUMMARY
;
;       Psect Name              Words     Attributes
;         $CODE$                 4709       RO  ,  I  ,  LCL,  REL,  CON
;         $FFF$                  1538       RW  ,  D  ,  GBL,  REL,  CON


;                           Library Statistics
;
```

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
DISK$USER2:[POWERS]ZRQADO.L16;3	404	181	44	21	00:00.1

; COMMAND QUALIFIERS

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

```
;    3214  0     module ZRQAM0 (
;    3215  0
;    3216  0     %title 'RD/RX EXERCISER'
;    3217  0                 ident = 'V01.6',
;    3218  0                 addressing_mode (absolute),
;    3219  0                 environment (noeis)
;    3220  0                 ) =
;    3221  0
;    3222  1     begin
;    3223  1
;    3224  1     %sbttl 'DECLARATIONS'
;    3225  1
;    3226  1     library 'ZRQADO.L16';                    ! RDRX EXERCISER GLOBAL LIBRARY
;    3227  1
;    3228  1     require 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;    4719  1
;    4720  1     forward routine
;    4721  1         NEX_TRAP : L$ISR novalue,
;    4722  1         EMS_01 : novalue,
;    4723  1         EMS_TIM : novalue,
;    4724  1         EMS_DBN : NOVALUE,                                    !ZZZ
;    4725  1         EMS_BLK : NOVALUE,                                    !ZZZ
;    4726  1         SET_CPAR : novalue,
;    4727  1         SET_UPAR : novalue;
;    4728  1
;    4729  1     external
;    4730  1         CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    4731  1                         ! RUN-TIME CONTROLLER STATUS TABLES
;    4732  1         CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    4733  1                         ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    4734  1         DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    4735  1                         ! DRIVER CONTROLLER TABLES
;    4736  1         DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    4737  1                         ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    4738  1         RDRX_ADDR : ref rdrx field (RC_REG),
;    4739  1                         ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    4740  1         IRDRX_ADDR : ref rdrx field (RC_REG),
;    4741  1                         ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    4742  1         BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                 !ZZZ
;    4743  1                         !CONTAINS LBNS (HI + LO FIELDS) FOR SEQUENTIAL  !ZZZ
;    4744  1                         !I/O TRANSFER FOR EACH UNIT.                    !ZZZ
;    4745  1         TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    4746  1                         ! STATISTICS TABLES
;    4747  1         T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    4748  1                         ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    4749  1         DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    4750  1                         !INFO FROM RECEIVE AND SEND CMDS                 ZZZ
;    4751  1         TRK_SGN : VECTOR [MAX_UNITS, BYTE, SIGNED], !CURRENT TRACK DIRECTION ZZZ
;    4752  1         RDM_CNT : WORD,                             !NO OF RANDOM NOS   \\KEEP   ZZZ
;    4753  1         RANDOM : VECTOR [RDM_LEN, WORD],            !RANDOM NO TABLE    //TOGETHER ZZZ
;    4754  1         C_ERR_TBL : blockvector [MAX_CTLR, C_ERR_LEN, word] field (C_ERR_FIELDS),
;    4755  1                         ! STATISTICS TABLE FOR CONTROLLER ERRORS
;    4756  1         MSCP_PKT : blockvector [PKT_CNT, PKT_LEN, word] field (PKT_FIELDS),
```

```
;  4757  1                          ! MSCP PACKET POOL
;  4758  1          IPKT_ADDR : ref block [PKT_LEN, word] field (PKT_FIELDS),
;  4759  1                          ! ADDRESS OF AN MSCP PACKET (INTERRUPT PROCESSING)
;  4760  1          PKT_USE : vector [PKT_CNT, byte, signed],
;  4761  1                          ! MSCP PACKET POOL ALLOCATION TABLE
;  4762  1          RETPKT : blockvector [RP_CNT, RP_LEN, word] field (RP_FIELDS),
;  4763  1                          ! RETURN PACKET POOL
;  4764  1          RP_USE : vector [RP_CNT, byte, signed],
;  4765  1                          ! RETURN PACKET POOL ALLOCATION TABLE
;  4766  1          RP_INDX : word,      ! CURRENT RETURN PACKET INDEX
;  4767  1          RP_ADDR : ref block [RP_LEN, word] field (RP_FIELDS),
;  4768  1                          ! CURRENT RETURN PACKET ADDRESS
;  4769  1          ELOG_PKT : blockvector [EP_CNT + 1, EP_LEN, word] field (EP_FIELDS),
;  4770  1                          ! ERROR-LOG PACKET SAVE AREA
;  4771  1          BUFF_ADDR : vector [MAX_BUF_CNT],                    ! TABLE OF I/O BUFFER DESCRIPTORS
;  4772  1          BUFF_OWN : vector [MAX_BUF_CNT, byte, signed],       ! I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;  4773  1          IODQ : vector [IODQ_LEN, byte],                      ! I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;  4774  1          IODQ_IN : word,                                      ! I/O DONE QUEUE IN POINTER
;  4775  1          IODQ_OUT : word,                                     ! I/O DONE QUEUE OUT POINTER
;  4776  1          ENTRY_REASON : byte,                                 ! CURRENT OPERATOR COMMAND
;  4777  1          EOP_FLAG : byte,                                     ! END-OF-PASS FLAG
;  4778  1          DUP_FLAGS : WORD                                     !DUP FLAGS       ZZZ
;  4779  1          CCTLR : word,                                        ! NUMBER OF "CURRENT" CONTROLLER
;  4780  1          CDISK : word,                                        ! CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  4781  1          CUOFF : word,                                        ! CURRENT UNIT CST OFFSET
;  4782  1          CTLR_CNT : word,                                     ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;  4783  1          DUR : vector [MAX_UNITS, byte],                      ! DROP UNIT REASON
;  4784  1          QIO : vector [MAX_CTLR, byte],                       ! NUMBER OF OUTSTANDING QIOs PER CONTROLLER
;  4785  1          FREE_MEM_ADDR,                                       ! START OF FREE MEMORY
;  4786  1          BYTS_PER_QIO : word,                                 ! SIZE (BYTES) OF AN I/O BUFFER
;  4787  1          ST_CODE : word,                                      ! CURRENT STATUS CODE
;  4788  1          SB_CODE : word,                                      ! CURRENT SUB-CODE
;  4789  1          STEP : word,                                         ! CURRENT STEP IN HARD_INIT
;  4790  1          OF_RC : signed word,                                 ! OFFSET (0 OR 2) TO READ IP OR SA
;  4791  1          SA_REG : word,                                       ! STORAGE FOR SA REGISTER READS AND WRITES
;  4792  1          CMD_TIME : word,                                     ! COMMAND TIMEOUT VALUE (IN SECONDS)
;  4793  1          NEX : word,                                          ! NON-EXISTENT MEMORY TRAP INDICATOR
;  4794  1          CRN_LOW : word,                                      ! COMMAND REF NUMBER OF LAST COMMAND SENT
;  4795  1          CRN_HIGH : word,                                     ! COMMAND REF NUMBER (HI ORDER)
;  4796  1          CREDIT_BAL : word,                                   ! CREDIT BALANCE
;  4797  1          NEXT_PKT_USE : byte,                                 ! POINTER TO NEXT ENTRY IN PKT_USE TABLE
;  4798  1          HOURS: byte,                                         ! TIME OF DAY (HOURS)
;  4799  1          MINUTES : byte,                                      ! TIME OF DAY (MINUTES)
;  4800  1          CLK_TICKS : word,                                    ! TIME OF DAY (LINE-CLOCK TICKS)
;  4801  1          CLK_PRESENT : byte,                                  ! FLAG INDICATES IF LINE-CLOCK PRESENT
;  4802  1          HOE_FLAG : byte,                                     ! FLAG INDICATES IF "HALT ON ERROR" FLAG SET
;  4803  1          FORCED_ERROR : byte,                                 ! "FORCED ERROR" DETECTED IN LAST READ
;  4804  1          FER_LBN : word,                                      ! LBN OF THE "FORCED ERROR" BLOCK
;  4805  1          FER_BC : word,                                       ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;  4806  1          INIT_OCCURED : byte,                                 ! EXERCISER INITIALIZATION COMPLETE
;  4807  1          ADDR_VECT_OK : byte,                                 ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;  4808  1          DBMS,
;  4809  1          P_INDEX : SIGNED WORD,              !CURRENT MESSAGE PACKET INDEX      ZZZ
```

```
;    4810  1          S_PATTERN : WORD,                !PATTERN FOR DUP WRITES          ZZZ
;    4811  1          S_DUPPKT : WORD,                 !DBN BYTE COUNTER                ZZZ
;    4812  1          DBM107,
;    4813  1          DU_MSG,
;    4814  1          DU_RSN : vector [11],
;    4815  1          RPT1,
;    4816  1          RPT2,
;    4817  1          RPT3,
;    4818  1          RPT4,
;    4819  1          RPT5,
;    4820  1          RPT6,
;    4821  1          RPT7,
;    4822  1          RPT8,
;    4823  1          RPT9,
;    4824  1          RPT10,
;    4825  1          RPT11,
;    4826  1          MSG_01,
;    4827  1          EGS_01,
;    4828  1          EBS_01,
;    4829  1          EBD_10,
;    4830  1          EBD_12,
;    4831  1          EBD_13,
;    4832  1          EBD_14,
;    4833  1          EBD_18,
;    4834  1          EBD_19,
;    4835  1          EBD_24,
;    4836  1          ERR_00,
;    4837  1          ERR_COD : vector [14],
;    4838  1          ELG_00,
;    4839  1          ELG_FMT : vector [5],
;    4840  1          EX_TIM,
;    4841  1          XX13,                                                         !ZZZ
;    4842  1          XX23,                                                         !ZZZ
;    4843  1          XX32,                                                         !ZZZ
;    4844  1          XX33,                                                         !ZZZ
;    4845  1          XX34,                                                         !ZZZ
;    4846  1          EX_SA,
;    4847  1          EX_SC,
;    4848  1          EX_SBO,
;    4849  1          EX_SB,
;    4850  1          EX_RP,
;    4851  1          EX_WRD,
;    4852  1          EX_CMD,
;    4853  1          EX_RD,
;    4854  1          EX_WRT,
;    4855  1          EX_CMP,
;    4856  1          EX_ONL,
;    4857  1          EX_ACC,
;    4858  1          EX_OP,
;    4859  1          EX_BB,
;    4860  1          EX_BB1,
;    4861  1          EX_BB0,
;    4862  1          EX_LBN,
```

M9


```
;    4863  1          EX_PBN,
;    4864  1          EX_LBR,
;    4865  1          EX_LBW,
;    4866  1          EX_RBN,
;    4867  1          EX_CBC,
;    4868  1          EX_CBR,
;    4869  1          EX_CBW,
;    4870  1          EX_BC,
;    4871  1          EX_BD,
;    4872  1          EX_BDR,
;    4873  1          EX_BDW,
;    4874  1          SC_SDI,
;    4875  1          SC_CON,
;    4876  1          SC_DUP,
;    4877  1          SC_ONL,
;    4878  1          SC_SON,
;    4879  1          SC_UNK,
;    4880  1          SC_VOL,
;    4881  1          SC_IOP,
;    4882  1          SC_DIS,
;    4883  1          SC_FER,
;    4884  1          SC_FE2,
;    4885  1          SC_ISH,
;    4886  1          SC_IS2,
;    4887  1          SC_DST,
;    4888  1          SC_DS2,
;    4889  1          SC_ECC,
;    4890  1          SC_ECD,
;    4891  1          SC_RCT,
;    4892  1          SC_FUL,
;    4893  1          SC_576,
;    4894  1          SC_FCT,
;    4895  1          SC_SWP,
;    4896  1          SC_HWP,
;    4897  1          SC_EC1,
;    4898  1          SC_EC2,
;    4899  1          SC_EC3,
;    4900  1          SC_EC4,
;    4901  1          SC_EC5,
;    4902  1          SC_EC6,
;    4903  1          SC_EC7,
;    4904  1          SC_EC8,
;    4905  1          SC_EC9,
;    4906  1          SC_ODA,
;    4907  1          SC_ODB,
;    4908  1          SC_NXM,
;    4909  1          SC_PAR,
;    4910  1          SC_CTO,
;    4911  1          SC_SDS,
;    4912  1          SC_LDC,
;    4913  1          SC_IDS,
;    4914  1          SC_SR1,
;    4915  1          SC_SR1,
```

N9


```
;    4916  1          SC_POE,
;    4917  1          SC_RDY,
;    4918  1          SC_CLK,
;    4919  1          SC_RSP,
;    4920  1          SC_SUR,
;    4921  1          SC_PSP,
;    4922  1          CER_01,
;    4923  1          CER_02,
;    4924  1          CNTR_ERR : vector [23],
;    4925  1          RDRX_ERR : vector [7],
;    4926  1          SPACE4,
;    4927  1          CRLF,
     4928  1          DASH,
;    4929  1          ASTERISK,
;    4930  1          HWQ1,
;    4931  1          HWQ2,
;    4932  1          HWQ3,
;    4933  1          HWQ4,
;    4934  1          HWQ5,
;    4935  1          HWQ6A,
;    4936  1          HWQ6B,
;    4937  1          HWQ7A,
;    4938  1          HWQ7B,
;    4939  1          HWQ8,
;    4940  1          HWQ9,
;    4941  1          HWQ10,                                  !ZZZ
;    4942  1          HWQ11,                                  !ZZZ
;    4943  1          SWQ1,
;    4944  1          SWQ2,
;    4945  1          SWQ4,
;    4946  1          SWQ7,
;    4947  1          SWQ9,
;    4948  1          SWQ10,
;    4949  1          SWQ11,
;    4950  1          SWQ12,
;    4951  1          SWQ13,
;    4952  1          SWQ14,
;    4953  1          SWQ15,
;    4954  1          SWQ17,
;    4955  1          SWQ19,
;    4956  1          SWQ20,
;    4957  1          SWQ21,
;    4958  1          SWQ22,
;    4959  1          SWQ23,
;    4960  1          SWQ24,
;    4961  1          SWQ25,
;    4962  1          EH_0,                                   !ZZZ
;    4963  1          EH_1,                                   !ZZZ
;    4964  1          EH_2,                                   !ZZZ
;    4965  1          EH_3,                                   !ZZZ
;    4966  1          EH_4,                                   !ZZZ
;    4967  1          EH_5,                                   !ZZZ
;    4968  1          EH_6,                                   !ZZZ
```

```
;    4969  1          EH 7,
;    4970  1          EH 8,
;    4971  1          EH 9,
;    4972  1          EH 10,
;    4973  1          EH 12,
;    4974  1          EH 13,
;    4975  1          SWM1,
;    4976  1          NULL,
;    4977  1          SWP FLAGS : word,
;    4978  1          [$HIMEM,
;    4979  1          [$LOMEM,
;    4980  1          [$UNIT];
;    4981  1      !   0 BRK;
;    4982  1
;    4983  1      own
;    4984  1          TBL_SUC : vector [17] initial (NULL, SC_SDI, SC_CON, NULL, SC_DUP, NULL, NULL,
;    4985  1                    NULL, SC_ONL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, SC_SDN),
;    4986  1          TBL_OFL : vector [9] initial (SC_UNK, SC_VOL, SC_IOP, NULL, SC_DUP, NULL, NULL,
;    4987  1                    NULL, SC_DIS),
;    4988  1          TBL_MFE : vector [11] initial (SC_FER, NULL, SC_ISH, SC_DST, SC_EC9, SC_EC6,
;    4989  1                    SC_ECT, SC_ECC, SC_RCT, SC_FUL, SC_EC1),
;    4990  1          TBL_WPT : vector [3] initial (NULL, SC_SWP, SC_HWP),
;    4991  1          TBL_DAT : vector [16] initial (SC_FE2, NULL, SC_IS2, SC_DS2, SC_EC9, NULL, NULL,
;    4992  1                    SC_ECD, SC_EC1, SC_EC2, SC_EC3, SC_EC4, SC_EC5, SC_EC6, SC_EC7, SC_EC8),
;    4993  1          TBL_HST : vector [5] initial (NULL, SC_ODA, SC_ODB, SC_NXM, SC_PAR),
;    4994  1          TBL_CNT : vector [4] initial (SC_CTO, SC_SDS, SC_FDC, SC_IDS),
;    4995  1          TBL_DRV : vector [9] initial (NULL, SC_SRT, SC_SRI, SC_POE, SC_RDY, SC_CLR, SC_RSP,
;    4996  1                    SC_SUR, SC_PSP);
```

```
;    4997  1     %sbttl 'TYPE AND DESCRIPTION'
;    4998  1
;    4999  1     EQUALS;
;    5000  1
;    5001  1     DEVTYP (%ascic'RQDX1 or RUX50');                    ! NAME OF DEVICE SUPPORTED BY PROGRAM
;    5002  1     DESCRIPT (%ascic'RD/RX EXERCISER');     ! TEST DESCRIPTION
```

```
;    5003  1      %sbttl 'HARDWARE PARAMETER CODING SECTION'
;    5004  1
;    5005  1      !+
;    5006  1      ! THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5007  1      ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5008  1      ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5009  1      ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5010  1      ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5011  1      ! WITH THE OPERATOR.
;    5012  1      !
;    5013  1
;    5014  1      BGNHRD;
;    5015  1
;    5016  1      GPRMA (HWQ1, 0, 0, %o'160000', %o'177777', YES, 1);                    ! IP ADDRESS
;    5017  1      GPRMA (HWQ2, 2, 0, %o'4', %o'774', YES, 1);                            ! VECTOR
;    5018  1      GPRMD (HWQ3, 4, 0, %o'377', %o'0', %o'7', YES, 1);                     ! BR LEVEL
;    5019  1      GPRMD (HWQ4, 6, 0, %o'17', %decimal'0', %decimal'15', YES, 1);         ! RDRX DRIVE NUMBER
;    5020  1      GPRML (HWQ10, 6, %o'000040', YES, 1);                                  !ALSO RUN DUP EXERCISER      ZZZ
;    5021  1      XFERF (NODU);                                                          !                            ZZZ
;    5022  1      GPRML (HWQ11, 6, %o'000100', YES, 1);                                  !WRITE DIAG AREA             ZZZ
;    5023  1      $L (NODU);                                                             !                            ZZZ
;    5024  1      GPRML (HWQ5, 6, %o'000200', YES, 1);                                   !TEST ENTIRE CUSTOMER AREA?  ZZZ
;    5025  1      XFERT (TOQ8);                                                          !BR IF YES                   ZZZ
;    5026  1      GPRMD (HWQ6A, 8, 0, %o'177777', %decimal'0', %o'177777', YES, 1);      ! STARTING LBN LO            ZZZ
;    5027  1      GPRMD (HWQ6B, 10, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! STARTING LBN HI            ZZZ
;    5028  1      GPRMD (HWQ7A, 12, 0, %o'177777', GP$ATLO (8), %o'177777', YES, 1);     ! ENDING LBN LO              ZZZ
;    5029  1      GPRMD (HWQ7B, 14, 0, %o'177777', %decimal'0', %o'177777', YES, 1);     ! ENDING LBN HI              ZZZ
;    5030  1      $L (TOQ8);                                                             !                            ZZZ
;    5031  1      GPRML (HWQ8, 6, %o'100000', NO, 1);                                    ! EXER ON CUST DATA AREA
;    5032  1      XFERF (HWDONE);                                                        ! NO    DONE
;    5033  1      GPRML (HWQ9, 6, %o'100000', NO, 1);                                    ! ** WARNING / CONFIRM
;    5034  1      $L (HWDONE);
;    5035  1
;    5036  1      ENDHRD;
```

```
;    5037  1      %sbttl 'SOFTWARE PARAMETER CODING SECTION'
;    5038  1
;    5039  1      !+
;    5040  1      ! THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
;    5041  1      ! THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES.  THE
;    5042  1      ! MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
;    5043  1      ! INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES.  THE
;    5044  1      ! MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
;    5045  1      ! WITH THE OPERATOR.
;    5046  1      !-
;    5047  1
;    5048  1      BGNSFT;
;    5049  1
;    5050  1      !GPRML (SWQ16, 4, SWF_TRC, YES, 1);                          ! ENABLE DIAGNOSTIC TRACE
;    5051  1      GPRMD (SWQ24, 10, D, %o'177777', 0, 2359, YES, 1);           ! START TIME
;    5052  1      GPRMD (SWQ1, 0, D, %o'177777', 0, 65535, YES, 1);            ! ERROR LIMIT
;    5053  1      GPRMD (SWQ2, 2, D, %o'177777', 0, 99, YES, 1);               ! TRANSFER LIMIT
;    5054  1      GPRMD (SWQ17, 8, D, %o'177777', 0, 100, YES, 1);             ! PERCENT OF RD OPERATIONS
;    5055  1      GPRML (SWQ15, 4, SWF_CST, YES, 1);                           ! CLEAR STATISTICAL TABLES ?
;    5056  1      GPRML (SWQ20, 4, SWF_FER, YES, 1);                           ! REWRITE BLOCKS WHEN "FORCED ERROR" BIT SET?
;    5057  1      GPRML (SWQ23, 4, SWF_BLK, YES, 1);                           ! HALT ON BAD-BLOCK TYPE ERRORS WITH 'HOE' FLAG?
;    5058  1      GPRML (SWQ21, 4, SWF_HRD, YES, 1);                           ! HALT ON HARD ERRORS WITH 'HOE' FLAG SET?
;    5059  1      GPRML (SWQ22, 4, SWF_SFT, YES, 1);                           ! HALT ON SOFT ERRORS WITH 'HOE' FLAG SET?
;    5060  1      GPRML (SWQ25, 4, SWF_TRY, YES, 1);                           ! COUNT EACH RETRY AS ANOTHER SOFT-ERROR?
;    5061  1      GPRML (SWQ4, 4, SWF_RDM, YES, 1);                            ! RANDOM SEEK MODE ?
;    5062  1      XFERF (SW1);                                                 ! IF NO, DO NEXT QUESTION
;    5063  1      XFER (SW2);                                                  !
;    5064  1      $L (SW1);                                                    !
;    5065  1      GPRML (SWQ19, 4, SWF_SEQ, YES, 1);                           ! RANDOM OR SEQUENTIAL SELECTION OF DRIVES
;    5066  1      $L (SW2);                                                    !
;    5067  1      GPRML (SWQ7, 4, SWF_CRC, YES, 1);                            ! READ-COMPARES AT CONTROLLER ?
;    5068  1      DISPLAY (SWM1);                                              ! REMAINING QUESTIONS ONLY APPLY ...
;    5069  1      GPRML (SWQ9, 4, SWF_CWC, YES, 1);                            ! WRITE-COMPARES AT CONTROLLER ?
;    5070  1      XFERF (SW3);                                                 ! IF NO, DO NEXT QUESTION
;    5071  1      XFER (SW4);                                                  !
;    5072  1      $L (SW3);                                                    !
;    5073  1      GPRML (SWQ10, 4, SWF_HWC, YES, 1);                           ! CHECK WRITES AT HOST BY READING ?
;    5074  1      $L (SW4);                                                    !
;    5075  1      GPRML (SWQ11, 4, SWF_UDP, YES, 1);                           ! USER-DEFINED DATA PATTERN ?
;    5076  1      XFERF (SW5);                                                 ! IF NO, DO NEXT QUESTION
;    5077  1      XFER (SW6);                                                  !
;    5078  1      $L (SW5);                                                    !
;    5079  1      GPRMD (SWQ12, 6, D, %o'177777', 0, DP CNT, YES, 1);          ! SELECT PRE-DEFINED DATA PATTERN
;    5080  1      XFER (SW7);                                                  ! DONE
;    5081  1      $L (SW6);                                                    !
;    5082  1      GPRMD (SWQ13, 12, D, %o'177777', 1, MAX UDP_CNT, YES, 1);    ! NO. OF WORDS IN USER DATA PATTERN
;    5083  1      GPRMD (SWQ14, 14, O, %o'177777', 0, %o'177777', NO, 12);     ! PATTERN VALUES
;    5084  1      $L (SW7);                                                    !
;    5085  1
;    5086  1      ENDSFT;
```

```
;    5087  1
;    5088  1
;    5089  1     %sbttl 'REPORT CODING SECTION'
;    5090  1
;    5091  1
;    5092  1     !+
;    5093  1     ! THE REPORT CODING SECTION CONTAINS THE
;    5094  1     ! "PRINTS" CALLS THAT GENERATE STATISTICAL REPORTS.
;    5095  1     !-
;    5096  1
;    5097  1
;    5098  2     BGNRPT;
;    5099  2
;    5100  2     local
;    5101  2         CUR_PRIORITY : word;
;    5102  2
;    5103  2     GETPRI (CUR_PRIORITY);
;    5104  2     SETPRI (PRIO4);
;    5105  2
;    5106  2     PRINTS (RPT1);
;    5107  2     PRINTS (RPT2);
;    5108  2     PRINTS (RPT3);
;    5109  2     PRINTS (RPT4);
;    5110  2     PRINTS (RPT5);
;    5111  2     PRINTS (RPT6);
;    5112  2
;    5113  2     incr CTLR from 0 to MAX_CTLR - 1 do
;    5114  2
;    5115  3         begin
;    5116  3         SET_CPAR (.CTLR);
;    5117  3
;    5118  3         incr DISK from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    5119  3
;    5120  4             begin
;    5121  4             SET_UPAR (.DISK);
;    5122  4
;    5123  4
;    5124  4             if .CST_ADDR [.DISK + OF_DATA, D_PRES] eql PRESENT
;    5125  4             then
;    5126  4
;    5127  5                 begin
; P  5128  5                 PRINTS (RPT7,
;    5129  5                     .L$LUN, .CST_ADDR [.DISK + OF_DATA, D_DISK_NUM], CST [.CTLR, .DISK + OF_NAME_0, D_NAME_0]);
; P  5130  5                 PRINTS (RPT8,
; P  5131  5                     .T_ADDR [TOT_READS_HI], .T_ADDR [TOT_READS_LO],
;    5132  5                     .T_ADDR [MTOT_BYT_RED], .T_ADDR [TOT_BYT_RED_HI], .T_ADDR [TOT_BYT_RED_LO]);
; P  5133  5                 PRINTS (RPT8,
; P  5134  5                     .T_ADDR [TOT_WRITES_HI], .T_ADDR [TOT_WRITES_LO],
;    5135  5                     .T_ADDR [MTOT_BYT_WRT], .T_ADDR [TOT_BYT_WRT_HI], .T_ADDR [TOT_BYT_WRT_LO]);
; P  5136  5                 PRINTS (RPT9,
; P  5137  5                     .T_ADDR [ERR_HRD_SEK], .T_ADDR [ERR_HRD_DAT], .T_ADDR [ERR_HRD_DRV], .T_ADDR [ERR_HRD_HST],
;    5138  5                     .T_ADDR [ERR_SFT_SEK], .T_ADDR [ERR_SFT_DAT], .T_ADDR [ERR_SFT_DRV], .T_ADDR [ERR_SFT_HST]);
;    5139  4                 end;
```

```
;    5140  3              end;
;    5141  3
;    5142  3
;    5143  3          if .CST [.CTLR, STATE] eql PRESENT
;    5144  3          then
;    5145  3
;    5146  4              begin
;    5147  4              PRINTS (RPT10);
;    5148  4              PRINTS (RPT11, .C_ERR_TBL [.CTLR, C_ERR_HRD], .C_ERR_TBL [.CTLR, C_ERR_SFT]);
;    5149  3              end;
;    5150  3
;    5151  3          PRINTS (CRLF);
;    5152  3
;    5153  2          end;
;    5154  2
;    5155  2      SETPRI (.CUR_PRIORITY);
;    5156  2
;    5157  1      ENDRPT;


                                        .TITLE  ZRQAM2 RD/RX EXERCISER
                                        .IDENT  /V01.6/
                                        .ENABL  AMA


000000                                  .PSECT  $CODE$,  RO
000000      122     121     104     L$DVTYP::
000003      130     061     040             .ASCII  /RQD/
000006      157     162     040             .ASCII  /X1 /
000011      122     125     130             .ASCII  /or /
000014      065     060     000             .ASCII  /RUX/
000017      000                             .ASCII  /50/<00>
000020      122     104     057             .ASCII  <00>
000023      122     130     040     L$DESC::.ASCII  /RD/<57>
000026      105     130     105             .ASCII  /RX /
000031      122     103     111             .ASCII  /EXE/
000034      123     105     122             .ASCII  /RCI/
000037      000                             .ASCII  /SER/
000040                                      .ASCII  <00>
000042  000000C                             .BLKB   2
                                    L$HRDLN::
000044  000031                              .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000046  000000G                     GP$1::  .WORD   31
000050  160000                              .WORD   HWQ1
000052  177777                              .WORD   -20000
000054  001031                              .WORD   1
000056  000000G                     GP$2::  .WORD   1031
000060  000004                              .WORD   HWQ2
000062  000774                              .WORD   4
000064  002032                              .WORD   774
000066  000000G                     GP$3::  .WORD   2032
000070  000377                              .WORD   HWQ3
                                            .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  0000000G                        .WORD   HWQ10
000114  000040                          .WORD   40
000116  000000C                 $NODU:  .WORD   <<<<$LNODU-$NODU>*400>+4>+40>
000120  003130                  GP$6::  .WORD   3130
000122  0000000G                        .WORD   HWQ11
000124  000100                          .WORD   100
000126  001004                  $LNODU: .WORD   1004
000130  003130                  GP$7::  .WORD   3130
000132  0000000G                        .WORD   HWQ5
000134  000200                          .WORD   200
000136  000000C                 $TOQ8:  .WORD   <<<<$LTOQ8-$TOQ8>*400>+4>+20>
000140  004032                  GP$8::  .WORD   4032
000142  0000000G                        .WORD   HWQ6A
000144  177777                          .WORD   -1
000146  000000                          .WORD   0
000150  177777                          .WORD   -1
000152  005032                  GP$9::  .WORD   5032
000154  0000000G                        .WORD   HWQ6B
000156  177777                          .WORD   -1
000160  000000                          .WORD   0
000162  177777                          .WORD   -1
000164  006432                  GP$10:: .WORD   6432
000166  0000000G                        .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  0000000G                        .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  0000000G                        .WORD   HWQ8
000220  100000                          .WORD   100000
000222  000000C                 $HWDONE:.WORD   <<<<$LHWDONE-$HWDONE>*400>+4>+40>
000224  003120                  GP$13:: .WORD   3120
000226  0000000G                        .WORD   HWQ9
000230  100000                          .WORD   -100000
000232  001004                  $LHWDONE:
                                        .WORD   1004
000234                          L$NDHRD::
                                        .BLKW   1
000236  000000C                 L$SFTLN::
```

```
                                        .WORD   <<<L$NDSFT-L$SFTLN>/2>-1>
000240  005052                GP$14::   .WORD   5052
000242  000000G                         .WORD   SWQ24
000244  177777                         .WORD   -1
000246  000000                         .WORD   0
000250  004467                         .WORD   4467
000252  000052                GP$15::   .WORD   52
000254  000000G                         .WORD   SWQ1
000256  177777                         .WORD   -1
000260  000000                         .WORD   0
000262  177777                         .WORD   -1
000264  001052                GP$16::   .WORD   1052
000266  000000G                         .WORD   SWQ2
000270  177777                         .WORD   -1
000272  000000                         .WORD   0
000274  000143                         .WORD   143
000276  004052                GP$17::   .WORD   4052
000300  000000G                         .WORD   SWQ17
000302  177777                         .WORD   -1
000304  000000                         .WORD   0
000306  000144                         .WORD   144
000310  002130                GP$18::   .WORD   2130
000312  000000G                         .WORD   SWQ15
000314  000200                         .WORD   200
000316  002130                GP$19::   .WORD   2130
000320  000000G                         .WORD   SWQ20
000322  004000                         .WORD   4000
000324  002130                GP$20::   .WORD   2130
000326  000000G                         .WORD   SWQ23
000330  040000                         .WORD   40000
000332  002130                GP$21::   .WORD   2130
000334  000000G                         .WORD   SWQ21
000336  010000                         .WORD   10000
000340  002130                GP$22::   .WORD   2130
000342  000000G                         .WORD   SWQ22
000344  020000                         .WORD   20000
000346  002130                GP$23::   .WORD   2130
000350  000000G                         .WORD   SWQ25
000352  100000                         .WORD    100000
000354  002130                GP$24::   .WORD   2130
000356  000000G                         .WORD   SWQ4
000360  000002                         .WORD   2
000362  000000C                $SW1:    .WORD   <<<<$LSW1  $SW1>*400>.4>.40>
000364  000000C                $SW2:    .WORD   <<<$LSW2-$SW2>*400>.4>
000366  001004                 $LSW1:   .WORD   1004
000370  002130                GP$25::   .WORD   2130
000372  000000G                         .WORD   SWQ19
000374  001000                         .WORD   1000
000376  001004                 $LSW2:   .WORD   1004
000400  002130                GP$26::   .WORD   2130
000402  000000G                         .WORD   SWQ7
000404  000004                         .WORD   4
000406  000003                GP$DISP::
```

```
000410  000000G                         .WORD   3
000412  002130                          .WORD   SWM1
000414  000000G                 GP$27:: .WORD   2130
000416  000020                          .WORD   SWQ9
000420  000000C                         .WORD   20
000422  000000C                 $SW3:   .WORD   <<<<$LSW3-$SW3>+400>+4>+40>
000424  001004                  $SW4:   .WORD   <<<$LSW4-$SW4>+400>+4>
000426  002130                  $LSW3:  .WORD   1004
000430  000000G                 GP$28:: .WORD   2130
000432  000040                          .WORD   SWQ10
000434  001004                          .WORD   40
000436  002130                  $LSW4:  .WORD   1004
000440  000000G                 GP$29:: .WORD   2130
000442  000100                          .WORD   SWQ11
000444  000000C                         .WORD   100
000446  000000C                 $SW5:   .WORD   <<<<$LSW5-$SW5>+400>+4>+40>
000450  001004                  $SW6:   .WORD   <<<$LSW6-$SW6>+400>+4>
000452  003052                  $LSW5:  .WORD   1004
000454  000000G                 GP$30:: .WORD   3052
000456  177777                          .WORD   SWQ12
000460  000000                          .WORD   -1
000462  000025                          .WORD   0
000464  000000C                         .WORD   25
000466  001004                  $SW7:   .WORD   <<<$LSW7-$SW7>+400>+4>
000470  006052                  $LSW6:  .WORD   1004
000472  000000G                 GP$31:: .WORD   6052
000474  177777                          .WORD   SWQ13
000476  000001                          .WORD   -1
000500  000020                          .WORD   1
000502  007222                          .WORD   20
000504  000000G                 GP$32:: .WORD   7222
000506  177777                          .WORD   SWQ14
000510  000000                          .WORD   -1
000512  177777                          .WORD   0
000514  000006                          .WORD   -1
000516  001004                          .WORD   6
000520                          $LSW7:  .WORD   1004
                                L$NDSFT::
                                        .BLKW   1


000000                                  .PSECT  $OWN$,  D
000000  000000G                 TBL.SUC:.WORD   NULL
000002  000000G                         .WORD   SC.SDI
000004  000000G                         .WORD   SC.CON
000006  000000G                         .WORD   NULL
000010  000000G                         .WORD   SC.DUP
000012  000000G                         .WORD   NULL
000014  000000G                         .WORD   NULL
000016  000000G                         .WORD   NULL
000020  000000G                         .WORD   SC.ONL
000022  000000G                         .WORD   NULL
```

```
000024  000000G                         .WORD   NULL
000026  000000G                         .WORD   NULL
000030  000000G                         .WORD   NULL
000032  000000G                         .WORD   NULL
000034  000000G                         .WORD   NULL
000036  000000G                         .WORD   NULL
000040  000000G                         .WORD   SC.SON
000042  000000G                 TBL.OFL:.WORD   SC.UNK
000044  000000G                         .WORD   SC.VOL
000046  000000G                         .WORD   SC.IOP
000050  000000G                         .WORD   NULL
000052  000000G                         .WORD   SC.DUP
000054  000000G                         .WORD   NULL
000056  000000G                         .WORD   NULL
000060  000000G                         .WORD   NULL
000062  000000G                         .WORD   SC.DIS
000064  000000G                 TBL.MFE:.WORD   SC.FER
000066  000000G                         .WORD   NULL
000070  000000G                         .WORD   SC.ISH
000072  000000G                         .WORD   SC.DST
000074  000000G                         .WORD   SC.EC9
000076  000000G                         .WORD   SC.576
000100  000000G                         .WORD   SC.FCT
000102  000000G                         .WORD   SC.ECC
000104  000000G                         .WORD   SC.RCT
000106  000000G                         .WORD   SC.FUL
000110  000000G                         .WORD   SC.EC1
000112  000000G                 TBL.WPT:.WORD   NULL
000114  000000G                         .WORD   SC.SWP
000116  000000G                         .WORD   SC.HWP
000120  000000G                 TBL.DAT:.WORD   SC.FE2
000122  000000G                         .WORD   NULL
000124  000000G                         .WORD   SC.IS2
000126  000000G                         .WORD   SC.DS2
000130  000000G                         .WORD   SC.EC9
000132  000000G                         .WORD   NULL
000134  000000G                         .WORD   NULL
000136  000000G                         .WORD   SC.ECD
000140  000000G                         .WORD   SC.EC1
000142  000000G                         .WORD   SC.EC2
000144  000000G                         .WORD   SC.EC3
000146  000000G                         .WORD   SC.EC4
000150  000000G                         .WORD   SC.EC5
000152  000000G                         .WORD   SC.EC6
000154  000000G                         .WORD   SC.EC7
000156  000000G                         .WORD   SC.EC8
000160  000000G                 TBL.HST:.WORD   NULL
000162  000000G                         .WORD   SC.ODA
000164  000000G                         .WORD   SC.ODB
000166  000000G                         .WORD   SC.NXM
000170  000000G                         .WORD   SC.PAR
000172  000000G                 TBL.CNT:.WORD   SC.CTO
000174  000000G                         .WORD   SC.SDS
```

```
000176  000000G                         .WORD   SC.EDC
000200  000000G                         .WORD   SC.IDS
000202  000000G               TBL.DRV:  .WORD   NULL
000204  000000G                         .WORD   SC.SRT
000206  000000G                         .WORD   SC.SRI
000210  000000G                         .WORD   SC.POE
000212  000000G                         .WORD   SC.RDY
000214  000000G                         .WORD   SC.CLK
000216  000000G                         .WORD   SC.RSP
000220  000000G                         .WORD   SC.SUR
000222  000000G                         .WORD   SC.PSP


                        .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                        .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                        .GLOBL  DUPPKT, TRK.SGN, RDM.CNT, RANDOM
                        .GLOBL  C.ERR.TBL, MSCP.PKT, IPKT.ADDR
                        .GLOBL  PKT.USE, RETPKT, RP.USE, RP.INDX
                        .GLOBL  RP.ADDR, ELOG.PKT, BUFF.ADDR, BUFF.OWN
                        .GLOBL  IODQ, IODQ.IN, IODQ.OUT, ENTRY.REASON
                        .GLOBL  EOP.FLAG, DUP.FLAGS, CCTLR, CDISK
                        .GLOBL  CUOFF, CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
                        .GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
                        .GLOBL  STEP, OF.RC, SA.REG, CMD.CODE
                        .GLOBL  NEX, CRN.LOW, CRN.HIGH, CREDIT.BAL
                        .GLOBL  NEXT.PKT.USE, HOURS, MINUTES, CLK.TICKS
                        .GLOBL  CLK.PRESENT, HOE.FLAG, FORCED.ERROR
                        .GLOBL  FER.LBN, FER.BC, INIT.OCCURED
                        .GLOBL  ADDR.VECT.OK, DBM5, P.INDEX, S.PATTERN
                        .GLOBL  S.DUPPKT, DBM107, DU.MSG, DU.RSN
                        .GLOBL  RPT1, RPT2, RPT3, RPT4, RPT5, RPT6
                        .GLOBL  RPT7, RPT8, RPT9, RPT10, RPT11
                        .GLOBL  MSG.01, EGS.01, EBS.01, EBD.10
                        .GLOBL  EBD.12, EBD.13, EBD.14, EBD.18
                        .GLOBL  EBD.19, EBD.24, ERR.00, ERR.COD
                        .GLOBL  ELG.00, ELG.FMT, EX.TIM, XX13
                        .GLOBL  XX23, XX32, XX33, XX34, EX.SA
                        .GLOBL  EX.SC, EX.SB0, EX.SB, EX.RP, EX.WRD
                        .GLOBL  EX.CMD, EX.RD, EX.WRT, EX.CMP
                        .GLOBL  EX.ONL, EX.ACC, EX.OP, EX.BB, EX.BB1
                        .GLOBL  EX.BB0, EX.LBN, EX.PBN, EX.LBR
                        .GLOBL  EX.LBW, EX.RBN, EX.CBC, EX.CBR
                        .GLOBL  EX.CBW, EX.BC, EX.BD, EX.BDR, EX.BDW
                        .GLOBL  SC.SDI, SC.CON, SC.DUP, SC.ONL
                        .GLOBL  SC.SON, SC.UNK, SC.VOL, SC.IOP
                        .GLOBL  SC.DIS, SC.FER, SC.FE2, SC.ISH
                        .GLOBL  SC.IS2, SC.DST, SC.DS2, SC.ECC
                        .GLOBL  SC.ECD, SC.RCT, SC.FUL, SC.576
                        .GLOBL  SC.FCT, SC.SWP, SC.HWP, SC.EC1
                        .GLOBL  SC.EC2, SC.EC3, SC.EC4, SC.EC5
                        .GLOBL  SC.EC6, SC.EC7, SC.EC8, SC.EC9
                        .GLOBL  SC.ODA, SC.ODB, SC.NXM, SC.PAR
                        .GLOBL  SC.CTO, SC.SDS, SC.EDC, SC.IDS
```

```
.GLOBL  SC.SRT, SC.SRI, SC.POE, SC.RDY
.GLOBL  SC.CLK, SC.RSP, SC.SUR, SC.PSP
.GLOBL  CER.01, CER.02, CNTR ERR, RDRX.ERR
.GLOBL  SPACE4, CRLF, DASH, ASTERISK, HWQ1
.GLOBL  HWQ2, HWQ3, HWQ4, HWQ5, HWQ6A
.GLOBL  HWQ6B, HWQ7A, HWQ7B, HWQ8, HWQ9
.GLOBL  HWQ10, HWQ11, SWQ1, SWQ2, SWQ4
.GLOBL  SWQ7, SWQ9, SWQ10, SWQ11, SWQ12
.GLOBL  SWQ13, SWQ14, SWQ15, SWQ17, SWQ19
.GLOBL  SWQ20, SWQ21, SWQ22, SWQ23, SWQ24
.GLOBL  SWQ25, EH.0, EH.1, EH.2, EH.3
.GLOBL  EH.4, EH.5, EH.6, EH.7, EH.8, EH.9
.GLOBL  EH.10, EH.12, EH.13, SWM1, NULL
.GLOBL  SWP.FLAGS, L$HIMEM, L$LUN, L$UNIT

100000          BIT15==         -100000
040000          BIT14==         40000
020000          BIT13==         20000
010000          BIT12==         10000
004000          BIT11==         4000
002000          BIT10==         2000
001000          BIT09==         1000
000400          BIT08==         400
000200          BIT07==         200
000100          BIT06==         100
000040          BIT05==         40
000020          BIT04==         20
000010          BIT03==         10
000004          BIT02==         4
000002          BIT01==         2
000001          BIT00==         1
001000          BIT9==          1000
000400          BIT8==          400
000200          BIT7==          200
000100          BIT6==          100
000040          BIT5==          40
000020          BIT4==          20
000010          BIT3==          10
000004          BIT2==          4
000002          BIT1==          2
000001          BIT0==          1
000040          EF.START==      40
000037          EF.RESTART==    37
000036          EF.CONTINUE==   36
000035          EF.NEW==        35
000034          EF.PWR==        34
000340          PRI07==         340
000300          PRI06==         300
000240          PRI05==         240
000200          PRI04==         200
000140          PRI03==         140
000100          PRI02==         100
```

```
000040                  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                  PNI==           1000
002000                  PRI==           2000
004000                  IXE==           4000
010000                  IBE==           10000
020000                  IER==           20000
040000                  LOE==           40000
100000                  HOE==           -100000
000044'                 L$HARD==        L$HRDLN+2
000240'                 L$SOFT==        L$SFTLN+2


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

000000  004137  000000G  LRPT:  JSR     R1,$SAVE4                       ;                       5086
000004  104440                 TRAP    40                              ;                       5105
000006  010004                 MOV     R0,R4                           ; *.CUR.PRIORITY
000010  012700  000200         MOV     #200,R0                         ;                       5104
000014  104441                 TRAP    41
000016  012746  000000G        MOV     #RPT1,-(SP)                     ;                       5106
000022  012746  000001         MOV     #1,-(SP)
000026  010600                 MOV     SP,R0                           ; SP,*
000030  104416                 TRAP    16
000032  012716  000000G        MOV     #RPT2,(SP)                      ;                       5107
000036  012746  000001         MOV     #1,-(SP)
000042  010600                 MOV     SP,R0                           ; SP,*
000044  104416                 TRAP    16
000046  012716  000000G        MOV     #RPT3,(SP)                      ;                       5108
000052  012746  000001         MOV     #1,-(SP)
000056  010600                 MOV     SP,R0                           ; SP,*
000060  104416                 TRAP    16
000062  012716  000000G        MOV     #RPT4,(SP)                      ;                       5109
000066  012746  000001         MOV     #1,-(SP)
000072  010600                 MOV     SP,R0                           ; SP,*
000074  104416                 TRAP    16
000076  012716  000000G        MOV     #RPT5,(SP)                      ;                       5110
000102  012746  000001         MOV     #1,-(SP)
000106  010600                 MOV     SP,R0                           ; SP,*
000110  104416                 TRAP    16
000112  012716  000000G        MOV     #RPT6,(SP)                      ;                       5111
000116  012746  000001         MOV     #1,-(SP)
000122  010600                 MOV     SP,R0                           ; SP,*
000124  104416                 TRAP    16
000126  005003                 CLR     R3                              ; CTLR                  5113
```

```
000130  010516                  1$:     MOV     R5,(SP)                         ; CTLR,*                        5116
000132  004737  0000000V                JSR     PC,SET.CPAR
000136  012702  000003                  MOV     #3,R2                           ; *,DISK                        5118
000142  010216                  2$:     MOV     R2,(SP)                         ; DISK,*                        5121
000144  004737  0000000V                JSR     PC,SET.UPAR
000150  010201                          MOV     R2,R1                           ; DISK,*                        5124
000152  006301                          ASL     R1
000154  063701  0000006                 ADD     CST.ADDR,R1
000160  032711  040000                  BIT     #40000,(R1)
000164  001535                          BEQ     3$
000166  010516                          MOV     R5,(SP)                         ; CTLR,*                        5129
000170  012746  000055                  MOV     #55,-(SP)
000174  004737  0000000G                JSR     PC,@L$MUL
000200  060200                          ADD     R2,R0                           ; DISK,*
000202  006300                          ASL     R0
000204  062700  0000006                 ADD     #CST,R0
000210  010016                          MOV     R0,(SP)
000212  062716  000012                  ADD     #12,(SP)
000216  111146                          MOVB    (R1),-(SP)
000220  042716  177760                  BIC     #177760,(SP)
000224  013746  0000004                 MOV     L$LUN,-(SP)
000230  012746  0000006                 MOV     #RPT7,-(SP)
000234  012746  000004                  MOV     #4,-(SP)
000240  010600                          MOV     SP,R0                           ; SP,*
000242  104416                          TRAP    16
000244  013700  0000006                 MOV     T.ADDR,R0                       ;                               5142
000250  016016  000030                  MOV     30(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  0000006                 MOV     #RPT8,-(SP)
000300  012746  000006                  MOV     #6,-(SP)
000304  010600                          MOV     SP,R0                           ; SP,*
000306  104416                          TRAP    16
000310  013700  0000006                 MOV     T.ADDR,R0                       ;                               5145
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  0000006                 MOV     #RPT8,-(SP)
000344  012746  000006                  MOV     #6,-(SP)
000350  010600                          MOV     SP,R0                           ; SP,*
000352  104416                          TRAP    16
000354  013700  0000006                 MOV     T.ADDR,R0                       ;                               5148
000360  005016                          CLR     (SP)
000362  116016  000055                  MOVB    55(R0),(SP)
000366  005046                          CLR     -(SP)
000370  116016  000054                  MOVB    54(R0),(SP)
000374  005046                          CLR     -(SP)
000376  116016  000053                  MOVB    53(R0),(SP)
000402  005046                          CLR     -(SP)
```

```
000404  116016  000052          MOVB    52(R0),(SP)
000410  005046                  CLR     -(SP)
000412  116016  000051          MOVB    51(R0),(SP)
000416  005046                  CLR     -(SP)
000420  116016  000050          MOVB    50(R0),(SP)
000424  005046                  CLR     -(SP)
000426  116016  000047          MOVB    47(R0),(SP)
000432  005046                  CLR     -(SP)
000434  116016  000046          MOVB    46(R0),(SP)
000440  012746  0000006         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                  ;                                       5102
000460  062702  000012      3$: ADD     #12,R2                  ; *,DISK                                5118
000464  020227  000041          CMP     R2,#41                  ; DISK,*
000470  003624                  BLE     2$
000472  010316                  MOV     R3,(SP)                 ; CTLR,*                                5143
000474  012746  000126          MOV     #126,-(SP)
000500  004737  0000000G        JSR     PC,BL$MUL
000504  005726                  TST     (SP)+
000506  005760  000002G         TST     CST 2(R0)
000512  100026                  BPL     4$
000514  012716  0000006         MOV     #RPT10,(SP)             ;                                       5147
000520  012746  000001          MOV     #1, (SP)
000524  010600                  MOV     SP,R0                   ; SP,*
000526  104416                  TRAP    16
000530  010300                  MOV     R3,R0                   ; CTLR,*                                5148
000532  006300                  ASL     R0
000534  005016                  CLR     (SP)
000536  116016  000001G         MOVB    C.ERR.TBL+1(R0),(SP)
000542  005046                  CLR     -(SP)
000544  116016  0000006         MOVB    C.ERR.TBL(R0),(SP)
000550  012746  0000006         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                  ;                                       5149
000570  012716  0000006     4$: MOV     #CRLF,(SP)              ;                                       5151
000574  012746  000001          MOV     #1, (SP)
000600  010600                  MOV     SP,R0                   ; SP,*
000602  104416                  TRAP    16
000604  005726                  TST     (SP)+                   ;                                       5112
000606  005203                  INC     R3                      ; CTLR                                  5113
000610  000243                  .WORD CLV!CLC

000612  003002                  BGT     5$
000614  000137  000652'         JMP     1$
000620  010300              5$: MOV     R3,R0                   ; CUR.PRIORITY,*                        5[illegible]
000622  104441                  TRAP    41
000624  062706  000016          ADD     #16,SP                  ;                                       5086
000630  000207                  RTS     PC
```

```
;  Routine Size:  205 words,    Routine Base:  $CODE$ + 0522
;  Maximum stack depth per invocation:  40 words


                        .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000522'  L$RPT:: JSR     PC,LRPT                 ;                5155
000004  104425                  TRAP    #25
000006  000207                  RTS     PC

;  Routine Size:  4 words,    Routine Base:  $CODE$ + 1354
;  Maximum stack depth per invocation:  2 words

;   5158  1
```

```
;    5159  1     %sbttl 'INITIALIZE SECTION'
;    5160  1
;    5161  2     BGNINIT;
;    5162  2
;    5163  2     local
;    5164  2         DELAY_MULT : word,
;    5165  2         FLAG : byte,
;    5166  2         TEMP : word,
;    5167  2         HWPT_REF : ref block [HWPT_LEN, word] field (HWP_FIELDS),
;    5168  2         CLEAR_TABLES : byte,
;    5169  2         SMALLEST_DRIVE : byte,
;    5170  2         HWPT_ADDRESS : vector [MAX_UNITS, word];
;    5171  2
;    5172  2     SETPRI (PRI07);                                  ! NO INTERRUPTS ALLOWED DURING INIT
;    5173  2
;    5174  2     if READEF (EF_NEW)                               ! IS THIS A NEW PASS?
;    5175  2     then
;    5176  3         begin
;    5177  3         ENTRY_REASON = NEW_PASS;
;    5178  3
;    5179  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5180  3         then
;    5181  3             CLEAR_TABLES = FALSE
;    5182  3         else
;    5183  3             CLEAR_TABLES = TRUE;
;    5184  3
;    5185  2         end;
;    5186  2
;    5187  2     if READEF (EF_START)                             ! IS THIS A START?
;    5188  2     then
;    5189  3         begin
;    5190  3         BRESET;
;    5191  3         ENTRY_REASON = START;
;    5192  3         CLEAR_TABLES = TRUE;
;    5193  3         ADDR_VECT_OK = FALSE;
;    5194  3         INIT_OCCURED = FALSE;
;    5195  2         end;
;    5196  2
;    5197  2     if READEF (EF_RESTART)                           ! IS THIS A RESTART?
;    5198  2     then
;    5199  3         begin
;    5200  3         ENTRY_REASON = RESTART;
;    5201  3         CLEAR_TABLES = TRUE;
;    5202  2         end;
;    5203  2
;    5204  2     if READEF (EF_CONTINUE)                          ! IS THIS A CONTINUE?
;    5205  2     then
;    5206  3         begin
;    5207  3         ENTRY_REASON = CONT;
;    5208  3
;    5209  4         if not BIT_TST (SWP_FLAGS, SWF_CST)
;    5210  3         then
;    5211  3             CLEAR_TABLES = FALSE
```

```
;    5212  3          else
;    5213  3              CLEAR_TABLES = TRUE;
;    5214  3
;    5215  2          end;
;    5216  2
;    5217  2      if READEF (EF_PWR)                                          ! ARE WE HERE BECAUSE OF POWER FAIL
;    5218  2      then
;    5219  3          begin
;    5220  3          ENTRY_REASON = PWR_FAIL;
;    5221  3          ADDR_VECT_OK = FALSE;
;    5222  3          INIT_OCCURED = FALSE;
;    5223  3          CLEAR_TABLES = TRUE;
;    5224  3          PRINTF (MSG_01);                                        ! "POWER DELAY - WAITING"
;    5225  3
;    5226  3          incr COUNT from 0 to 60 do                              ! WAIT APPROX. 60 SECONDS
;    5227  4              begin
;    5228  4              DELAY_MULT = 333;
;    5229  4              DELAY (.DELAY_MULT);
;    5230  4              BREAK;                                              ! BREAK FOR ACT
;    5231  3              end;
;    5232  3
;    5233  2          end;
;    5234  2
;    5235  2      !SETVEC (0_TVEC, 0_BRK, PRI07);                             ! SET UDT TRAP VECTOR
;    5236  2
;    5237  2      !+
;    5238  2      !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
;    5239  2      !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
;    5240  2      !       (DIAGNOSTIC IS ABORTED).
;    5241  2      !-
;    5242  2
;    5243  2      if .L$UNIT gtru MAX_UNITS
;    5244  2      then
;    5245  3          begin
;    5246  3          ERRSF (1, EGS_01, EMS_01);
;    5247  3          DOCLN;
;    5248  2          end;
;    5249  2
;    5250  2      !+
;    5251  2      !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
;    5252  2      !       ALL RUN-TIME CONTROLLER STATUS TABLES (CSTs) ARE CLEARED TO 0, THEN
;    5253  2      !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
;    5254  2      !-
;    5255  2
;    5256  2      if .ENTRY_REASON neq NEW_PASS
;    5257  2      then
;    5258  3          begin
;    5259  3          SMALLEST_DRIVE = 255;                                   ! LARGEST DISK NO. ALLOWED BY MSCP
;    5260  3
;    5261  3          incr COUNT from 0 to ((MAX_CTLR * CST_LEN * 2) - 2) by 2 do
;    5262  3              (CST + .COUNT) = 0;
;    5263  3
;    5264  3          incr UNIT from 0 to (.L$UNIT - 1) do                    ! LOOP THROUGH ALL UNITS
```

```
;    5265  3
;    5266  3          if (HWPT_ADDRESS [.UNIT] = GPHARD (.UNIT, HWPT_REF)) neqa 0                ! IF HWP TABLE FOUND
;    5267  3          then
;    5268  3
;    5269  3              if .HWPT_REF [HWP_DISK_NUM] lssu .SMALLEST_DRIVE                       ! FIND OUT THE SMALLEST DISK NUMBER
;    5270  3              then
;    5271  3                  SMALLEST_DRIVE = .HWPT_REF [HWP_DISK_NUM];
;    5272  3
;    5273  3      incr UNIT from 0 to (.L$UNIT - 1) do                                           ! LOOP THROUGH ALL UNITS
;    5274  3
;    5275  3          if .HWPT_ADDRESS [.UNIT] neqa 0                                            ! IF HWP TABLE FOUND
;    5276  3          then
;    5277  4              begin
;    5278  4              FLAG = NOT_FOUND;
;    5279  4              HWPT_REF = .HWPT_ADDRESS [.UNIT];
;    5280  4
;    5281  4              incr CTLR from 0 to (MAX_CTLR - 1) do                                  ! LOOP THROUGH ALL CSTs
;    5282  4
;    5283  4                  if .CST [.CTLR, IP_ADDR] eqla .HWPT_REF [HWP_IP_ADDR]
;    5284  4                  then
;    5285  4
;    5286  4                      if .CST [.CTLR, (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE
;    5287  4                              + OF_UN + OF_DATA, D_PRES] eql NOT_PRESENT
;    5288  4                      then
;    5289  5                          begin                                                      ! IF EMPTY SLOT FOUND
;    5290  5                          TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;    5291  5                          CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    5292  5                                                                                     ! COPY DISK ADDR AND PROT BIT
;    5293  5                          CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    5294  5                          CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    5295  5                          CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    5296  5
;    5297  5                          CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =
;    5298  5                                  .HWPT_REF [HWP_BEG_TRK];                           !ZZZ
;    5299  5                          CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =                     !ZZZ
;    5300  5                                  .HWPT_REF [HWP_BEG_TRK1];                          !ZZZ
;    5301  5                          CST [.CTLR, .TEMP + OF_END, D_END0] =                      !ZZZ
;    5302  5                                  .HWPT_REF [HWP_END_TRK];                           !ZZZ
;    5303  5                          CST [.CTLR, .TEMP + OF_END1, D_END1] =                     !ZZZ
;    5304  5                                  .HWPT_REF [HWP_END_TRK1];                          !ZZZ
;    5305  5
;    5306  5                          CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;               !ZZZ
;    5307  5                          CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =             !ZZZ
;    5308  5                                  NOT (.HWPT_REF [HWP_DISK_DUPEX]);                  !ZZZ
;    5309  5
;    5310  5                          CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =               !ZZZ
;    5311  5                                  (.HWPT_REF [HWP_DISK_DUPWT]);                      !ZZZ
;    5312  5                          CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;                !ZZZ
;    5313  5                          FLAG = FOUND;
;    5314  5                          exitloop;
;    5315  5                          end
;    5316  4                  else
;    5317  5                      begin                                                          ! DUPLICATE UNIT
```

```
;    5318  5                          PRINTF (CER_01, .HWPT_REF [HWP_DISK_NUM], .HWPT_REF [HWP_IP_ADDR));
;    5319  5                                                                          ! "DUPLICATE UNIT: X AT IP: XXXXXX"
;    5320  5                          DUR [.UNIT] = DU_CONF;                          ! CONFIGURATION ERROR
;    5321  5                          DODU (.UNIT);                                   ! DROP UNIT
;    5322  5                          FLAG = FOUND;
;    5323  5                          exitloop;
;    5324  4                          end;
;    5325  4
;    5326  4                  if .FLAG eql NOT_FOUND                                  ! IF NO IP MATCH TO EXISTING CST
;    5327  4                  then
;    5328  5                      begin
;    5329  5
;    5330  5                      incr CTLR from 0 to (MAX_CTLR - 1) do               ! LOOP THROUGH EACH CST
;    5331  5
;    5332  5                          if .CST [.CTLR, IP_ADDR] eqla 0                 ! IF EMPTY CST FOUND
;    5333  5                          then
;    5334  6                              begin
;    5335  6                              CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;    5336  6                              CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;    5337  6                              CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;    5338  6                              TEMP = (.HWPT_REF [HWP_DISK_NUM] - .SMALLEST_DRIVE) * UNIT_SIZE + OF_UN;
;    5339  6                              CST [.CTLR, .TEMP + OF_DATA, D_ALL] = .HWPT_REF [HWP_DISK];
;    5340  6                                                                          ! COPY DISK ADDR AND PROT BIT
;    5341  6                              CST [.CTLR, .TEMP + OF_DATA, D_UNIT] = .UNIT;
;    5342  6                              CST [.CTLR, .TEMP + OF_DATA, D_FATAL] = FALSE;
;    5343  6                              CST [.CTLR, .TEMP + OF_DATA, D_PRES] = PRESENT;
;    5344  6
;    5345  6                              IF .HWPT_REF [HWP_ENTIRE] EQL TRUE          !ZZZ IF DEFAULT TEST RANGE,
;    5346  6                              THEN HWPT_REF [HWP_END_TRK] = ALL_ONES;     !ZZZ MAKE HI ADDR ALL ONES
;    5347  6
;    5348  6
;    5349  6                              CST [.CTLR, .TEMP + OF_BEG, D_BEG0] =       !ZZZ
;    5350  6                                      .HWPT_REF [HWP_BEG_TRK];            !ZZZ
;    5351  6                              CST [.CTLR, .TEMP + OF_BEG1, D_BEG1] =      !ZZZ
;    5352  6                                      .HWPT_REF [HWP_BEG_TRK1];           !ZZZ
;    5353  6                              CST [.CTLR, .TEMP + OF_END, D_END0] =       !ZZZ
;    5354  6                                      .HWPT_REF [HWP_END_TRK];            !ZZZ
;    5355  6                              CST [.CTLR, .TEMP + OF_END1, D_END1] =      !ZZZ
;    5356  6                                      .HWPT_REF [HWP_END_TRK1];           !ZZZ
;    5357  6
;    5358  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, D_DBN] = 0;    !ZZZ
;    5359  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, NODUPMEDIA] =  !ZZZ
;    5360  6                                  NOT (.HWPT_REF [HWP_DISK_DUPEX]);               !ZZZ
;    5361  6                              CST [.CTLR, .TEMP + OF_DUPFLAGS, DUPWRITE] =    !ZZZ
;    5362  6                                  (.HWPT_REF [HWP_DISK_DUPWT]);           !ZZZ
;    5363  6                              CST [.CTLR, .TEMP + OF_COUNT, D_COUNT] = 0;     !ZZZ
;    5364  6                              FLAG = FOUND;
;    5365  6                              exitloop;
;    5366  5                              end;                                        ! IF EMPTY CST FOUND
;    5367  5
;    5368  5                  if .FLAG eql NOT_FOUND                                  ! IF NO EMPTY CST FOUND
;    5369  5                  then
;    5370  6                      begin
```

```
;    5371  6                    PRINTF (CER_02, MAX_CTLR);                          ! "MORE THAN X IP ADDRESSES."
;    5372  6                    DUR [.UNIT] = DU_CONF;                              ! CONFIGURATION ERROR
;    5373  6                    DODU (.UNIT);                                       ! DROP UNIT
;    5374  5                    end;
;    5375  5
;    5376  4                end;                                                    ! IF NO IP ADDR MATCH IN CST
;    5377  4
;    5378  3            end;                                                        ! IF GPHARD RETURNS A HWP TABLE
;    5379  3        !
;    5380  3        ! CONFIGURATON CHECK FOR LEGAL RDRX UNIT MIX BECAUSE WE HAVE DIFFERENT
;    5381  3        ! DRIVES :  THE RD51, RD52, AND RX50.
;    5382  3        ! (NEEDED?)
;    5383  3        !
;    5384  2        end;                                                            ! END OF "NON NEW_PASS" INIT
;    5385  2
;    5386  2    if .ENTRY_REASON eql NEW_PASS
;    5387  2    then
;    5388  3        begin
;    5389  3
;    5390  3        incr UNIT from 0 to (.L$UNIT - 1) do
;    5391  3            GPHARD (.UNIT, HWPT_REF);                                   ! DUMMY GPHARDs FOR NEW PASS
;    5392  3
;    5393  3        incr CTLR from 0 to (MAX_CTLR - 1) do
;    5394  4            begin
;    5395  4            CST [.CTLR, U_CNT] = 0;                                     ! REINITIALIZE UNIT COUNT
;    5396  4
;    5397  4            incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    5398  4                CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;       ! START EACH UNIT AS OFFLINE
;    5399  4
;    5400  3            end;
;    5401  3
;    5402  2        end;
;    5403  2
;    5404  2    if .ENTRY_REASON eql START
;    5405  2    then
;    5406  3        begin
;    5407  3        CTLR_CNT = 0;                                                   ! NUMBER OF CONFIGURED CONTROLLERS
;    5408  3
;    5409  3        incr CTLR from 0 to (MAX_CTLR - 1) do
;    5410  3
;    5411  3            if .CST [.CTLR, IP_ADDR] neqa 0                             ! IF CONTROLLER IS PRESENT
;    5412  3            then
;    5413  3                CTLR_CNT = .CTLR_CNT + 1;                               ! INCREMENT CONTROLLER COUNT
;    5414  3
;    5415  3        MEMORY (FREE_MEM_ADDR);                                         ! GET START OF FREE MEMORY
;    5416  3
;    5417  2        end;                                                            ! END OF "START" INITIALIZATION
;    5418  2
;    5419  2    !+
;    5420  2    !       CLEAR STATISTICS TABLES
;    5421  2    !
;    5422  2
;    5423  2    incr UNITS from 0 to MAX_UNITS - 1 do                               ! CLEAR CURRENT STATISTICS
```

```
;    5424  2          incr COUNT from 0 to TALLY_CLEAR - 1 do
;    5425  2              TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;
;    5426  2
;    5427  2      if .CLEAR_TABLES                                        ! IF CLEAR TABLES ON EVERY PASS
;    5428  2      then
;    5429  2          incr UNITS from 0 to MAX_UNITS - 1 do
;    5430  2              incr COUNT from TALLY_CLEAR to TALLY_LEN - 1 do  ! INITIALIZE TOTALS
;    5431  2                  TALLY [.UNITS * TALLY_LEN + .COUNT] = 0;     !
;    5432  2
;    5433  2      if .CLEAR_TABLES
;    5434  2      then
;    5435  2          incr CTLR from 0 to MAX_CTLR - 1 do
;    5436  3              begin
;    5437  3              C_ERR_TBL [.CTLR, C_ERR_HRD] = 0;               ! INITALIZE CONTROLLER ERRORS
;    5438  3              C_ERR_TBL [.CTLR, C_ERR_SFT] = 0;               !
;    5439  2              end;
;    5440  2
;    5441  2      !+
;    5442  2      !       MISCELLANEOUS INITIALIZATON
;    5443  2      !-
;    5444  2
;    5445  2      incr CTLR from 0 to (MAX_CTLR - 1) do                   ! INIT NO. OF OUTSTANDING QIOs
;    5446  2          QIO [.CTLR] = 0;
;    5447  2
;    5448  2      incr COUNT from 0 to (RP_CNT - 1) do                    ! INITIALIZE RETURN PACKET POOL
;    5449  2          RP_USE [.COUNT] = -1;
;    5450  2
;    5451  2      if .CLK_PRESENT                                         ! STOP CLOCK IF PRESENT
;    5452  2      then
;    5453  2          LINE_CLOCK = 0;
;    5454  2
;    5455  2      IODQ_IN = IODQ_OUT = 0;                                 ! INIT I/O DONE QUEUE POINTERS
;    5456  2      CRN_LOW = CRN_HIGH = 0;                                 ! INIT COMMAND REFERENCE NUMBER
;    5457  2      SETPRI (PRI00);                                         ! SET PROGRAM PRIORITY TO 0
;    5458  2
;    5459  1      ENDINIT;

                                .GLOBL  L$DLY

                                .SBTTL  LINIT INITIALIZE SECTION
000000  004137  000000G    LINIT:  JSR     R1,$SAVE5                   ;                                5157
000004  162706  000030            SUB     #30,SP
000010  012700  000340            MOV     #340,R0                     ;                                5172
000014  104441                    TRAP    41
000016  012700  000035            MOV     #35,R0                      ;                                5174
000022  104447                    TRAP    47
000024  103014                    BHIS    2$
000026  112737  000005  000000G   MOVB    #5,ENTRY.REASON             ;                                5177
000034  105737  000000G           TSTB    SWP.FLAGS                   ;                                5179
000040  100403                    BMI     1$
000042  105066  000010            CLRB    10(SP)                      ; CLEAR.TABLES                   5181
```

```
000046  000403                          1$:     BR      2$                              ;
000050  112766  000001  000010                  MOVB    #1,10(SP)                       ; *,CLEAR.TABLES                5179
000056  012700  000040                  2$:     MOV     #40,R0                          ;                               5183
000062  104447                                  TRAP    47                                                              5187
000064  103013                                  BHIS    3$
000066  104433                                  TRAP    33                              ;                               5189
000070  112737  000001  000000G                MOVB    #1,ENTRY.REASON                 ;                               5191
000076  112766  000001  000010                  MOVB    #1,10(SP)                       ; *,CLEAR.TABLES                5192
000104  105037  000000G                         CLRB    ADDR.VECT.OK                    ;                               5193
000110  105037  000000G                         CLRB    INIT.OCCURED                    ;                               5194
000114  012700  000037                  3$:     MOV     #37,R0                          ;                               5197
000120  104447                                  TRAP    47
000122  103006                                  BHIS    4$
000124  112737  000002  000000G                MOVB    #2,ENTRY.REASON                 ;                               5200
000132  112766  000001  000010                  MOVB    #1,10(SP)                       ; *,CLEAR.TABLES                5201
000140  012700  000036                  4$:     MOV     #36,R0                          ;                               5204
000144  104447                                  TRAP    47
000146  103014                                  BHIS    6$
000150  112737  000003  000000G                MOVB    #3,ENTRY.REASON                 ;                               5207
000156  105737  000000G                         TSTB    SWP.FLAGS                       ;                               5209
000162  100403                                  BMI     5$
000164  105066  000010                          CLRB    10(SP)                          ; CLEAR.TABLES                  5211
000170  000403                                  BR      6$                              ;                               5209
000172  112766  000001  000010          5$:     MOVB    #1,10(SP)                       ; *,CLEAR.TABLES                5213
000200  012700  000034                  6$:     MOV     #34,R0                          ;                               5217
000204  104447                                  TRAP    47
000206  103043                                  BHIS    12$
000210  112737  000004  000000G                MOVB    #4,ENTRY.REASON                 ;                               5220
000216  105037  000000G                         CLRB    ADDR.VECT.OK                    ;                               5221
000222  105037  000000G                         CLRB    INIT.OCCURED                    ;                               5222
000226  112766  000001  000010                  MOVB    #1,10(SP)                       ; *,CLEAR.TABLES                5223
000234  012746  000000G                         MOV     #MSG.01,-(SP)                   ;                               5224
000240  012746  000001                          MOV     #1,-(SP)
000244  010600                                  MOV     SP,R0                           ; SP,*
000246  104417                                  TRAP    17
000250  012702  000075                          MOV     #75,R2                          ; *,COUNT                       5226
000254  012703  000515                  7$:     MOV     #515,R3                         ; *,DELAY.MULT                  5228
000260  010301                                  MOV     R3,R1                           ; DELAY.MULT,$$TMP2             5229
000262  001411                          8$:     BEQ     11$
000264  013700  000000G                         MOV     L$DLY,R0                        ; *,$$TMP1
000270  001404                                  BEQ     10$
000272  005066  000022                  9$:     CLR     22(SP)                          ; $$TMP
000276  005300                                  DEC     R0                              ; $$TMP1
000300  001374                                  BNE     9$
000302  005301                          10$:    DEC     R1                              ; $$TMP2
000304  000766                                  BR      8$
000306  104422                          11$:    TRAP    22
000310  005302                                  DEC     R2                              ; COUNT                         5236
000312  001360                                  BNE     7$
000314  022626                                  CMP     (SP)+,(SP)+                     ;                               5239
000316  023727  000000G 000004          12$:    CMP     L$UNIT,#4                       ;                               5243
000324  101405                                  BLOS    13$
000326  104454                                  TRAP    54                              ;                               5246
```

```
000330  000001                                  .WORD   1
000332  000000G                                 .WORD   EGS.01
000334  000000V                                 .WORD   EMS.01
000336  104444                                  TRAP    44
000340  123727  000000G 000005          13$:    CMPB    ENTRY.REASON,#5         ;                               5256
000346  001002                                  BNE     14$
000350  000137  003362'                          JMP     41$
000354  112766  000377  000006          14$:    MOVB    #377,6(SP)              ; *,SMALLEST.DRIVE              5259
000362  005000                                  CLR     R0                      ; COUNT                         5261
000364  005060  000000G               15$:    CLR     CST(R0)                 ; *(COUNT)                      5262
000370  062700  000002                          ADD     #2,R0                   ; *,COUNT                       5261
000374  020027  000124                          CMP     R0,#124                 ; COUNT,*
000400  003771                                  BLE     15$
000402  013704  000000G                         MOV     L$UNIT,R4               ;                               5264
000406  005003                                  CLR     R3                      ; UNIT
000410  000435                                  BR      18$
000412  010302                          16$:    MOV     R3,R2                   ; UNIT,*                        5266
000414  006302                                  ASL     R2
000416  012700  000020                          MOV     #20,R0
000422  060600                                  ADD     SP,R0                   ; HWPT.ADDRESS,*
000424  060002                                  ADD     R0,R2
000426  010300                                  MOV     R3,R0                   ; UNIT,*
000430  104442                                  TRAP    42
000432  010001                                  MOV     R0,R1                   ; *,HWPT.REF
000434  010112                                  MOV     R1,(R2)                 ; HWPT.REF,*
000436  001421                                  BEQ     17$
000440  005002                                  CLR     R2                      ;                               5269
000442  156602  000006                          BISB    6(SP),R2                ; SMALLEST.DRIVE,*
000446  116100  000006                          MOVB    6(R1),R0                ; *(HWPT.REF),*
000452  042700  177760                          BIC     #177760,R0
000456  020002                                  CMP     R0,R2
000460  103010                                  BHIS    17$
000462  116100  000006                          MOVB    6(R1),R0                ; *(HWPT.REF),*                 5271
000466  042700  177760                          BIC     #177760,R0
000472  105066  000006                          CLRB    6(SP)                   ; SMALLEST.DRIVE
000476  050066  000006                          BIS     R0,6(SP)                ; *,SMALLEST.DRIVE
000502  005203                          17$:    INC     R3                      ; UNIT                          5264
000504  020304                          18$:    CMP     R3,R4                   ; UNIT,*
000506  002741                                  BLT     16$
000510  013766  000000G 000014                  MOV     L$UNIT,14(SP)           ;                               5273
000516  005004                                  CLR     R4                      ; UNIT
000520  000137  003340'                          JMP     39$
000524  010400                          19$:    MOV     R4,R0                   ; UNIT,*                        5275
000526  006300                                  ASL     R0
000530  012703  000020                          MOV     #20,R3
000534  060603                                  ADD     SP,R3                   ; HWPT.ADDRESS,*
000536  060300                                  ADD     R3,R0
000540  005710                                  TST     (R0)
000542  001002                                  BNE     20$
000544  000137  003336'                          JMP     38$
000550  105066  000004          20$:    CLRB    4(SP)                   ; FLAG                          5278
000554  011001                                  MOV     (R0),R1                 ; *,HWPT.REF                    5279
000556  005016                                  CLR     (SP)                    ; CTLR                          5281
```

```
000560  011646                          21$:    MOV     (SP),-(SP)              ; CTLR,*                        5283
000562  012746  000126                          MOV     #126,-(SP)
000566  004737  000000G                        JSR     PC,BL$MUL
000572  022626                                 CMP     (SP)+,(SP)+
000574  026011  000000G                        CMP     CST(R0),(R1)            ; *,HWPT.REF
000600  001402                                 BEQ     22$
000602  000137  002612'                        JMP     27$
000606  012766  000001  000012      22$:      MOV     #1,12(SP)               ;                               5313
000614  112766  000001  000004              MOVB    #1,4(SP)                ; *,FLAG
000622  012705  000006                        MOV     #6,R5                   ;                               5286
000626  060105                               ADD     R1,R5                   ; HWPT.REF,*
000630  111546                               MOVB    (R5),-(SP)
000632  042716  177760                       BIC     #177760,(SP)
000636  005000                               CLR     R0
000640  156600  000010                       BISB    10(SP),R0               ; SMALLEST.DRIVE,*
000644  160016                               SUB     R0,(SP)
000646  012746  000012                       MOV     #12,-(SP)
000652  004737  000000G                      JSR     PC,BL$MUL
000656  010066  000006                       MOV     R0,6(SP)
000662  022626                               CMP     (SP)+,(SP)+
000664  011646                               MOV     (SP),-(SP)              ; CTLR,*                        5287
000666  012746  000053                       MOV     #53,-(SP)
000672  004737  000000G                      JSR     PC,BL$MUL
000676  010003                               MOV     R0,R3
000700  022626                               CMP     (SP)+,(SP)+
000702  066600  000002                       ADD     2(SP),R0
000706  006300                               ASL     R0
000710  032760  040000  000006G             BIT     #40000,CST+6(R0)
000716  001121                               BNE     26$
000720  016602  000002                       MOV     2(SP),R2                ; *,TEMP                        5290
000724  062702  000003                       ADD     #3,R2                   ; *,TEMP
000730  010300                               MOV     R3,R0                   ;                               5291
000732  060200                               ADD     R2,R0                   ; TEMP,*
000734  006300                               ASL     R0
000736  062700  000000G                      ADD     #CST,R0
000742  011510                               MOV     (R5),(R0)
000744  010446                               MOV     R4,-(SP)                ; UNIT,*                        5293
000746  000316                               SWAP    (SP)
000750  042716  170377                       BIC     #170377,(SP)
000754  042710  007400                       BIC     #7400,(R0)
000760  052610                               BIS     (SP)+,(R0)
000762  042710  010000                       BIC     #10000,(R0)             ;                               5294
000766  052710  040000                       BIS     #40000,(R0)             ;                               5295
000772  010300                               MOV     R3,R0                   ;                               5297
000774  060200                               ADD     R2,R0                   ; TEMP,*
000776  006300                               ASL     R0
001000  016160  000010  000002G             MOV     10(R1),CST+2(R0)        ; *(HWPT.REF),*
001006  010300                               MOV     R3,R0                   ;                               5299
001010  060200                               ADD     R2,R0                   ; TEMP,*
001012  006300                               ASL     R0
001014  016160  000012  000004G             MOV     12(R1),CST+4(R0)        ; *(HWPT.REF),*
001022  010300                               MOV     R3,R0                   ;                               5301
001024  060200                               ADD     R2,R0                   ; TEMP,*
```

```
001026  006300                          ASL     R0
001030  016160  000014  000006G          MOV     14(R1),CST+6(R0)        ; *(HWPT.REF),*
001036  010300                          MOV     R3,R0                   ;                       5303
001040  060200                          ADD     R2,R0                   ; TEMP,*
001042  006300                          ASL     R0
001044  016160  000016  000010G          MOV     16(R1),CST+10(R0)       ; *(HWPT.REF),*
001052  010300                          MOV     R3,R0                   ;                       5306
001054  060200                          ADD     R2,R0                   ; TEMP,*
001056  006300                          ASL     R0
001060  062700  000020G                  ADD     #CST+20,R0
001064  105010                          CLRB    (R0)
001066  111546                          MOVB    (R5),-(SP)              ;                       5308
001070  005046                          CLR     -(SP)
001072  032766  000040  000002          BIT     #40,2(SP)
001100  001401                          BEQ     23$
001102  005216                          INC     (SP)
001104  005116                  23$:    COM     (SP)
001106  011646                          MOV     (SP),-(SP)
001110  042710  100000                  BIC     #100000,(R0)
001114  006026                          ROR     (SP)+
001116  103002                          BCC     24$
001120  052710  100000                  BIS     #100000,(R0)
001124  005726                  24$:    TST     (SP)+
001126  111516                          MOVB    (R5),(SP)               ;                       5310
001130  042710  010000                  BIC     #10000,(R0)
001134  032726  000100                  BIT     #100,(SP)+
001140  001402                          BEQ     25$
001142  052710  010000                  BIS     #10000,(R0)
001146  010300                  25$:    MOV     R3,R0                   ;                       5312
001150  060200                          ADD     R2,R0                   ; TEMP,*
001152  006300                          ASL     R0
001154  005060  0000226                 CLR     CST+22(R0)
001160  000427                          BR      28$                     ;                       5289
001162  011146                  26$:    MOV     (R1),-(SP)              ; HWPT.REF,*            5318
001164  111546                          MOVB    (R5),-(SP)
001166  042716  177760                  BIC     #177760,(SP)
001172  012746  000000G                  MOV     #CER.01,-(SP)
001176  012746  000003                  MOV     #3,-(SP)
001202  010600                          MOV     SP,R0                   ; SP,*
001204  104417                          TRAP    17
001206  062706  000010                  ADD     #10,SP
001212  112764  000001  000000G          MOVB    #1,DUR(R4)              ; *,*(UNIT)             5320
001220  010400                          MOV     R4,R0                   ; UNIT,*                5321
001222  104451                          TRAP    51
001224  000405                          BR      28$                     ;                       5317
001226  005216                  27$:    INC     (SP)                    ; CTLR                  5281
001230  000243                          .WORD CLV!CLC

001232  003002                          BGT     28$
001234  000137  002144'                  JMP     21$
001240  105766  000004          28$:    TSTB    4(SP)                   ; FLAG                  5326
001244  001402                          BEQ     29$
001246  000137  003336'                  JMP     38$
```

```
001500 005066 000012            CLR     12(SP)                  ; CTLR                          5330
001504 016646 000012            MOV     12(SP), -(SP)           ; CTLR,*                        5332
001510 012746 000126            MOV     #126, -(SP)
001514 004737 0000000G          JSR     PC,BLI$MUL
001520 022626                   CMP     (SP)+,(SP)+
001522 005760 0000000G          TST     CST(R0)
001526 001174                   BNE     35$
001530 011160 0000000G          MOV     (R1),CST(R0)            ; HWPT.REF,*                    5335
001534 016103 000002            MOV     2(R1),R3                ; *(HWPT.REF),*                 5336
001540 042703 177000            BIC     #177000,R3
001544 042760 000777 0000006    BIC     #777,CST+2(R0)
001552 050360 0000026           BIS     R3,CST+2(R0)
001556 116160 000004 0000046    MOVB    4(R1),CST+4(R0)         ; *(HWPT.REF),*                 5337
001564 012705 000006            MOV     #6,R5                   ;                               5338
001570 060105                   ADD     R1,R5                   ; HWPT.REF,*
001572 111546                   MOVB    (R5),-(SP)
001574 042716 177760            BIC     #177760,(SP)
001600 005000                   CLR     R0
001602 156600 000010            BISB    10(SP),R0               ; SMALLEST.DRIVE,*
001606 160016                   SUB     R0,(SP)
001610 012746 000012            MOV     #12.,-(SP)
001614 004737 0000000G          JSR     PC,BLI$MUL
001620 005726                   TST     (SP)+
001622 010002                   MOV     R0,R2                   ; *,TEMP
001624 062702 000005            ADD     #5,R2                   ; *,TEMP
001630 016616 000014            MOV     14(SP),(SP)             ; CTLR,*                        5340
001634 012746 000005            MOV     #5,-(SP)
001640 004737 0000000G          JSR     PC,BLI$MUL
001644 010003                   MOV     R0,R3
001646 005726                   TST     (SP)+
001650 060200                   ADD     R2,R0                   ; TEMP,*
001652 006300                   ASL     R0
001654 062700 0000000G          ADD     #CST,R0
001660 011510                   MOV     (R5),(R0)
001662 010416                   MOV     R4,-(SP)                ; UNIT,*                        5341
001664 000516                   SWAB    (SP)
001666 042716 170377            BIC     #170377,(SP)
001672 042710 007400            BIC     #7400,(R0)
001676 052610                   BIS     (SP)+,(R0)
001700 042710 010000            BIC     #10000,(R0)             ;                               5342
001704 052710 040000            BIS     #40000,(R0)             ;                               5343
001710 105715                   TSTB    (R5)                    ;                               5344
001712 100005                   BPL     31$
001714 012761 177777 000014     MOV     #-1,14(R1)              ; *,*HWPT.REF                   5345
001722 010300                   MOV     R3,R0                   ;                               5346
001724 060200                   ADD     R2,R0                   ; TEMP,*
001726 006300                   ASL     R0
001730 016160 000010 0000026    MOV     10(R1),CST+2(R0)        ; *(HWPT.REF),*
001736 010300                   MOV     R3,R0                   ;                               5351
001740 060200                   ADD     R2,R0                   ; TEMP,*
001742 006300                   ASL     R0
001744 016160 000012 0000046    MOV     12(R1),CST+4(R0)        ; *(HWPT,REF),*
001752 010300                   MOV     R3,R0                   ;                               5353
```

```
001526  060200                          ADD     R2,R0                   ; TEMP,*
001530  006500                          ASL     R0
001532  016160  000014  000006          MOV     14(R1),CST+6(R0)        ; *(HWPT.REF),*
001540  010300                          MOV     R3,R0                   ;                       ?35?
001542  060200                          ADD     R2,R0                   ; TEMP,*
001544  006300                          ASL     R0
001546  016160  000016  000010          MOV     16(R1),CST+10(R0)       ; *(HWPT.REF),*
001554  010300                          MOV     R3,R0                   ;                       ?35?
001556  060200                          ADD     R2,R0                   ; TEMP,*
001560  006300                          ASL     R0
001562  062700  000020                  ADD     #CST+20,R0
001566  105010                          CLRB    (R0)
001570  111546                          MOVB    (R5), (SP)              ;                       ?46?
001572  005046                          CLR     (SP)
001574  032766  000040  000002          BIT     #40,2(SP)
001602  001401                          BEQ     52$
001604  005216                          INC     (SP)
001606  005116                  52$:    COM     (SP)
001610  011646                          MOV     (SP), -(SP)
001612  042710  100000                  BIC     #100000,(R0)
001616  006026                          ROR     (SP)+
001620  103002                          BCC     33$
001622  052710  100000                  BIS     #100000,(R0)
001626  005726                  33$:    TST     (SP)+
001630  111516                          MOVB    (R5),(SP)               ;                       ?46?
001632  042710  010000                  BIC     #10000,(R0)
001636  032726  000100                  BIT     #100,(SP)+
001642  001402                          BEQ     34$
001644  052710  010000                  BIS     #10000,(R0)
001650  010300                  34$:    MOV     R3,R0                   ;                       ?46?
001652  060200                          ADD     R2,R0                   ; TEMP,*
001654  006500                          ASL     R0
001656  005060  000020                  CLR     CST+20(R0)
001662  112766  000001  000004          MOVB    #1,4(SP)                ; *,FLAG                ?36?
001670  000410                          BR      37$                     ;                       ?37?
001672  005266  000012          35$:    INC     12(SP)                  ; CTLR                  ?40?
001676  000243                          .WORD CLV!CLC

001700  003002                          BGT     36$
001702  000137  002640                  JMP     30$
001706  105766  000004          36$:    TSTB    4(SP)                   ; FLAG                  ?38?
001712  001017                  37$:    BNE     38$
001714  012746  000001                  MOV     #1, -(SP)               ;                       ?41?
001720  012746  000000                  MOV     #CER,0(SP), -(SP)
001724  012746  000002                  MOV     #2, -(SP)
001730  010600                          MOV     SP,R0                   ; SP,*
001732  104417                          TRAP    17
001734  112764  000001  000000          MOVB    #1,D%R(R4)              ; *,*(UNIT)             ?42?
001742  010400                          MOV     R4,R0                   ; UNIT,*                ?42?
001744  104451                          TRAP    51
001746  062706  000006                  ADD     #6,SP                   ;                       ?42?
001752  005204                  38$:    INC     R4                      ; UNIT                  ?39?
001754  020466  000014          39$:    CMP     R4,14(SP)               ; UNIT,*
```

```
001760  002002              BGE     40$
001762  000137  002110'      JMP     19$
001766  123727  000000G 000005  40$:  CMPB    ENTRY,REASON,#5          ;                                  5386
001774  001051              BNE     46$
001776  013704  000000G     41$:    MOV     I$UNIT,R4                ;                                  5390
002002  005003              CLR     R3                               ; UNIT
002004  000401              BR      43$
002006  010300              42$:    MOV     R3,R0                    ; UNIT,*                           5391
002010  104402              TRAP    42
002012  010001              MOV     R0,R1                            ; *,HWPT.REF
002014  005203              INC     R3                               ; UNIT                             5390
002016  020304              43$:    CMP     R3,R4                    ; UNIT,*
002020  002772              BLT     42$
002022  005003              CLR     R3                               ; CTLR                             5393
002024  010346              44$:    MOV     R3,-(SP)                 ; CTLR,*                           5395
002026  012746  000126            MOV     #126.,-(SP)
002032  004737  000000G           JSR     PC,BL$MUL
002036  105060  000056G           CLRB    CST+5(R0)
002042  010316              MOV     R3,(SP)                          ; CTLR,*                           5398
002044  012746  000053            MOV     #53.,-(SP)
002050  004737  000000G           JSR     PC,BL$MUL
002054  012701  000003            MOV     #3,R1                            ; *,OFFSET                 5397
002060  010002              45$:    MOV     R0,R2                    ;                                  5398
002062  060102              ADD     R1,R2                            ; OFFSET,*
002064  006302              ASL     R2
002066  042762  020000  000000G   BIC     #20000,CST(R2)
002074  062701  000012            ADD     #12,R1                           ; *,OFFSET                 5401
002100  020127  000041            CMP     R1,#41                           ; OFFSET,*
002104  003765              BLE     45$
002106  062706  000006            ADD     #6,SP                    ;                                  5394
002112  005203              INC     R3                               ; CTLR                             5393
002114  000243              .WORD CLV!CLC

002116  003742              BLE     44$
002120  123727  000000G 000001  46$:  CMPB    ENTRY,REASON,#1          ;                                  5404
002126  001017              BNE     49$
002130  005037  000000G           CLR     CTLR.CNT                 ;                                  5407
002134  005000              CLR     R0                               ; CTLR                             5409
002136  005760  000000G     47$:    TST     CST(R0)                  ; *(CTLR)                          5411
002142  001402              BEQ     48$
002144  005237  000000G           INC     CTLR.CNT                 ;                                  5413
002150  062700  000126      48$:    ADD     #126,R0                  ; *,CTLR                           5409
002154  000243              .WORD CLV!CLC

002156  003767              BLE     47$
002160  104431              TRAP    51
002162  010037  000000G           MOV     R0,FREE.MEM.ADDR         ;                                  5415
002166  005001              49$:    CLR     R1                               ; UNITS                    5423
002170  005003              50$:    CLR     R3                               ; COUNT                    5424
002172  010300              51$:    MOV     R3,R0                            ; COUNT,*                  5426
002174  060100              ADD     R1,R0                            ; UNITS,*
002176  006300              ASL     R0
002200  005060  000000G           CLR     TALLY(R0)
```

```
002204  005203                          INC     R3                      ; COUNT                 5424
002206  020327  000006                  CMP     R3,#6                   ; COUNT,*
002212  003767                          BLE     51$
002214  062701  000033                  ADD     #33,R1                  ; *,UNITS               5423
002220  020127  000121                  CMP     R1,#121                 ; UNITS,*
002224  003761                          BLE     50$
002226  032766  000001  000010          BIT     #1,10(SP)               ; *,CLEAR.TABLES        5427
002234  001436                          BEQ     55$
002236  005001                          CLR     R1                      ; UNITS                 5429
002240  012703  000007          52$:    MOV     #7,R3                   ; *,COUNT               5430
002244  010300          53$:    MOV     R3,R0                   ; COUNT,*               5431
002246  060100                          ADD     R1,R0                   ; UNITS,*
002250  006300                          ASL     R0
002252  005060  000000G                 CLR     TALLY(R0)
002256  005203                          INC     R3                      ; COUNT                 5430
002260  020327  000032                  CMP     R3,#32                  ; COUNT,*
002264  003767                          BLE     53$
002266  062701  000033                  ADD     #33,R1                  ; *,UNITS               5429
002272  020127  000121                  CMP     R1,#121                 ; UNITS,*
002276  003760                          BLE     52$
002300  032766  000001  000010          BIT     #1,10(SP)               ; *,CLEAR.TABLES        5433
002306  001411                          BEQ     55$
002310  005000                          CLR     R0                      ; CTLR                  5435
002312  105060  000000G         54$:    CLRB    C.ERR.TBL(R0)           ; *(CTLR)               5437
002316  105060  000001G                 CLRB    C.ERR.TBL+1(R0)         ; *(CTLR)               5438
002322  062700  000002                  ADD     #2,R0                   ; *,CTLR                5435
002326  000243                          .WORD CLV!CLC
002330  003770                          BLE     54$
002332  005000          55$:    CLR     R0                      ; CTLR                  5445
002334  105060  000000G         56$:    CLRB    QIO(R0)                 ; *(CTLR)               5446
002340  005200                          INC     R0                      ; CTLR                  5445
002342  000243                          .WORD CLV!CLC

002344  003773                          BLE     56$
002346  005000                          CLR     R0                      ; COUNT                 5448
002350  112760  000377  000000G 57$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)            5449
002356  005200                          INC     R0                      ; COUNT                 5448
002360  020027  000007                  CMP     R0,#7                   ; COUNT,*
002364  003771                          BLE     57$
002366  132737  000001  000000G         BITB    #1,CLK.PRESENT          ;                       5451
002374  001402                          BEQ     58$
002376  005037  177546                  CLR     @#177546                ;                       5454
002402  005037  000000G         58$:    CLR     IODQ.OUT                ;                       5455
002406  005037  000000G                 CLR     IODQ.IN
002412  005037  000000G                 CLR     CRN.HIGH                ;                       5456
002416  005037  000000G                 CLR     CRN.LOW
002422  005000                          CLR     R0                      ;                       5457
002424  104441                          TRAP    41
002426  062706  000030                  ADD     #30,SP                  ;                       5157
002432  000207                          RTS     PC

; Routine Size:  654 words,    Routine Base:  $CODE$ + 1364
```

```
; Maximum stack depth per invocation:  24 words


                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001364'          L$INIT::JSR     PC,LINIT                ;                5457
000004  104411                          TRAP    11
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 4020
; Maximum stack depth per invocation:  2 words
```

```
;    5460  1      %sbttl 'AUTODROP SECTION
;    5461  1
;    5462  1      !+
;    5463  1      ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF
;    5464  1      ! THE "ADR" FLAG WAS SET.  THE UNIT(S) UNDER TEST ARE CHECKED TO
;    5465  1      ! SEE IF THEY WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY
;    5466  1      ! DROPPED FROM TESTING.
;    5467  1      !-
;    5468  1
;    5469  2      BGNAUTO;
;    5470  2
;    5471  2      !IF BIT_TST (SWP_FLAGS, SWF_IRC)
;    5472  2      !then
;    5473  2      !   PRINTF (DBM3);
;    5474  2
;    5475  2      return;
;    5476  2
;    5477  1      ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                      ;                       5459

; Routine Size:  1 word,        Routine Base:  $CODE$ + 4030
; Maximum stack depth per invocation:  0 words

                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  004030'         L$AUTO::JSR     PC,LAUTO                ;                       5475
000004  104461                          TRAP    61
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4032
; Maximum stack depth per invocation:  2 words
```

```
;    5478  1     %sbttl 'CLEANUP CODING SECTION'
;    5479  1
;    5480  1     !+
;    5481  1     ! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
;    5482  1     ! AFTER THE HARDWARE TESTS HAVE BEEN PERFORMED.
;    5483  1     !-
;    5484  1
;    5485  2     BGNCLN;
;    5486  2
;    5487  2     DORPT;
;    5488  2
;    5489  2     !CLRVEC (O_IVEC);                                   ! RETURN ODT TRAP TO DIAGNOSTIC SUPERVISER
;    5490  2
;    5491  2     if .CLK_PRESENT
;    5492  2     then
;    5493  3         begin
;    5494  3         LINE_CLOCK = 0;                                 ! STOP THE LINE-CLOCK
;    5495  3     !   CLRVEC (%o'100');                               ! RETURN LINE-CLOCK'S VECTOR TO SUPERVISOR
;    5496  2         end;
;    5497  2
;    5498  2     incr CTLR from 0 to (MAX_CTLR - 1) do               ! FOR EACH CONTROLLER
;    5499  2
;    5500  2         if (RDRX_ADDR = .CST [.CTLR, IP_ADDR]) neqa 0   ! IF CONTROLLER EXISTS
;    5501  2         then
;    5502  3             begin
;    5503  3
;    5504  3             if .ADDR_VECT_OK
;    5505  3             then
;    5506  4                 begin
;    5507  4
;    5508  4                 if .DCT [.CTLR, STAT] eql ONLINE        ! IF CONTROLLER ALIVE
;    5509  4                 then
;    5510  4
;    5511  4                     incr COUNT from 1 to 10000 do
;    5512  5                         begin
;    5513  5                         DELAY (1);
;    5514  5
;    5515  5                         if .DCT [.CTLR, CRING_CNT] eql 0    ! WAIT TILL OUTSTANDING COMMANDS FINISHED
;    5516  5                         then
;    5517  5                             exitloop;
;    5518  5
;    5519  4                         end;
;    5520  4
;    5521  4                 WRT_RDRX (RCIP, RC_ALL, ALL_ONES);      ! WRITE IP TO STOP DEVICE
;    5522  3                 end;
;    5523  3
;    5524  3             CLRVEC (.CST[.CTLR, VEC_ADDR]);             ! RETURN CONTROLLER'S TRAP VECTOR TO SUPERVISOR
;    5525  2             end;
;    5526  2
;    5527  1     ENDCLN;
```

```
000000  004137  000000G             LCLEAN: JSR     R1,$SAVE5                       ;                               5477
000004  005746                             TST     -(SP)
000006  104424                             TRAP    24                              ;                               5485
000010  132737  000001  000000G            BITB    #1,CLK.PRESENT                  ;                               5491
000016  001402                             BEQ     1$
000020  005037  177546                     CLR     @#177546                        ;                               5494
000024  005005                       1$:   CLR     R5                              ; CTLR                          5498
000026  010546                       2$:   MOV     R5,-(SP)                        ; CTLR,*                        5500
000030  012746  000126                     MOV     #126,-(SP)
000034  004737  000000G                   JSR     PC,BL$MUL
000040  010001                             MOV     R0,R1
000042  022626                             CMP     (SP)+,(SP)+
000044  016137  000000G 000000G            MOV     CST(R1),RDRX.ADDR
000052  001450                             BEQ     10$
000054  132737  000001  000000G            BITB    #1,ADDR.VECT.OK                 ;                               5504
000062  001437                             BEQ     9$
000064  010546                             MOV     R5,-(SP)                        ; CTLR,*                        5508
000066  012746  000022                     MOV     #22,-(SP)
000072  004737  000000G                   JSR     PC,BL$MUL
000076  022626                             CMP     (SP)+,(SP)+
000100  005760  000000G                   TST     DCT(R0)
000104  100022                             BPL     8$
000106  012704  023420                     MOV     #23420,R4                       ; *,COUNT                       5511
000112  012703  000001                3$:  MOV     #1,R3                           ; *,$$TMP2                      5513
000116  001410                       4$:   BEQ     7$
000120  013702  000000G                   MOV     L$DLY,R2                        ; *,$$TMP1
000124  001403                             BEQ     6$
000126  005016                       5$:   CLR     (SP)                            ; $$TMP
000130  005302                             DEC     R2                              ; $$TMP1
000132  001375                             BNE     5$
000134  005303                       6$:   DEC     R3                              ; $$TMP2
000136  000767                             BR      4$
000140  105760  000000G              7$:   TSTB    DCT(R0)                         ;                               5515
000144  001402                             BEQ     8$                              ;                               5517
000146  005304                             DEC     R4                              ; COUNT                         5511
000150  001360                             BNE     3$
000152  012700  177777               8$:   MOV     #-1,R0                          ; *,RC.REG                      5521
000156  010077  000000G                   MOV     R0,@#RDRX.ADDR                  ; RC.REG,*
000162  016100  000002G              9$:   MOV     CST+2(R1),R0                    ;                               5524
000166  042700  177000                     BIC     #177000,R0
000172  104436                             TRAP    36
000174  005205                       10$:  INC     R5                              ; CTLR                          5498
000176  000243                             .WORD CLV!CLC

000200  003712                             BLE     2$
000202  005726                             TST     (SP)+                           ;                               5477
000204  000207                             RTS     PC

; Routine Size:  67 words,    Routine Base:  $CODE$ + 4042
; Maximum stack depth per invocation:  10 words
```

```
                                        .SBTTL  L$CLEAN CLEANUP CODING SECTION

000000  004737  004042'        L$CLEAN::
                                        JSR     PC,LCLEAN                       ;                       5525
000004  104412                          TRAP    12
000006  000207                          RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 4250
; Maximum stack depth per invocation:  2 words
```

```
;    5528  1      %sbttl 'DROP UNIT SECTION'
;    5529  1
;    5530  1      !+
;    5531  1      ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
;    5532  1      ! TO NO LONGER BE TESTED.
;    5533  1      !-
;    5534  1
;    5535  2      BGNDU;
;    5536  2
;    5537  2      local
;    5538  2          UNIT : word,                                                    ! UNIT NUMBER
;    5539  2          PRINT : byte initial (byte (FALSE));                            ! NO PRINTING
;    5540  2
;    5541  2      label
;    5542  2          SEARCH;
;    5543  2
;    5544  3      begin
;    5545  3
;    5546  3      register
;    5547  3          INPUT = 0;                                                      ! R0 = UNIT NO.
;    5548  3
;    5549  3      UNIT = .INPUT;                                                      ! GET UNIT NUMBER
;    5550  2      end;                                                                ! UNDECLARE R0
;    5551  2
;    5552  3      if BIT_TST (SWP_FLAGS, SWF_TRC)
;    5553  2      then
;    5554  2          PRINTF (DBM5, .UNIT);
;    5555  2
;    5556  2      SEARCH :                                                            ! SEARCH BLOCK
;    5557  3      begin
;    5558  3
;    5559  3      incr CTLR from 0 to (MAX_CTLR - 1) do                               ! FOR EACH CNTR
;    5560  3
;    5561  3          incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    5562  3
;    5563  3              if (.CST [.CTLR, .OFFSET + OF_DATA, D_UNIT] eql .UNIT) and      ! IF UNIT MATCHES
;    5564  4                 (.CST [.CTLR, .OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    5565  3              then
;    5566  4                  begin
;    5567  4
;    5568  4                  if (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE) or  ! IF UNIT ALIVE
;    5569  4                     (.DUR [.UNIT] eql DU_ONLINE) or
;    5570  5                     (.DUR [.UNIT] eql DU_PROTECT)
;    5571  4                  then
;    5572  5                      begin
;    5573  5                      PRINT = TRUE;                                       ! O.K. TO PRINT
;    5574  5
;    5575  5                      if (.CST [.CTLR, U_CNT] gtru 0) and
;    5576  6                         (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    5577  5                      then
;    5578  5                          CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;   ! DECREMENT COUNT
;    5579  5
;    5580  5                      if (.CST [.CTLR, U_CNT] eql 0) and
```

```
;    5581  6                    (.CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE)
;    5582  5                then
;    5583  5                    EOP_FLAG = TRUE;                                    ! ALL UNITS OFFLINE
;    5584  5
;    5585  5                CST [.CTLR, .OFFSET + OF_DATA, D_STAT] = OFFLINE;       ! MARK UNIT OFFLINE
;    5586  4                end;                                                    ! IF UNIT ALIVE
;    5587  4
;    5588  4            leave SEARCH;                                               ! EXIT SEARCH BLOCK
;    5589  3            end;                                                        ! IF UNIT FOUND
;    5590  3
;    5591  2     end;
;    5592  2
;    5593  2     if .PRINT or                                                       ! IF OK TO PRINT
;    5594  2        (.DUR [.UNIT] eql DU_CONF) or
;    5595  2        (.DUR [.UNIT] eql DU_INIT) or
;    5596  2        (.DUR [.UNIT] eql DU_ONLINE) or
;    5597  3        (.DUR [.UNIT] eql DU_PROTECT)
;    5598  2     then
;    5599  3         begin
;    5600  3         PRINTF (DU_MSG, .UNIT);                                        ! "UNIT XX DROPPED"
;    5601  3         PRINTF (.DU_RSN [.DUR [.UNIT]]);                               ! REASON
;    5602  2         end;
;    5603  2
;    5604  1     ENDDU;
```

```
                        .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G  LDU:    JSR     R1,$SAVE5                       ;                       5527
000004  005746                   TST     -(SP)
000006  105046                   CLRB    -(SP)                           ; PRINT
000010  010001                   MOV     R0,R1                           ; INPUT,UNIT            5549
000012  032737  000001  000000G  BIT     #1,SWP.FLAGS                    ;                       5552
000020  001411                   BEQ     1$
000022  010146                   MOV     R1,-(SP)                        ; UNIT,*                5554
000024  012746  000000G          MOV     #DBM5,-(SP)
000030  012746  000002           MOV     #2,(SP)
000034  010600                   MOV     SP,R0                           ; SP,*
000036  104417                   TRAP    17
000040  062706  000006           ADD     #6,SP
000044  005005           1$:     CLR     R5                              ; CTLR                  5559
000046  010546           2$:     MOV     R5,-(SP)                        ; CTLR,*                5563
000050  012746  000053           MOV     #53,(SP)
000054  004737  000000G          JSR     PC,BL$MUL
000060  010066  000006           MOV     R0,6(SP)
000064  012703  000003           MOV     #3,R3                           ; *,OFFSET              5561
000070  010300           3$:     MOV     R3,R0                           ; OFFSET,*              5563
000072  066600  000006           ADD     6(SP),R0
000076  006300                   ASL     R0
000100  012702  000000G          MOV     #CST,R2
000104  060002                   ADD     R0,R2
000106  010104                   MOV     R1,R4                           ; UNIT,*
000110  011200                   MOV     (R2),R0
000112  000300                   SWAB    R0
```

```
000114  042700  177760                  BIC     #177760,R0
000120  020004                          CMP     R0,R4
000122  001055                          BNE     9$
000124  032712  040000                  BIT     #40000,(R2)             ;                       5564
000130  001452                          BEQ     9$
000132  005004                          CLR     R4                      ;                       5568
000134  032712  020000                  BIT     #20000,(R2)
000140  001402                          BEQ     4$
000142  005204                          INC     R4
000144  000410                          BR      5$
000146  126127  000000G 000007  4$:     CMPB    DUR(R1),#7              ; *(UNIT),*             5569
000154  001404                          BEQ     5$
000156  126127  000000G 000011          CMPB    DUR(R1),#11             ; *(UNIT),*             5570
000164  001032                          BNE     8$
000166  112766  000001  000004  5$:     MOVB    #1,4(SP)                ; *,PRINT               5573
000174  010516                          MOV     R5,(SP)                 ; CTLR,*                5575
000176  012746  000126                  MOV     #126,-(SP)
000202  004737  000000G                 JSR     PC,BL$MUL
000206  005726                          TST     (SP)+
000210  062700  000004G                 ADD     #CST+4,R0
000214  105760  000001                  TSTB    1(R0)
000220  001404                          BEQ     6$
000222  006004                          ROR     R4                      ;                       5576
000224  105660  000001                  SBCB    1(R0)                   ;                       5578
000230  001006                          BNE     7$                      ;                       5580
000232  032712  020000          6$:     BIT     #20000,(R2)             ;                       5581
000236  001403                          BEQ     7$
000240  112737  000001  000000G         MOVB    #1,EOP.FLAG             ;                       5583
000246  042712  020000          7$:     BIC     #20000,(R2)             ;                       5585
000252  022626                  8$:     CMP     (SP)+,(SP)+             ;                       5586
000254  000411                          BR      10$
000256  062703  000012          9$:     ADD     #12,R3                  ; *,OFFSET              5561
000262  020327  000041                  CMP     R3,#41                  ; OFFSET,*
000266  003700                          BLE     3$
000270  022626                          CMP     (SP)+,(SP)+
000272  005205                          INC     R5                      ; CTLR                  5559
000274  000243                          .WORD CLV!CLC

000276  003663                          BLE     2$
000300  032716  000001          10$:    BIT     #1,(SP)                 ; *,PRINT               5593
000304  001020                          BNE     11$
000306  126127  000000G 000001          CMPB    DUR(R1),#1              ; *(UNIT),*             5594
000314  001414                          BEQ     11$
000316  126127  000000G 000002          CMPB    DUR(R1),#2              ; *(UNIT),*             5595
000324  001410                          BEQ     11$
000326  126127  000000G 000007          CMPB    DUR(R1),#7              ; *(UNIT),*             5596
000334  001404                          BEQ     11$
000336  126127  000000G 000011          CMPB    DUR(R1),#11             ; *(UNIT),*             5597
000344  001024                          BNE     12$
000346  010146                  11$:    MOV     R1,-(SP)                ; UNIT,*                5600
000350  012746  000000G                 MOV     #DU.MSG,-(SP)
000354  012746  000002                  MOV     #2, (SP)
000360  010600                          MOV     SP,R0                   ; SP,*
```

```
000362  104417                          TRAP    17
000364  116101  000006                  MOVB    DUR(R1),R1              ; *(UNIT),*                     5601
000370  042701  177400                  BIC     #177400,R1
000374  006301                          ASL     R1
000376  016116  000006                  MOV     DU.RSN(R1),(SP)
000402  012746  000001                  MOV     #1,-(SP)
000406  010600                          MOV     SP,R0                   ; SP,*
000410  104417                          TRAP    17
000412  062706  000010                  ADD     #10,SP                  ;                               5599
000416  022626                  12$:    CMP     (SP)+,(SP)+             ;                               5527
000420  000207                          RTS     PC

; Routine Size:  137 words,     Routine Base:  $CODE$ + 4260
; Maximum stack depth per invocation:  14 words



                                        .SBTTL  L$DU DROP UNIT SECTION
000000  004737  004260'         L$DU::  JSR     PC,LDU                  ;                               5602
000004  104453                          TRAP    53
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4702
; Maximum stack depth per invocation:  2 words
```

```
;    5600  1      %sbttl 'ADD UNIT SECTION
;    5601  1
;    5602  1      !.
;    5603  1      ! THE ADD UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES
;    5604  1      !  TO BE EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK
;    5605  1      !  TO THE TEST CYCLE.
;    5606  1      !
;    5607  1
;    5608  1      BGNAU;
;    5609  1
;    5610  1      local
;    5611  1          STINDX : word,
;    5612  1          ENDIDX : word;
;    5613  1
;    5614  1      register
;    5615  1          UNIT = 0;                               ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;    5616  1
;    5617  1      IF BIT TST (SWP_FLAGS, SWS_CST)
;    5618  1      then
;    5619  1          begin                                   ! IF CLEAR STAT. TABLES TRUE....
;    5620  1          STINDX = .UNIT * TALLY_LEN;             ! ZERO OUT
;    5621  1          ENDIDX = .STINDX + TALLY_LEN - 1;       !    ADDED
;    5622  1          incr COUNT from .STINDX to .ENDIDX do   !    UNIT'S
;    5623  1              TALLY [.COUNT] = 0;                 !    STATISTICS
;    5624  1
;    5625  1          end;
;    5626  1
;    5627  1      ENDAU;


                                .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G    LAU:  JSR     R1,$SAVE2                 ;                              5604
000004  105737  000000G          TSTB    SWP.FLAGS                 ;                              5617
000010  100003                  BPL     5$
000012  010046                  MOV     R0, -(SP)                 ; UNIT,*                       5620
000014  012746  000055          MOV     #55, -(SP)
000020  004737  000000G          JSR     PC,@#$MUL
000024  010002                  MOV     R0,R2
000026  062702  000052          ADD     #52,R2                    ; *,ENDIDX
000032  010001                  MOV     R0,R1                     ; STINDX,COUNT                 5622
000034  005501                  DEC     R1                        ; COUNT
000036  000404                  BR      2$
000040  010100              1$: MOV     R1,R0                     ; COUNT,*                      5623
000042  006300                  ASL     R0
000044  005060  000000G        CLR     TALLY(R0)
000050  005201              2$: INC     R1                        ; COUNT
000052  020102                  CMP     R1,R2                     ; COUNT,ENDIDX
000054  003771                  BLE     1$
000056  022626                  CMP     (SP)+,(SP)+
000060  000207              5$: RTS     PC                        ;                              5626

; Routine Size:  25 words,    Routine Base:  $CODE$ + 4710
```

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

                                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  00471C'                      L$AU::  JSR     PC,L$AU                 !                       5651
000004  104452                                      TRAP    %0
000006  000207                                      RTS     PC

; Routine Size:  4 words,        Routine Base:  $CODE$ + 4774
; Maximum stack depth per invocation:  2 words
```

```
;    5651  1      %sbttl 'NON-EXISTENT MEMORY TRAP HANDLER'
;    5652  1
;    5653  1      !+
;    5654  1      !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
;    5655  1      !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON EXISTENT
;    5656  1      !       MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RDRX
;    5657  1      !       REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
;    5658  1      !-
;    5659  1
;    5660  1      BGNSRV (NEX TRAP);
;    5661  1
;    5662  1      NEX = TRUE;                                     ! NEX TRAP OCCURRED
;    5663  1
;    5664  1      ENDSRV;


                                        .SBTTL  NEX.TRAP NON-EXISTENT MEMORY TRAP HANDLER
000000  012737  000001  000006  NEX.TRAP::
                                        MOV     #1,NEX                  ;                       5662
000006  000002                          RTI                             ;                       5663

; Routine Size:  4 words,       Routine Base:  $CODE$ + 5004
; Maximum stack depth per invocation:  0 words
```

```
;    5648  1     %sbttl 'TIME OF DAY'
;    5649  1
;    5650  1     !+
;    5651  1     !       THIS INTERRUPT SERVICE ROUTINE KEEPS TRACK OF THE TIME-OF-DAY
;    5652  1     !
;    5653  1
;    5654  2     BGNSRV (TIME);
;    5655  2
;    5656  2     CLK_TICKS = .CLK_TICKS + 1;             ! INCREMENT CLOCK TICKS
;    5657  2
;    5658  2     if .CLK_TICKS gequ 3600
;    5659  2     then
;    5660  3         begin
;    5661  3         MINUTES = .MINUTES + 1;             ! UPDATE MINUTE COUNT
;    5662  3         CLK_TICKS = 0;
;    5663  2         end;
;    5664  2
;    5665  2     if .MINUTES gequ 60
;    5666  2     then
;    5667  3         begin
;    5668  3         HOURS = .HOURS + 1;                 ! UPDATE HOUR COUNT
;    5669  3         MINUTES = 0;
;    5670  2         end;
;    5671  2
;    5672  2     if .HOURS gequ 24
;    5673  2     then
;    5674  2         HOURS = 0;                          ! RATIONALIZE HOURS
;    5675  2
;    5676  1     ENDSRV;


                                        .SBTTL  TIME TIME OF DAY
000000  005237  000000G           TIME::  INC     CLK_TICKS                       ;5656
000004  023727  000000G 007020          CMP     CLK_TICKS,#7020                 ;5658
000012  103404                          BLO     1$
000014  105237  000000G                 INCB    MINUTES                         ;5661
000020  005037  000000G                 CLR     CLK_TICKS                       ;5662
000024  123727  000000G 000074  1$:     CMPB    MINUTES,#74                     ;5665
000032  103404                          BLO     2$
000034  105237  000000G                 INCB    HOURS                           ;5668
000040  105037  000000G                 CLRB    MINUTES                         ;5669
000044  123727  000000G 000030  2$:     CMPB    HOURS,#30                       ;5672
000052  103402                          BLO     3$
000054  105037  000000G                 CLRB    HOURS                           ;5674
000060  000002                  3$:     RTI                                     ;5654

; Routine Size:  25 words,    Routine Base:  $CODE$ + 5014
; Maximum stack depth per invocation:  0 words
```

```
;    5677  1      %sbttl 'GLOBAL ROUTINES'
;    5678  1
;    5679  1      global routine SET_CPAR (CTLR) : novalue =
;    5680  1
;    5681  1      !+
;    5682  1      !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    5683  1      !       FOR THE GIVEN CONTROLLER NUMBER.
;    5684  1      !
;    5685  1      !       INPUTS:
;    5686  1      !               CTLR  - CONTROLLER NUMBER
;    5687  1      !
;    5688  1      !       IMPLICIT OUTPUTS:
;    5689  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    5690  1      !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;    5691  1      !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;    5692  1      !               RDRX_ADDR - ADDRESS OF CONTROLLER'S IP REGISTER
;    5693  1      !-
;    5694  1
;    5695  2          begin
;    5696  2          CCTLR = .CTLR;                          ! SET CURRENT CONTROLLER NUMBER
;    5697  2          CST_ADDR = CST + (.CTLR * CST_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S CST
;    5698  2          DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2);     ! CALCULATE ADDRESS OF CONTROLLER'S DCT
;    5699  2          RDRX_ADDR = .CST_ADDR [IP_ADDR];            ! GET CONTROLLER'S DEVICE ADDRESS
;    5700  1          end;

                                        .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                 SET.CPAR::
                                        MOV     R1,-(SP)                ;
000002  016601  000004                  MOV     4(SP),R1                ; CTLR,*                5696
000006  010137  000000G                 MOV     R1,CCTLR
000012  010146                          MOV     R1,-(SP)                ;                       5697
000014  012746  000126                  MOV     #126,(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  062700  000000G                 ADD     #CST,R0
000030  010037  000000G                 MOV     R0,CST.ADDR
000034  010116                          MOV     R1,(SP)                 ;                       5698
000036  012746  000022                  MOV     #22,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  062700  000000G                 ADD     #DCT,R0
000052  010037  000000G                 MOV     R0,DCT.ADDR
000056  017737  000000G 000000G         MOV     @CST.ADDR,RDRX.ADDR     ;                       5699
000064  062706  000006                  ADD     #6,SP                   ;                       5700
000070  012601                          MOV     (SP)+,R1                ;                       5679
000072  000207                          RTS     PC

; Routine Size:  30 words,    Routine Base:  $CODE$ + 5076
; Maximum stack depth per invocation:  5 words
```

```
;    5701  1     global routine SET_UPAR (OFFSET) : novalue =
;    5702  1     !+
;    5703  1     !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
;    5704  1     !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
;    5705  1     !
;    5706  1     !       INPUTS:
;    5707  1     !               OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
;    5708  1     !                        DESCRIBES A UNIT
;    5709  1     !
;    5710  1     !       IMPLICIT INPUTS:
;    5711  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    5712  1     !
;    5713  1     !       IMPLICIT OUTPUTS:
;    5714  1     !               CUOFF - CURRENT UNIT'S CST OFFSET
;    5715  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    5716  1     !               L$LUN - CURRENT UNIT NUMBER (DRS UNIT NUMBER)
;    5717  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    5718  1     !-
;    5719  2         begin
;    5720  2         CUOFF = .OFFSET;
;    5721  2         CDISK = .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM];
;    5722  2         L$LUN = .CST_ADDR [.OFFSET + OF_DATA, D_UNIT];
;    5723  2         T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
;    5724  1         end;

                                .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146          SET.UPAR::
                                MOV     R1,-(SP)                ;                               5701
000002  016637  000004  000000G MOV     4(SP),CUOFF             ; OFFSET,*                      5720
000010  016600  000004          MOV     4(SP),R0                ; CUOFF,*                       5721
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                 ;                               5722
000036  000301                  SWAB    R1
000040  042701  177760          BIC     #177760,R1
000044  010137  000000G         MOV     R1,L$LUN
000050  010146                  MOV     R1,-(SP)                ; L$LUN,*                       5723
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)+             ;                               5724
000074  012601                  MOV     (SP)+,R1                ;                               5701
000076  000207                  RTS     PC

; Routine Size:  30 words,      Routine Base:  $CODE$ + 517C
; Maximum stack depth per invocation:  4 words
```

```
;    5725  1
;    5726  1     global routine GET_PKT (CTLR) =
;    5727  1
;    5728  1     !+
;    5729  1     !       THIS ROUTINE SEARCHES THE MSCP PACKET POOL ALLOCATION TABLE (PKT_USE)
;    5730  1     !       FOR A FREE MSCP PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;    5731  1     !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED
;    5732  1     !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    5733  1     !
;    5734  1     !       INPUTS:
;    5735  1     !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    5736  1     !-
;    5737  1
;    5738  2         begin
;    5739  2
;    5740  2         local
;    5741  2             index : signed word initial (-1),
;    5742  2             RING_ADDR : word,
;    5743  2             PACKET_OWNED : byte,
;    5744  2             NEXT_PACKET : byte;
;    5745  2
;    5746  2
;    5747  2         NEXT_PACKET = .NEXT_PKT_USE;                    ! NEXT PACKET TO TRY
;    5748  2
;    5749  2         incr COUNT from 0 to (PKT_CNT - 1) do           ! FOR EACH ENTRY IN ALLOCATION TABLE
;    5750  3             begin
;    5751  3             PACKET_OWNED = FALSE;
;    5752  3
;    5753  3             if .PKT_USE [.NEXT_PACKET] lss 0            ! IF ENTRY INDICATES FREE PACKET
;    5754  3
;    5755  3             then
;    5756  4                 begin
;    5757  4                 RING_ADDR = .DCT_ADDR [RR_BEG];         ! FIRST RESPONSE PACKET'S ADDRESS
;    5758  4
;    5759  4                 incr I from 1 to (RRING_LEN + CRING_LEN) do ! FOR EACH PACKET ADDRESS
;    5760  4
;    5761  4                     if (..RING_ADDR eqla .MSCP_PKT [.NEXT_PACKET, PKT_LO]) and
;    5762  5                        (((.RING_ADDR + 2) and ED_OWN) eql ED_OWN)
;    5763  5
;    5764  4                     then
;    5765  5                         begin                           ! CHECK ADDRESS AND OWNERSHIP
;    5766  5                         PACKET_OWNED = TRUE;            ! PACKET OWNED BY CONTROLLER
;    5767  5                         exitloop;
;    5768  5                         end
;    5769  4                     else
;    5770  4                         RING_ADDR = .RING_ADDR + 4;     ! ADDRESS OF NEXT PACKET IN RING
;    5771  4
;    5772  4                 if not .PACKET_OWNED                    ! IF NOT ALREADY USED
;    5773  4
;    5774  4                 then
;    5775  5                     begin
;    5776  5                     PKT_USE [.NEXT_PACKET] = .CTLR;     ! ALLOCATE PACKET TO CONTROLLER
;    5777  5                     index = .NEXT_PACKET;
```

```
;    5778  5
;    5779  5                    incr J from 2 to (PKT_LEN - 1) do       ! ZERO OUT PACKET
;    5780  5                        MSCP_PKT [.NEXT_PACKET, .J, 0, 16, 0] = 0;
;    5781  5
;    5782  5                    exitloop;                               ! DONE
;    5783  5
;    5784  4                    end;
;    5785  4
;    5786  3                end;
;    5787  3
;    5788  3            NEXT_PACKET = .NEXT_PACKET + 1;                 ! TRY NEXT PACKET IN RING
;    5789  3
;    5790  3            if .NEXT_PACKET gequ PKT_CNT
;    5791  3
;    5792  3            then
;    5793  3                NEXT_PACKET = 0;                            ! IF BEYOND ALL PACKETS, START AT THE TOP
;    5794  3
;    5795  2            end;
;    5796  2
;    5797  2        if (.index geq 0) and                               ! IF PACKET FOUND
;    5798  3           (.PKT_USE [.index] geq 0)
;    5799  2        then
;    5800  2
;    5801  3            begin
;    5802  3            MSCP_PKT [.index, MSGLEN] = SZ_GEN;             ! PACKET SIZE - ONLY ONLINE AND SCC CHANGE IT
;    5803  3            MSCP_PKT [.index, CREDITS] = 1;                 ! CREDIT SIZE
;    5804  3            NEXT_PKT_USE = .NEXT_PACKET + 1;                ! NEXT PACKET TO ALLOCATE
;    5805  3
;    5806  3            if.NEXT_PKT_USE gequ PKT_CNT
;    5807  3            then
;    5808  3                NEXT_PKT_USE = 0;                           ! CYCLE BACK TO BEGINNING IF AT END
;    5809  3
;    5810  2            end;
;    5811  2
;    5812  2        return .index;
;    5813  2
;    5814  1        end;


                                .SBTTL  GET.PKT GLOBAL ROUTINES
000000  004137  000000G    GET.PKT::
                            JSR     R1,$SAVE5                       ;                               5126
000004  162706  000006              SUB     #6,SP
000010  012704  177777              MOV     #-1,R4                  ; *,INDEX                       5138
000014  113766  000000G 000004      MOVB    NEXT.PKT,USE ?(SP)      ; *,NEXT.PACKET                 5147
000022  012766  000014  000002      MOV     #14,2(SP)               ; *,COUNT                       5149
000030  105016              1$:     CLRB    (SP)                    ; PACKET.OWNED                  5151
000032  005001                      CLR     R1                      ;                               5153
000034  156601  000004              BISB    4(SP),R1                ; NEXT.PACKET,*
000040  105761  000000G             TSTB    PKT.USE(R1)
000044  002072                      BGE     7$
000046  013700  000000G             MOV     DCT.ADDR,R0             ;                               5159
000052  016005  000004              MOV     4(R0),R5                ; *,RING.ADDR
```

```
000056  010146                          MOV     R1,-(SP)                ;                               5761
000060  012746  000106                  MOV     #106,-(SP)
000064  004737  000000G                 JSR     PC,BL$MUL
000070  012702  000010                  MOV     #10,R2                  ; *,I                           5759
000074  021560  000000G         2$:     CMP     (R5),MSCP.PKT(R0)       ; RING.ADDR,*                   5761
000100  001014                          BNE     3$
000102  012703  000002                  MOV     #2,R3                   ;                               5762
000106  060503                          ADD     R5,R3                   ; RING.ADDR,*
000110  042703  077777                  BIC     #77777,R3
000114  020327  100000                  CMP     R3,#-100000
000120  001004                          BNE     3$
000122  112766  000001  000004          MOVB    #1,4(SP)                ; *,PACKET.OWNED                5766
000130  000404                          BR      4$                      ;                               5765
000132  062705  000004         3$:     ADD     #4,R5                   ; *,RING.ADDR                   5770
000136  005302                          DEC     R2                      ; I                             5759
000140  001355                          BNE     2$
000142  032766  000001  000004  4$:     BIT     #1,4(SP)                ; *,PACKET.OWNED                5772
000150  001027                          BNE     6$
000152  116661  000030  000000G         MOVB    30(SP),PKT.USE(R1)      ; CTLR,*                        5776
000160  010104                          MOV     R1,R4                   ; *,INDEX                       5777
000162  010116                          MOV     R1,(SP)                 ;                               5780
000164  012746  000043                  MOV     #43,-(SP)
000170  004737  000000G                 JSR     PC,BL$MUL
000174  005726                          TST     (SP)+
000176  012702  000002                  MOV     #2,R2                   ; *,J                           5779
000202  010003         5$:     MOV     R0,R3                   ;                               5780
000204  060203                          ADD     R2,R3                   ; J,*
000206  006303                          ASL     R3
000210  005063  000000G                 CLR     MSCP.PKT(R3)
000214  005202                          INC     R2                      ; J                             5779
000216  020227  000042                  CMP     R2,#42                  ; J,*
000222  003767                          BLE     5$
000224  022626                          CMP     (SP)+,(SP)+             ;                               5775
000226  000414                          BR      9$
000230  022626         6$:     CMP     (SP)+,(SP)+             ;                               5756
000232  105266  000004         7$:     INCB    4(SP)                   ; NEXT.PACKET                   5788
000236  126627  000004  000014          CMPB    4(SP),#14               ; NEXT.PACKET,*                 5790
000244  103402                          BLO     8$
000246  105066  000004                  CLRB    4(SP)                   ; NEXT.PACKET                   5793
000252  005366  000002         8$:     DEC     2(SP)                   ; COUNT                         5749
000256  001264                          BNE     1$
000260  005704         9$:     TST     R4                      ; INDEX                         5797
000262  002435                          BLT     11$
000264  105764  000000G                 TSTB    PKT.USE(R4)             ; *(INDEX)                      5798
000270  002432                          BLT     11$
000272  010446                          MOV     R4, (SP)                ; INDEX,*                       5802
000274  012746  000106                  MOV     #106,-(SP)
000300  004737  000000G                 JSR     PC,BL$MUL
000304  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000312  142760  000017  000010G         BICB    #17,MSCP.PKT+10(R0)     ;                               5803
000320  152760  000001  000010G         BISB    #1,MSCP.PKT+10(R0)
000326  005000                          CLR     R0                      ;                               5804
000330  156600  000010                  BISB    10(SP),R0               ; NEXT.PACKET,*
```

```
000334  005200                          INC     R0
000336  110037  000000G                 MOVB    R0,NEXT.PKT.USE
000342  120027  000014                  CMPB    R0,#14                  ; NEXT.PKT.USE,*                5806
000346  103402                          BLO     10$
000350  105037  000000G                 CLRB    NEXT.PKT.USE            ;                               5808
000354  022626                  10$:    CMP     (SP)+,(SP)+             ;                               5801
000356  010400                  11$:    MOV     R4,R0                   ; INDEX,*                       5748
000360  062706  000006                  ADD     #6,SP                   ;                               5726
000364  000207                          RTS     PC

; Routine Size:  123 words,    Routine Base:  $CODE$ + 5272
; Maximum stack depth per invocation:  13 words


;    5815  1
;    5816  1
```

```
;    5817  1
;    5818  1
;    5819  1     global routine PUT_PKT (index) : novalue =
;    5820  1
;    5821  1     !+
;    5822  1     !       THE MSCP PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    5823  1     !       ROUTINE.
;    5824  1     !-
;    5825  1
;    5826  1
;    5827  2         begin
;    5828  2
;    5829  2
;    5830  2         local
;    5831  2             RING_ADDR : word,
;    5832  2             OWNER : word;
;    5833  2
;    5834  2         RING_ADDR = .DCT_ADDR [RR_BEG];                                 ! ADDRESS IN FIRST RESPONSE RING
;    5835  2
;    5836  2         incr COUNT from 1 to (RRING_LEN + CRING_LEN) do                 ! FOR EACH ADDRESS IN THE RINGS
;    5837  3             begin
;    5838  3
;    5839  3             if .MSCP_PKT [.index, PKT_LO] eqla ..RING_ADDR              ! IF ADDRESS MATCHES
;    5840  3
;    5841  3             then
;    5842  4                 begin
;    5843  4                 OWNER = .RING_ADDR + 2;                                 ! ADDRESS OF OWNERSHIP WORD
;    5844  4                 .OWNER = ..OWNER and (not (ED_OWN)) and (not (ED_FLAG)) ;   ! GIVE OWNERSHIP TO HOST
;    5845  3                 end;
;    5846  3
;    5847  3
;    5848  3             RING_ADDR = .RING_ADDR + 4;                                 ! LOOK AT NEXT PACKET ADDRESS IN RING
;    5849  2             end;
;    5850  2
;    5851  2
;    5852  2         PKT_USE [.index] = -1;
;    5853  2
;    5854  1         end;


                                .SBTTL  PUT_PKT GLOBAL ROUTINES
000000  004137  0000006   PUT_PKT::
                            JSR     R1,$SAVE4                   ;                       5817
000004  013700  0000006            MOV     DCT_ADDR,R0         ;                       5834
000010  016001  000004             MOV     4(R0),R1            ; *,RING_ADDR
000014  016602  000014             MOV     14(SP),R2           ; INDEX,*               5839
000020  010246                     MOV     R2,-(SP)
000022  012746  000106             MOV     #106,-(SP)
000026  004737  000000G            JSR     PC,@#$MUL
000032  012704  000010             MOV     #10,R4              ; *,COUNT               5836
000036  026011  0000006    1$:     CMP     MSCP_PKT(R0),(R1)   ; *,RING_ADDR           5839
000042  001005                     BNE     2$
000044  012703  000002             MOV     #2,R3               ; *,OWNER               5843
```

```
000050  060103                          ADD     R1,R3                   ; RING,ADDR,OWNER
000052  042713  140000                  BIC     #140000,(R3)            ; *,OWNER                       5844
000056  062701  000004          2$:     ADD     #4,R1                   ; *,RING,ADDR                   5848
000062  005304                          DEC     R4                      ; COUNT                         5836
000064  001364                          BNE     1$
000066  112762  000377  000000G         MOVB    #377,PKT.USE(R2)        ;                               5852
000074  022626                          CMP     (SP)+,(SP)+             ;                               5827
000076  000207                          RTS     PC                      ;                               5819

; Routine Size:  30 words,    Routine Base:  $CODE$ + 5660
; Maximum stack depth per invocation:  8 words


;   5855  1
;   5856  1
```

```
;    5857  1      routine PUTA_PKT (CTLR) : novalue =
;    5858  1
;    5859  1      !+
;    5860  1      !       THIS ROUTINE DEALLOCATES ALL MSCP PACKETS WHICH HAVE BEEN ALLOCATED
;    5861  1      !       TO A PARTICULAR CONTROLLER.
;    5862  1      !
;    5863  1      !       INPUTS:
;    5864  1      !               CTLR - CONTROLLER NUMBER
;    5865  1      !-
;    5866  1
;    5867  1          incr COUNT from 0 to (PKT_CNT - 1) do       ! FOR EACH ENTRY IN ALLOCATION TABLE
;    5868  1
;    5869  1              if .PKT_USE [.COUNT] eql .CTLR          ! IF PACKET IS ALLOCATED TO GIVEN CONTROLLER
;    5870  1              then
;    5871  1                  PKT_USE [.COUNT] = -1;              ! DEALLOCATE IT


                                        .SBTTL  PUTA,PKT GLOBAL ROUTINES
000000  010146                 PUTA.PKT:
                                        MOV     R1,-(SP)                ;                               5857
000002  005000                         CLR     R0                      ; COUNT                         5867
000004  116001  000000G        1$:     MOVB    PKT.USE(R0),R1          ; *(COUNT),*                    5869
000010  020166  000004                  CMP     R1,4(SP)                ; *,CTLR
000014  001003                         BNE     2$
000016  112760  000377  000000G        MOVB    #377,PKT.USE(R0)        ; *,*(COUNT)                    5871
000024  005200                 2$:     INC     R0                      ; COUNT                         5867
000026  020027  000013                 CMP     R0,#13                  ; COUNT,*
000032  003764                         BLE     1$
000034  012601                         MOV     (SP)+,R1                ;                               5857
000036  000207                         RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 5760
; Maximum stack depth per invocation:  2 words
```

```
;    5872  1      global routine GET_RETPKT (CTLR) =
;    5873  1
;    5874  1      !++
;    5875  1      !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;    5876  1      !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER.  IF ONE IS
;    5877  1      !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;    5878  1      !       THE CALLER, OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;    5879  1      !
;    5880  1      !       INPUTS:
;    5881  1      !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;    5882  1      !
;    5883  1
;    5884  2          begin
;    5885  2
;    5886  2          local
;    5887  2              index : signed word initial (-1);       ! ASSUME NONE AVAILABLE
;    5888  2
;    5889  2          incr COUNT from 0 to (RP_CNT - 1) do        ! FOR EACH ENTRY IN TABLE
;    5890  2
;    5891  2              if .RP_USE [.COUNT] eql 0               ! IF FREE RETPKT IS FOUND
;    5892  3              then
;    5893  3                  begin
;    5894  3                  RP_USE [.COUNT] = .CTLR;            ! ALLOCATE RETURN PACKET TO CONTROLLER
;    5895  3                  index = .COUNT;
;    5896  3
;    5897  3                  incr J from 0 to (RP_LEN - 1) do    ! ZERO OUT RETPKT
;    5898  3                      RETPKT [.COUNT, .J, 0, 16, 0] = 0;
;    5899  3
;    5900  3                  exitloop;                           ! DONE
;    5901  2                  end;
;    5902  2
;    5903  2          return .index;                              ! RETURN PACKET INDEX (OR -1) TO CALLER
;    5904  1          end;


                                .SBTTL  GET_RETPKT GLOBAL ROUTINES
00000000 004157 00000G  GET_RETPKT::
                        JSR     R1,$SAVE4
00000004 012704 177777          MOV     #-1,R4          ;       #INDEX
00000010 005001                 CLR     R1              ; COUNT
00000012 105761 000000G 1$:     TSTB    RP_USE(R1)      ; #(COUNT)
00000016 002025                 BGE     2$
00000020 116661 000014 000000G  MOVB    14(SP),RP_USE(R1)       ; CTLR,#(COUNT)
00000026 010104                 MOV     R1,R4           ; COUNT,INDEX
00000030 010146                 MOV     R1,-(SP)        ; COUNT,-(SP)
00000032 012746 000020          MOV     #20,-(SP)       ;
00000036 004737 000000G         JSR     PC,$INDEX
00000042 062626                 CMP     (SP)+,(SP)+
00000044 005001                 CLR     R1              ;
00000046 010004                 MOV     R0,R4           ;
00000050 063004                 ADD     R1,R4           ;
00000052 005004                 CLR     R4
00000054 005064 000000G         CLR     RETPKT(R4)
```

```
000060  005202                    INC     R2              ; J                     5897
000062  020227  000025            CMP     R2,#25          ; J,*
000066  003767                    BLE     2$
000070  000404                    BR      4$              ;                       5875
000072  005201           3$:      INC     R1              ; COUNT                 5889
000074  020127  000007            CMP     R1,#7           ; COUNT,*
000100  003744                    BLE     1$
000102  010500           4$:      MOV     R5,R0           ; INDEX,*               5884
000104  000207                    RTS     PC              ;                       5872

; Routine Size:  35 words,      Routine Base:  $CODE$ + 6020
; Maximum stack depth per invocation:  8 words
```

```
;    5905  1      global routine PUT_RETPKT (index) : novalue =
;    5906  1
;    5907  1      !+
;    5908  1      !       THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;    5909  1      !       ROUTINE.
;    5910  1      !
;    5911  1
;    5912  1          RP_USE [.index] = 1;


                                        .SBTTL  PUT_RETPKT GLOBAL ROUTINES
000000  016600  000002              PUT_RETPKT::
                                        MOV     2(SP),R0                ; INDEX,*                       5912
000004  112760  000377  000006         MOVB    #377,RP_USE(R0)
000012  000207                          RTS     PC                      ;                               5905

; Routine Size:  5 words,       Routine Base:  $CODE$ + 6126
; Maximum stack depth per invocation:  0 words
```

```
;    5913  1
;    5914  1
;    5915  1     global routine GET_IO_BUFF (ADDR) : novalue =
;    5916  1
;    5917  1
;    5918  1     !+
;    5919  1     !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
;    5920  1     !       POOL.
;    5921  1     !
;    5922  1     !       INPUTS:
;    5923  1     !               ADDR    ADDRESS TO STORE THE 2 WORD BUFFER DESCRIPTOR
;    5924  1     !
;    5925  1     !       IMPLICIT INPUTS:
;    5926  1     !               CCTLR   CURRENT CONTROLLER NUMBER
;    5927  1     !
;    5928  1     !       OUTPUTS:
;    5929  1     !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
;    5930  1     !               WORDS AT "ADDR" AND "ADDR + 2", OTHERWISE, A ZERO IS RETURNED
;    5931  1     !               AT 'ADDR" IF NO BUFFERS ARE AVAILABLE.
;    5932  1     !-
;    5933  1
;    5934  1
;    5935  1
;    5936  2     begin
;    5937  2     .ADDR = 0;                                          ! ASSUME FAILURE
;    5938  2
;    5939  2     incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do   ! FOR EACH ENTRY IN BUFFER TABLE
;    5940  2
;    5941  2         if .BUFF_OWN [.COUNT] lss 0                     ! IF BUFFER IS FREE
;    5942  2
;    5943  2         then
;    5944  2
;    5945  3             begin
;    5946  3             BUFF_OWN [.COUNT] = .CCTLR;                 ! ALLOCATE BUFFER TO CONTROLLER
;    5947  3             .ADDR = .BUFF_ADDR [.COUNT];                ! RETURN BUFFER DESCRIPTOR
;    5948  3             exitloop;                                   ! DONE
;    5949  2             end;
;    5950  2
;    5951  2
;    5952  1     end;                                                ! ROUTINE GET_IO_BUFF


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

```
000036  000404                          BR      3$                              ;                       5945
000040  005201                  2$:     INC     R1                              ; COUNT                 5949
000042  020127  000007                  CMP     R1,#7                           ; COUNT,#
000046  003760                          BLE     1$
000050  012601                  3$:     MOV     (SP)+,R1                        ;                       5915
000052  000207                          RTS     PC

; Routine Size:  22 words,    Routine Base:  $CODE$ + 6142
; Maximum stack depth per invocation:  2 words


;    5954  1
;    5954  1
```

```
;    5955  1      global routine PUT_IO_BUFF (ADDR) : novalue =
;    5956  1
;    5957  1      !+
;    5958  1      !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;    5959  1      !       TO THE BUFFER POOL.
;    5960  1      !
;    5961  1      !       INPUTS:
;    5962  1      !               ADDR - ADDRESS OF THE 2 WORD BUFFER DESCRIPTOR TO BE
;    5963  1      !                       DEALLOCATED
;    5964  1      !-
;    5965  1
;    5966  1          incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do       ! FOR EACH ENTRY IN BUFFER TABLE
;    5967  1
;    5968  1          if .BUFF_ADDR [.COUNT] eqla ..ADDR                          ! IF THIS IS THE BUFFER'S ENTRY
;    5969  1          then
;    5970  2              begin
;    5971  2              BUFF_OWN [.COUNT] = -1;                                 ! DEALLOCATE BUFFER
;    5972  2              exitloop;                                               ! DONE
;    5973  1              end;


                                        .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  010146                      PUT.IO.BUFF::
                                    MOV     R1,-(SP)                ;                       5955
000002  005001                      CLR     R1                      ; COUNT                 5966
000004  010100                  1$: MOV     R1,R0                   ; COUNT,*               5968
000006  006300                      ASL     R0
000010  026076  000000G 000004      CMP     BUFF.ADDR(R0),@4(SP)    ; *,ADDR
000016  001004                      BNE     2$
000020  112761  000377  000000G     MOVB    @377,BUFF.OWN(R1)       ; *,*(COUNT)            5971
000026  000404                      BR      3$                      ;                       5970
000030  005201                  2$: INC     R1                      ; COUNT                 5966
000032  020127  000007              CMP     R1,@7                   ; COUNT,*
000036  003762                      BLE     1$
000040  012601                  3$: MOV     (SP)+,R1                ;                       5955
000042  000207                      RTS     PC

; Routine Size:  18 words,    Routine Base:  $CODE$ + 6D16
; Maximum stack depth per invocation:  2 words
```

```
;    5974  1      global routine PUTA_BUFF : novalue =
;    5975  1
;    5976  1      !+
;    5977  1      !       THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
;    5978  1      !       THE CURRENT CONTROLLER (CCTLR).
;    5979  1      !-
;    5980  1
;    5981  1          incr COUNT from 0 to (QIO_PER_CTLR * MAX_CTLR - 1) do        ! FOR EACH ENTRY IN BUFFER TABLE
;    5982  1
;    5983  1              if .BUFF_OWN [.COUNT] eql .CCTLR                         ! IF THIS BUFFER ALLOCATED TO CURRENT CONTROLLER
;    5984  1              then
;    5985  1                  BUFF_OWN [.COUNT] = -1;                              ! DEALLOCATE IT

                                .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                PUTA.BUFF::
                                MOV     R1,-(SP)                        ;                                       5974
000002  005000                 CLR     R0                              ; COUNT                                 5981
000004  116001  000000G        1$:     MOVB    BUFF.OWN(R0),R1         ; *(COUNT),*                            5983
000010  020137  000000G                CMP     R1,CCTLR
000014  001003                         BNE     2$
000016  112760  000377  000000G        MOVB    #377,BUFF.OWN(R0)       ; *,*(COUNT)                            5985
000024  005200                 2$:     INC     R0                      ; COUNT                                 5981
000026  020027  000007                 CMP     R0,#7                   ; COUNT,*
000032  003764                         BLE     1$
000034  012601                         MOV     (SP)+,R1                ;                                       5974
000036  000207                         RTS     PC

; Routine Size:  16 words,    Routine Base:  $CODE$ + 6262
; Maximum stack depth per invocation:  2 words
```

```
;    5986  1     global routine OUT_IODQ =
;    5987  1
;    5988  1     !+
;    5989  1     !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;    5990  1     !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;    5991  1     !       QUEUE IS ALSO UPDATED.
;    5992  1     !
;    5993  1     !       INPUTS:
;    5994  1     !               NONE
;    5995  1     !
;    5996  1     !       OUTPUTS:
;    5997  1     !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;    5998  1     !-
;    5999  1
;    6000  2         begin
;    6001  2
;    6002  2         local
;    6003  2             index : word;
;    6004  2
;    6005  2         index = .IODQ[.IODQ_OUT];                   ! GET NEXT RETPKT INDEX
;    6006  2         IODQ_OUT = .IODQ_OUT + 1;                   ! ADVANCE "OUT" POINTER
;    6007  2
;    6008  2         if .IODQ_OUT gequ IODQ_LEN                  ! IF BEYOND END OF QUEUE
;    6009  2         then
;    6010  2             IODQ_OUT = 0;                           ! SET POINTER TO BEGINNING OF QUEUE
;    6011  2
;    6012  2         return .index;                              ! RETURN INDEX TO CALLER
;    6013  1         end;

                                .SBTTL  OUT,IODQ GLOBAL ROUTINES
000000  013700  000000G     OUT.IODQ::
                            MOV     IODQ.OUT,R0             ;                               ;6005
000004  116000  000000G             MOVB    IODQ(R0),R0             ; *,INDEX
000010  042700  177400              BIC     #177400,R0              ; *,INDEX
000014  005237  000000G             INC     IODQ.OUT                ;                       ;6006
000020  023727  000000G 000010      CMP     IODQ.OUT,#10            ;                       ;6008
000026  103404                      BLO     1$
000030  005037  000000G             CLR     IODQ.OUT                ;                       ;6010
000034  000207              1$:     RTS     PC                      ;                       ;5986

; Routine Size:  15 words,    Routine Base:  $CODE$ + 6322
; Maximum stack depth per invocation:  0 words
```

```
;    6014  1      global routine IN_IODQ (index) : novalue =
;    6015  1
;    6016  1      !+
;    6017  1      !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;    6018  1      !       UPDATES THE IODQ_IN POINTER.
;    6019  1      !-
;    6020  1
;    6021  1          if ((.IODQ_IN + 1) eql .IODQ_OUT) or
;    6022  1             (.IODQ_IN - (IODQ_LEN - 1) eql .IODQ_OUT)
;    6023  1          then
;    6024  1              return
;    6025  1          else
;    6026  2              begin
;    6027  2              IODQ [.IODQ_IN] = .index;                  ! LOAD INDEX INTO QUEUE
;    6028  2              IODQ_IN = .IODQ_IN + 1;                    ! ADVANCE "IN" POINTER
;    6029  2
;    6030  2              if .IODQ_IN gequ IODQ_LEN                  ! IF BEYOND END OF QUEUE
;    6031  2              then
;    6032  2                  IODQ_IN = 0;                           ! CYCLE BACK TO BEGINNING OF QUEUE
;    6033  2
;    6034  1              end;                                       ! IF IODQ IS NOT FULL


                                        .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                  IN.IODQ::
000002  013701  000000G                MOV     R1, -(SP)                       ;                       6014
000006  010100                         MOV     IODQ.IN,R1                      ;                       6021
000010  005200                         MOV     R1,R0
000012  020037  000000G                INC     R0
000016  001421                         CMP     R0,IODQ.OUT
000020  010100                         BEQ     1$
000022  162700  000007                 MOV     R1,R0                           ;                       6022
000026  020037  000000G                SUB     #7,R0
000032  001413                         CMP     R0,IODQ.OUT
000034  116661  000004  000000G        BEQ     1$                              ;                       6024
000042  005237  000000G                MOVB    4(SP),IODQ(R1)                  ; INDEX,*               6027
000046  023727  000000G  000010        INC     IODQ.IN                         ;                       6028
000054  103402                         CMP     IODQ.IN,#10                     ;                       6030
000056  005037  000000G                BLO     1$
000062  012601                         CLR     IODQ.IN                         ;                       6033
000064  000207                 1$:     MOV     (SP)+,R1                        ;                       6014
                                        RTS     PC

; Routine Size:  27 words,    Routine Base:  $CODE$ + 6360
; Maximum stack depth per invocation:  2 words
```

```
;    6055  1
;    6056  1
;    6057  1     global routine DROP_CTLR (CTLR, REASON) : novalue =
;    6058  1
;    6059  1
;    6060  1
;    6061  1     !+
;    6062  1     !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;    6063  1     !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;    6064  1     !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;    6065  1     !       SECTION.
;    6066  1     !-
;    6067  1
;    6068  1
;    6069  2         begin
;    6070  2
;    6071  2         local
;    6072  2             UNIT;
;    6073  2
;    6074  2         incr N from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do        ! FOR EACH UNIT
;    6075  2
;    6076  2             if .CST [.CTLR, .N + OF_DATA, D_PRES] eql PRESENT                                       ! IF CONFIGURED
;    6077  3             then
;    6078  3                 begin
;    6079  3                 UNIT = .CST [.CTLR, .N + OF_DATA, D_UNIT];                                          ! DRS UNIT NUMBER
;    6080  3                 DUR [.UNIT] = .REASON;                                                              ! DROP REASON
;    6081  3                 DODU (.UNIT);                                                                       ! DROP UNIT
;    6082  3                 end;
;    6083  2
;    6084  1         end;


                        .SBTTL  DROP.CTLR GLOBAL ROUTINES
000000  004137  000000G       DROP.CTLR::
                        JSR     R1,$SAVE5                       ;                                       0053
000004  016646  000014          MOV     14(SP), -(SP)           ; CTLR,*                                0056
000010  012746  000053          MOV     #53, -(SP)
000014  004737  000000G         JSR     PC,BL$MUL
000020  010003                  MOV     R0,R3
000022  012702  000003          MOV     #3,R2
000026  010300                  MOV     R3,R0                   ; *,N                                   0054
000030  060200          1$:     ADD     R2,R0                   ;                                       0056
000032  006300                  ASL     R0                      ; N,*
000034  032760  040000  000000G BIT     #40000,CST(R0)
000042  001412                  BEQ     2$
000044  016001  000000G         MOV     CST(R0),R1              ; *,UNIT                                0059
000050  000301                  SWAB    R1                      ; UNIT
000052  042701  177760          BIC     #177760,R1              ; *,UNIT
000056  116661  000016  000000G MOVB    16(SP),DUR(R1)          ; REASON,*(UNIT)                        0060
000064  010100                  MOV     R1,R0                   ; UNIT,*                                0061
000066  104451                  TRAP    51
000070  062702  000001          ADD     #1,R2                   ; *,N                                   0054
000074  020227  000041          CMP     R2,#41                  ; N,*
```

```
000100  003752                          BLE     1$                              ;                       6049
000102  022626                          CMP     (SP)+,(SP)+                     ;                       6037
000104  000207                          RTS     PC

; Routine Size:  35 words,    Routine Base:  $CODE$ + 6446
; Maximum stack depth per invocation:  8 words


;   6065  1
;   6066  1
```

```
;    6067  1     global routine DRV_CTLERR (CTLR) : novalue =
;    6068  1
;    6069  1     !+
;    6070  1     !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;    6071  1     !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED.  ITS PURPOSE IS TO
;    6072  1     !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;    6073  1     !       PROGRAM.  THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;    6074  1     !       C-RING COUNT, AND DEALLOCATING MSCP PACKETS DESCRIBED IN THE RESPONSE
;    6075  1     !       RING.
;    6076  1     !
;    6077  1     !       INPUTS:
;    6078  1     !               CTLR - DYING CONTROLLER NUMBER
;    6079  1     !-
;    6080  1
;    6081  2         begin
;    6082  2
;    6083  2         local
;    6084  2             D_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS);  ! CONTROLLER'S DCT ADDRESS
;    6085  2
;    6086  2         D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                       ! GET CONTROLLER'S DCT ADDR
;    6087  2         D_ADDR [WORD0] = OFFLINE;                                   ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;    6088  2         PUTA_PKT (.CTLR);                                           ! RELEASE ALL PACKETS ALLOCATED TO CONTROLLER
;    6089  2         DROP_CTLR (.CTLR, DU_CFATAL);                               ! DROP ALL UNITS ON THE CONTROLLER
;    6090  1         end;                                                        ! ROUTINE DRV_CTLERR

                        .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  010146        DRV.CTLERR::
                        MOV     R1,-(SP)                        ;                       6067
000002  016601  000004          MOV     4(SP),R1                ; CTLR,*                6086
000006  010146                  MOV     R1,-(SP)
000010  012746  000022          MOV     #22,-(SP)
000014  004737  0000000G        JSR     PC,BL$MUL
000020  062700  0000000G        ADD     #DCT,R0
000024  005010                  CLR     (R0)                    ; D.ADDR                6087
000026  010116                  MOV     R1,(SP)                 ;                       6088
000030  004737  005760'          JSR     PC,PUTA.PKT
000034  010116                  MOV     R1,(SP)                 ;                       6089
000036  012746  000006          MOV     #6,-(SP)
000042  004737  006446'          JSR     PC,DROP.CTLR
000046  062706  000006          ADD     #6,SP                   ;                       6081
000052  012601                  MOV     (SP)+,R1                ;                       6090
000054  000207                  RTS     PC

; Routine Size:  23 words,    Routine Base:  $CODE$ + 6554
; Maximum stack depth per invocation:  5 words
```

```
;    6091  1      global routine SEND (index) =
;    6092  1
;    6093  1      !+
;    6094  1      !       IF THE CURRENT RDRX IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;    6095  1      !       ROUTINE "SENDS" A COMMAND TO THE RDRX BY LOADING THE PACKET
;    6096  1      !       DESCRIPTOR OF AN MSCP PACKET INTO THE COMMAND RING AND READING THE
;    6097  1      !       DEVICE'S IP REGISTER.  IF THE
;    6098  1      !       CURRENT RDRX IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;    6099  1      !       THE  CALLER, AND NO ACTION IS TAKEN.
;    6100  1      !
;    6101  1      !       INPUTS:
;    6102  1      !               INDEX - INDEX OF MSCP PACKET CONTAINING THE COMMAND TO
;    6103  1      !                       BE SENT
;    6104  1      !
;    6105  1      !       IMPLICIT INPUTS:
;    6106  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    6107  1      !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    6108  1      !-
;    6109  1
;    6110  2          begin
;    6111  2
;    6112  2          local
;    6113  2              SLOT_ADDR,
;    6114  2              TEMP : word,
;    6115  2              CUR_PRIORITY : word;
;    6117  2          if (.DCT_ADDR [CRING_CNT] lssu CRING_LEN) and                   ! IF CRING IS NOT FULL AND
;    6118  3             ((.DCT_ADDR [STAT] eql ONLINE) or                            !   IF DEVICE IS ONLINE OR
;    6119  3              (.MSCP_PKT [.index, OPCODE] eql OP_SCC))                    !   IT IS A SET-CTRL-CHAR COMMAND
;    6120  2          then
;    6121  2
;    6122  4              if (not ((.MSCP_PKT [.index, OPCODE] eql OP_ACC) or (.MSCP_PKT [.index, OPCODE] eql OP_ONL) or
;    6123  4                       (.MSCP_PKT [.index, OPCODE] eql OP_RD) or (.MSCP_PKT [.index, OPCODE] eql OP_SCC) or
;    6124  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_SDD) OR                  !ZZZ
;    6125  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_RCD) OR                  !ZZZ
;    6126  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_GDS) OR                  !ZZZ
;    6127  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ELP) OR                  !ZZZ
;    6128  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_ABT) OR                  !ZZZ
;    6129  4              (.MSCP_PKT [.INDEX, OPCODE] EQL OP_LSP) OR                  !ZZZ
;    6130  3                       (.MSCP_PKT [.index, OPCODE] eql OP_WRT)))
;    6131  2              then
;    6132  3                  begin
;    6133  3                  PRINTF (OBM107, .MSCP_PKT [.index, OPCODE]);
;    6134  3                  return FAILURE;
;    6135  3                  end
;    6136  2              else
;    6137  3                  begin
;    6138  3
;    6139  3                  do
;    6140  4                      BREAK                                               ! LOOP TILL CREDIT BALANCE POSITIVE
;    6141  4                  until ((.MSCP_PKT [.index, CMD_TYPE] eql IMM_CMD) and
;    6142  3                         (.CREDIT_BAL gequ 1)) or
;    6143  3                        (.CREDIT_BAL gtru 1);
```

```
                MSCP PKT [.index, CRN LO] = (CRN LOW = .CRN LOW + 1);         ! ASSIGN CMD REF NUM

                IF .CRN LOW eql 0
                then
                    CRN HIGH = .CRN HIGH + 1;                                 ! CMD REF NUM (HIGH ORDER)

                MSCP PKT [.index, CRN_HI] = .CRN HIGH;                        !
                SLOT ADDR = .DCT ADDR [CR NEXT];                              ! ADDR OF NEXT COMMAND SLOT

                do                                                            ! WAIT TILL NEXT SLOT HOST OWNED
                    BREAK                                                     !
                until ((.(.SLOT ADDR + 2) and ED OWN) eql 0);                 !

                GETPRI (CUR PRIORITY);                                        ! NO INTERRUPTS WHILE POINTERS UPDATED
                SETPRI (PRIO4);                                               !

                .SLOT ADDR = .MSCP PKT [.index, PKT LO];                      ! LOAD BUFF DESC (LO) INTO COMMAND SLOT
                SLOT ADDR = .SLOT ADDR + 2;                                   ! ADVANCE TO NEXT WORD
                .SLOT ADDR = .MSCP PKT [.index, PKT HI];                      ! LOAD BUFF DESC (HI) INTO COMMAND SLOT
                .SLOT ADDR = ..SLOT ADDR and (not (ED FLAG));                 ! CLEAR INTERRUPT FLAG IN CASE SET
                .SLOT ADDR = ..SLOT ADDR or ED OWN;                           ! GIVE OWNERSHIP TO CONTROLLER
                SLOT ADDR = .SLOT ADDR + 2;                                   ! ADVANCE TO NEXT COMMAND SLOT

                IF .SLOT ADDR gtra .DCT ADDR [CR END]                         ! IF BEYOND END OF CRING
                then
                    SLOT ADDR = .DCT ADDR [CR BEG];                           ! CYCLE BACK TO BEGINNING

                DCT ADDR [CR NEXT] = .SLOT ADDR;                              ! RESTORE CR NEXT POINTER IN DCT
                DCT ADDR [CRING CNT] = .DCT ADDR [CRING CNT] + 1;             ! INCR # OF COMMANDS IN CRING
                IF (.MSCP PKT [.INDEX, CONNID] EQL CID MSCP)                  !IF MSCP COMMAND
                THEN (CREDIT BAL = .CREDIT BAL - 1);                          !DECR CREDIT BALANCE
                TEMP = .RDRX ADDR [RCIP, RC ALL];                             ! READ IP TO FORCE PORT TO POLL
                SETPRI (.CUR PRIORITY);                                       ! LOWER PRIORITY
                return SUCCESS;
                end

        else
            return FAILURE;                                                   ! IF DEVICE IS NOT ONLINE

        end;                                                                  ! ROUTINE SEND

                                                  .SBTTL  SEND GLOBAL ROUTINES
000000  004177  000000              SEND::  JSR     R1,$SAVE4
000004  005746                              TST     -(SP)
000006  122727  000000 00000A               CMPB    @DCT,ADDR,#4            ;                 6...
000014  103100                              BHIS    3$
000016  005727  000000                      TST     @DCT,ADDR               ;                 6...
000022  100415                              BMI     1$
000024  016646  000014                      MOV     14(SP),-(SP)            ; INDEX,*         6114
000030  012746  000106                      MOV     #106,-(SP)
000034  004737  000000                      JSR     PC,BL$MUL
```

```
000040  020626          CMP     (SP)+,(SP)+
000042  126027 000026 000004    CMPB    MSCP,PKT+,22(R0),#4
000050  001167          BNE     10$
000052  016646 000014   1$:     MOV     14(SP), -(SP)           ; INDEX,+
000056  012746 000106          MOV     #106, -(SP)
000062  004737 0000006         JSR     PC,@#$MUL
000066  010002          MOV     R0,R2
000070  020626          CMP     (SP)+,(SP)+
000072  005000          CLR     R0
000074  156200 000026    BISB    MSCP,PKT+,22(R2),R0
000100  020027 000030    CMP     R0,#30
000104  001445          BEQ     5$
000106  020027 000011    CMP     R0,#11
000112  001442          BEQ     5$
000114  020027 000041    CMP     R0,#41          ;
000120  001437          BEQ     5$
000122  020027 000004    CMP     R0,#4
000126  001434          BEQ     5$
000130  020027 000005    CMP     R0,#5           ;
000134  001431          BEQ     5$
000136  020027 000001    CMP     R0,#1           ;
000142  001426          BEQ     5$
000144  020027 000003    CMP     R0,#3           ;
000150  001423          BEQ     5$
000152  020027 000002    CMP     R0,#2           ;
000156  001420          BEQ     5$
000160  020027 000007    CMP     R0,#7           ;
000164  001415          BEQ     5$
000166  020027 000040    CMP     R0,#40          ;
000172  001412          BEQ     5$
000174  010046          MOV     R0, -(SP)       ;
000176  012746 0000003         MOV     #DBM10$, -(SP)
000202  012746 000002          MOV     #2, -(SP)
000206  010600          MOV     SP,R0           ; SP,*
000210  104417          TRAP    17
000212  062706 000006          ADD     #6,SP           ;
000216  000404          BR      10$             ;
000220  104402   2$:    TRAP    2               ;
000222  105762 000046    5$:    TSTB    MSCP,PKT+,40(R2)        ;
000226  001003          BNE     4$              ;
000230  005737 0000000         TST     CREDIT,BAL              ;
000234  001004          BNE     5$              ;
000236  027727 0000000 000001 4$:    CMP     CREDIT,BAL,#1           ;
000244  101765          BLOS    5$              ;
000246  013700 0000000   5$:    MOV     CRN,LOW,R0              ;
000252  005200          INC     R0              ;
000254  010037 0000000         MOV     R0,CRN,LOW              ;
000260  010062 000026          MOV     R0,MSCP,PKT+,22(R2)     ;
000264  001002          BNE     6$              ;
000266  005237 0000000         INC     CRN,HIGH                ;
000272  013762 0000000 000140 6$:    MOV     CRN,HIGH,MSCP,PKT+,140(R2)      ;
000300  013500 0000000         MOV     DCT,ADDR,R0             ;
000304  016001 000020          MOV     20(R0),R1               ; *,SLOT,ADDR
```

```
000510  104420          7$:     TRAP    20                      ;                               6154
000512  032761  100000  000002          BIT     #100000,2(R1)   ; *,*(SLOT.ADDR)               6156
000520  001373                  BNE     7$                      ;                               6157
000522  104440                  TRAP    40                      ;                               6158
000524  010005                  MOV     R0,R5                   ; *,CUR.PRIORITY                6159
000526  012700  000200          MOV     #200,R0                 ;                               6160
000532  104441                  TRAP    41                      ;                               6161
000534  016201  000006          MOV     MSCP.PKT(R2),(R1)+      ; *,SLOT.ADDR                   6162
000540  016211  000026          MOV     MSCP.PKT+2(R2),(R1)     ; *,SLOT.ADDR                   6163
000544  042711  040000          BIC     #40000,(R1)             ; *,SLOT.ADDR                   6164
000550  052711  100000          BIS     #100000,(R1)+           ; *,SLOT.ADDR                   6165
000554  013700  000006          MOV     DCT.ADDR,R0             ;                               6166
000560  020160  000012          CMP     R1,12(R0)               ; SLOT.ADDR,*                   6167
000564  101403                  BLOS    8$                      ;                               6168
000566  016001  000010          MOV     10(R0),R1               ; *,SLOT.ADDR                   6169
000572  010160  000020  8$:     MOV     R1,20(R0)               ; SLOT.ADDR,*                   6170
000576  105210                  INCB    (R0)                    ;                               6171
000600  105762  000116          TSTB    MSCP.PKT+11(R2)         ;                               6172
000604  001002                  BNE     9$                      ;                               6173
000606  005537  000006          DEC     CREDIT.BAD              ;                               6174
000612  017716  000006  9$:     MOV     @RDRX.ADDR,(SP)         ; *,RC.REG                      6175
000616  010500                  MOV     R5,R0                   ; CUR.PRIORITY,*                6176
000620  104441                  TRAP    41                      ;                               6177
000622  012700  000001          MOV     #1,R0                   ;                               6178
000626  000401                  BR      11$                     ;                               6179
000630  005000          10$:    CLR     R0                      ;                               6180
000632  005726          11$:    TST     (SP)+                   ;                               6091
000634  000207                  RTS     PC
```

; Routine Size: 145 words, Routine Base: $CODE$ + 6630
; Maximum stack depth per invocation: 10 words

```
;    6185  1      global routine WAIT : novalue =
;    6186  1
;    6187  1      !+
;    6188  1      !       THE PURPOSE OF THIS ROUTINE IS TO KILL TIME UNTIL AN RDRX INTERRUPT
;    6189  1      !       RESULTS IN A RETURN PACKET INDEX BEING DEPOSITED INTO THE I/O DONE
;    6190  1      !       QUEUE (IODQ).
;    6191  1      !-
;    6192  1
;    6193  1          do
;    6194  1              BREAK                                   ! BREAK FOR ACT
;    6195  1          until .IODQ_IN neq .IODQ_OUT;


                                .SBTTL  WAIT GLOBAL ROUTINES
00000000  1044,00               WAIT::
00000000                        1$:     TRAP    0                       ;                       6194
00000002  02737  0000006G 0000006G      CMP     IODQ_IN,IODQ_OUT        ;                       6195
00000010  001375                        BEQ     1$
00000012  000207                        RTS     PC                      ;                       6185

; Routine Size:  6 words,    Routine Base:  $CODE$ + 7270
; Maximum stack depth per invocation:  2 words

;    6196  1
```

```
;    6197  1
;    6198  1      GLOBAL ROUTINE MODULAS (LO_LIMIT, HI_LIMIT) =                          !777
;    6199  1                                                                            !777
;    6200  1      !+ THE PURPOSE OF THIS ROUTINE IS TO GET A RANDOM NUMBER BETWEEN      !777
;    6201  1      !  THE LOW AND HIGH LIMITS. THIS SHOULD WORK FOR A 16 BIT WORD.       !777
;    6202  1      !  THE "MOD" FUNC ONLY WORKS ON 15 BITS.                              !777
;    6203  1                                                                            !777
;    6204  2      BEGIN                                                                 !777
;    6205  2      OWN      X : WORD;                  !VARIABLE FOR RANDOM WD TABLE     !777
;    6206  2      LOCAL    ANSWER : UNSIGNED WORD,    !FINAL ANSWER                     !777
;    6207  2               SAVESZ : UNSIGNED WORD,    !SAVES SIZE OF WINDOW             !777
;    6208  2               SIZE : UNSIGNED WORD;      !SIZE OF WINDOW                   !777
;    6209  2                                                                           !777
;    6210  2                                                                           !777
;    6211  2      X = .X + 1;                                                          !777
;    6212  2      IF .X GEQ RDM_LEN                                                    !777
;    6213  2      THEN X = 0;                        !KEEP ROTATING RANDOM NUMBERS USED !777
;    6214  2
                          !777
;    6215  2      SIZE = .HI_LIMIT - .LO_LIMIT;                                        !777
;    6216  2      SAVESZ = .HI_LIMIT - .LO_LIMIT;                                      !777
;    6217  3      IF (.SIZE LEQU %O'077777')         !IF BIT 15 NOT SET                !777
;    6218  3      THEN ANSWER = ((.RANDOM[.X] AND %O'077777') MOD (.SIZE + 1))         !777
;    6219  3                                         !ONLY 15 BIT WD, SO TAKE RANDOM SAMPLE !777
;    6220  2      ELSE                               !16 BIT WD                        !777
;    6221  3              BEGIN                                                        !777
;    6222  3              SIZE = .SIZE / 2;          !MAKES SIZE A 15 BIT LENGTH, OR DIV BY 2!777
;    6223  3              ANSWER = (.RANDOM[.X] AND %O'077777') MOD (.SIZE + 1);       !777
;    6224  3                                         !GIVES 15 BIT RANDOM NUMBER       !777
;    6225  3              ANSWER = .ANSWER * 2;      !BUILD UP TO REGULAR SIZE         !777
;    6226  3              ANSWER = .ANSWER + (.RANDOM[.X + 1] AND 1);                  !777
;    6227  3                                         !RANDOM'S FIRST BIT 0             !777
;    6228  4              IF (.ANSWER GTRU SAVESZ) !ITS POSSIBLE TO BE 1 LARGER THAN SIZE !777
;    6229  3              THEN ANSWER = .SAVESZ;     !SO CHECK.                        !777
;    6230  2              END;                                                         !777
;    6231  2      RETURN .ANSWER;
;    6232  1      END;                               !END MODULAS ROUTINE              !777


007304                                  X:      .BLKW   1


                                        .SBTTL  MODULAS: GLOBAL ROUTINES
000000  004137  000000G                MODULAS::
                                        JSR     R1,$SAVE2
000004  005746                          TST     -(SP)
000006  005237  007304                  INC     X
000012  023727  007304  000020          CMP     X,#20
000020  003402                          BLT     1$
000022  005037  007304                  CLR     X
000026  016600  000012          1$:     MOV     12(SP),R0       ; R0:=LIMIT1,*
000032  166600  000014                  SUB     14(SP),R0       ; R0:=LIMIT2-*
000036  010001                          MOV     R0,R1           ; *,SIZE
000040  010016                          MOV     R0,(SP)         ; *,SAVESZ
```

```
000042  013700  007304'          MOV     X,R0                    ;                       6218
000046  006300                   ASL     R0
000050  020127  077777           CMP     R1,#77777               ; SIZE,*                6217
000054  101011                   BHI     2$
000056  016046  000000G          MOV     RANDOM(R0),-(SP)        ;                       6218
000062  042716  100000           BIC     #100000,(SP)
000066  010146                   MOV     R1,-(SP)                ; SIZE,*
000070  005216                   INC     (SP)
000072  004737  000000G          JSR     PC,BL$MOD
000076  000451                   BR      3$                      ;                       6221
000100  006201                2$:ASR     R1                      ; SIZE                  6222
000102  016046  000000G          MOV     RANDOM(R0),-(SP)        ;                       6223
000106  042716  100000           BIC     #100000,(SP)
000112  010146                   MOV     R1,-(SP)                ; SIZE,*
000114  005216                   INC     (SP)
000116  004737  000000G          JSR     PC,BL$MOD
000122  006300                   ASL     R0                      ; ANSWER                6225
000124  013701  007304'          MOV     X,R1                    ;                       6226
000130  006301                   ASL     R1
000132  116102  000000G          MOVB    RANDOM+1(R1),R2
000136  042702  177776           BIC     #177776,R2
000142  060200                   ADD     R2,R0                   ; *,ANSWER
000144  012701  000004           MOV     #4,R1                   ;                       6228
000150  060601                   ADD     SP,R1                   ; SAVES2,*
000152  020001                   CMP     R0,R1                   ; ANSWER,*
000154  101402                   BLOS    3$
000156  016600  000004           MOV     4(SP),R0                ; SAVES2,ANSWER         6232
000162  062706  000006        3$:ADD     #6,SP                   ;                       6198
000166  000207                   RTS     PC

; Routine Size:  60 words,    Routine Base:  $CODE$ + 7306
; Maximum stack depth per invocation:  7 words
```

```
;     6233  1     %sbttl 'ERROR MESSAGE SUBROUTINES'
;     6234  1
;     6235  1     routine EMS_SA : novalue =
;     6236  1
;     6237  1     !+
;     6238  1     !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;     6239  1     !       THE CONTENTS OF THE SA REGISTER.
;     6240  1     !
;     6241  1
;     6242  2         begin
;     6243  2
;     6244  2         if .SA_REG eql %o'177777'                                   ! IF CONTROLLER TIME-OUT
;     6245  2         then
;     6246  3             begin
;     6247  3             PRINTX (CRLF);
;     6248  3             PRINTX (ASTERISK);
;     6249  3             PRINTX (.CNTR_ERR [0]);
;     6250  3             end
;     6251  2         else
;     6252  2
;     6253  2             if (.SA_REG and %o'003777') lequ 22                     ! IF GENERIC CONTROLLER ERROR
;     6254  2             then
;     6255  3                 begin
;     6256  3                 PRINTX (CRLF);
;     6257  3                 PRINTX (ASTERISK);
;     6258  3                 PRINTX (.CNTR_ERR [.SA_REG and %o'003777']);
;     6259  3                 end
;     6260  2             else
;     6261  2
;     6262  2                 if ((.SA_REG and %o'003777') - 400) lequ 6          ! IF RDRX SPECIFIC CONTROLLER ERROR
;     6263  2                 then
;     6264  3                     begin
;     6265  3                     PRINTX (CRLF);
;     6266  3                     PRINTX (ASTERISK);
;     6267  3                     PRINTX (.RDRX_ERR [(.SA_REG and %o'003777') - 400]);
;     6268  3                     end
;     6269  2                 else
;     6270  2                     PRINTX (EX_SA, .SA_REG);                       ! JUST PRINT CONTENTS OF SA
;     6271  2
;     6272  2         EMS_TIM ();                                                ! TIME
;     6273  1         end;


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

```
000054  012746  000001          MOV     #1, -(SP)
000060  010600                  MOV     SP,R0                   ; SP,*
000062  104415                  TRAP    15
000064  013716  000000G         MOV     CNTR.ERR,(SP)           ;                       6249
000070  012746  000001          MOV     #1, -(SP)
000074  010600                  MOV     SP,R0                   ; SP,*
000076  104415                  TRAP    15
000100  000425                  BR      3$                      ;                       6246
000102  010100          1$:     MOV     R1,R0                   ;                       6253
000104  042700  174000          BIC     #^C<17400>,R0
000110  020027  000026          CMP     R0,#26
000114  101030                  BHI     2$
000116  012746  000000G         MOV     #CRLF, -(SP)            ;                       6256
000122  012746  000001          MOV     #1, -(SP)
000126  010600                  MOV     SP,R0                   ; SP,*
000130  104415                  TRAP    15
000132  012716  000000G         MOV     #ASTERISK,(SP)          ;                       6257
000136  012746  000001          MOV     #1, -(SP)
000142  010600                  MOV     SP,R0                   ; SP,*
000144  104415                  TRAP    15
000146  013700  000000G         MOV     SA.REG,R0               ;                       6258
000152  042700  174000          BIC     #^C<17400>,R0
000156  006300                  ASL     R0
000160  016016  000000G         MOV     CNTR.ERR(R0),(SP)
000164  012746  000001          MOV     #1, -(SP)
000170  010600                  MOV     SP,R0                   ; SP,*
000172  104415                  TRAP    15
000174  000437                  BR      5$                      ;                       6255
000176  010100          2$:     MOV     R1,R0                   ;                       6260
000200  042700  174000          BIC     #^C<17400>,R0
000204  162700  000020          SUB     #20,R0
000210  020027  000006          CMP     R0,#6
000214  101051                  BHI     4$
000216  012746  000000G         MOV     #CRLF, -(SP)            ;                       6265
000222  012746  000001          MOV     #1, -(SP)
000226  010600                  MOV     SP,R0                   ; SP,*
000230  104415                  TRAP    15
000232  012716  000000G         MOV     #ASTERISK,(SP)          ;                       6266
000236  012746  000001          MOV     #1, -(SP)
000242  010600                  MOV     SP,R0                   ; SP,*
000244  104415                  TRAP    15
000246  013700  000000G         MOV     SA.REG,R0               ;                       6267
000252  042700  174000          BIC     #^C<17400>,R0
000256  006300                  ASL     R0
000260  016016  176340G         MOV     RDRX.ERR 1440(R0),(SP)
000264  012746  000001          MOV     #1, -(SP)
000270  010600                  MOV     SP,R0                   ; SP,*
000272  104415                  TRAP    15
000274  005726          3$:     TST     (SP)+                   ;                       6269
000276  000407                  BR      5$                      ;                       6270
000300  010146          4$:     MOV     R1, -(SP)               ;                       6270
000302  012746  000000G         MOV     #LEX.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      ;                       6233
000302  062706  000006                  ADD     #6,SP           ;                       6234
000306  012601                          MOV     (SP)+,R1        ;                       6235
000310  000207                          RTS     PC

; Routine Size:  101 words,     Routine Base:  $CODE$ + 7476
; Maximum stack depth per invocation:  7 words
```

```
;    6274  1      routine EMS_SBC : novalue =
;    6275  1
;    6276  1      !+
;    6277  1      !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SB_CODE" (SUB CODE) IF
;    6278  1      !       EITHER THE STATUS CODE (ST CODE) OR THE SUB-CODE IS NON ZERO. (A
;    6279  1      !       NON-ZERO SUB CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
;    6280  1      !       HAS MEANING WITH A NON-ZERO STATUS CODE).
;    6281  1      !-
;    6282  1
;    6283  2          begin
;    6284  2
;    6285  2          if (.ST_CODE or .SB_CODE) neq 0                                 ! PRINT SUB CODE ONLY ON ERROR
;    6286  2          then
;    6287  3              begin
;    6288  3              PRINTX (EX_SB);                                             ! SUB-CODE :
;    6289  3
;    6290  3              case .ST_CODE from ST_SUC to ST_DRV of
;    6291  3                  set
;    6292  3
;    6293  3                  [ST_SUC]:           if .SB_CODE lequ 16                 ! SUCCESS SUB-CODES
;    6294  3                                      then
;    6295  3                                          PRINTX (.TBL_SUC [.SB_CODE]);
;    6296  3
;    6297  3                  [ST_CMD]:           PRINTX (EX_SBO, .SB_CODE / 8);      ! INVALID COMMAND
;    6298  3
;    6299  3                  [ST_ABO]:           ;                                   ! COMMAND ABORTED
;    6300  3
;    6301  3                  [ST_OFL]:           if .SB_CODE lequ 8                  ! UNIT OFFLINE
;    6302  3                                      then
;    6303  3                                          PRINTX (.TBL_OFL [.SB_CODE]);
;    6304  3
;    6305  3                  [ST_AVL]:           ;                                   ! UNIT AVAILABLE
;    6306  3
;    6307  3                  [ST_MFE]:           if .SB_CODE lequ 10                 ! MEDIA FORMAT ERROR
;    6308  3                                      then
;    6309  3                                          PRINTX (.TBL_MFE [.SB_CODE]);
;    6310  3
;    6311  3                  [ST_WPT]:           if (.SB_CODE / 128) lequ 2          ! WRITE PROTECTED
;    6312  3                                      then
;    6313  3                                          PRINTX (.TBL_WPT [(.SB_CODE / 128)]);
;    6314  3
;    6315  3                  [ST_CMP]:           ;                                   ! COMPARE ERROR
;    6316  3
;    6317  3                  [ST_DAT]:           if .SB_CODE lequ 15                 ! DATA ERROR
;    6318  3                                      then
;    6319  3                                          PRINTX (.TBL_DAT [.SB_CODE]);
;    6320  3
;    6321  3                  [ST_HST]:           if .SB_CODE lequ 4                  ! HOST ACCESS ERROR
;    6322  3                                      then
;    6323  3                                          PRINTX (.TBL_HST [.SB_CODE]);
;    6324  3
;    6325  3                  [ST_CNT]:           if .SB_CODE lequ 4                  ! CONTROLLER ERROR
;    6326  3                                      then
```

```
;    6327  3                                  PRINTX (.TBL_CNT [.SB_CODE]);
;    6328  3
;    6329  3          [ST_DRV]:               if .SB_CODE lequ 8                      ! DRIVE ERROR
;    6330  3                                  then
;    6331  3                                      PRINTX (.TBL_DRV [.SB_CODE]);
;    6332  3
;    6333  3          [outrange]:             PRINTX (EX_SBO, .SB_CODE);              ! JUST PRINT SUB-CODE IF NO MATCH
;    6334  3          tes;
;    6335  3
;    6336  2              end;
;    6337  2
;    6338  1          end;


                                    .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000  013700  000000G      EMS.SBC:MOV     ST.CODE,R0                      ;                       6285
000004  053700  000000G              BIS     SB.CODE,R0
000010  001001                       BNE     1$
000012  000207                       RTS     PC
000014  012746  000000G      1$:     MOV     #EX.SB,-(SP)                    ;                       6288
000020  012746  000001              MOV     #1,(SP)
000024  010600                       MOV     SP,R0                           ; SP,*
000026  104415                       TRAP    15
000030  013700  000000G              MOV     ST.CODE,R0                      ;                       6290
000034  020027  000013              CMP     R0,#13
000040  101003                       BHI     3$
000042  006300                       ASL     R0
000044  066007  000000'              ADD     P.AAA(R0),PC                    ; Case dispatch
000050  013716  000000G      3$:     MOV     SB.CODE,(SP)                    ;                       6333
000054  012746  000000G              MOV     #EX.SBO,-(SP)
000060  012746  000002              MOV     #2,(SP)
000064  010600                       MOV     SP,R0                           ; SP,*
000066  104415                       TRAP    15
000070  022626                       CMP     (SP)+,(SP)+
000072  000435                       BR      6$
000074  023727  000000G 000020 4$:   CMP     SB.CODE,#20                     ;                       6292
000102  101165                       BHI     14$                             ;                       6293
000104  013700  000000G              MOV     SB.CODE,R0                      ;                       6295
000110  006300                       ASL     R0
000112  016016  000000'              MOV     TBL.SUC(R0),(SP)
000116  012746  000001              MOV     #1,(SP)
000122  010600                       MOV     SP,R0                           ; SP,*
000124  104415                       TRAP    15
000126  000565                       BR      15$
000130  013716  000000G      5$:     MOV     SB.CODE,(SP)                    ;                       6297
000134  012746  000010              MOV     #10,(SP)
000140  004737  000000G              JSR     PC,@#$DTV
000144  010016                       MOV     R0,(SP)
000146  012746  000000G              MOV     #EX.SBO,-(SP)
000152  012746  000002              MOV     #2,(SP)
000156  010600                       MOV     SP,R0                           ; SP,*
000160  104415                       TRAP    15
000162  062706  000006              ADD     #6,SP
```

```
000166  000546                          6$:     BR      16$                             ;                       6290
000170  023727  000000G 000010          7$:     CMP     SB.CODE,#10                     ;                       6301
000176  101142                                  BHI     16$
000200  013700  000000G                         MOV     SB.CODE,R0                      ;                       6303
000204  006300                                  ASL     R0
000206  016016  000042'                         MOV     TBL.OFL(R0),(SP)
000212  012746  000001                          MOV     #1,-(SP)
000216  010600                                  MOV     SP,R0                           ; SP,*
000220  104415                                  TRAP    15
000222  000527                                  BR      15$
000224  023727  000000G 000012          8$:     CMP     SB.CODE,#12                     ;                       6307
000232  101124                                  BHI     16$
000234  013700  000000G                         MOV     SB.CODE,R0                      ;                       6309
000240  006300                                  ASL     R0
000242  016016  000064'                         MOV     TBL.MFE(R0),(SP)
000246  012746  000001                          MOV     #1,-(SP)
000252  010600                                  MOV     SP,R0                           ; SP,*
000254  104415                                  TRAP    15
000256  000511                                  BR      15$
000260  013716  000000G                 9$:     MOV     SB.CODE,(SP)                    ;                       6311
000264  012746  000200                          MOV     #200,(SP)
000270  004737  000000G                         JSR     PC,BL$DIV
000274  005726                                  TST     (SP)+
000276  020027  000002                          CMP     R0,#2
000302  101100                                  BHI     16$
000304  006300                                  ASL     R0                              ;                       6313
000306  016016  000112'                         MOV     TBL.WPT(R0),(SP)
000312  012746  000001                          MOV     #1,-(SP)
000316  010600                                  MOV     SP,R0                           ; SP,
000320  104415                                  TRAP    15
000322  000467                                  BR      15$
000324  023727  000000G 000017          10$:    CMP     SB.CODE,#17                     ;                       6317
000332  101064                                  BHI     16$
000334  013700  000000G                         MOV     SB.CODE,R0                      ;                       6319
000340  006300                                  ASL     R0
000342  016016  000120'                         MOV     TBL.DAT(R0),(SP)
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                           ; SP,*
000354  104415                                  TRAP    15
000356  000451                                  BR      15$
000360  023727  000000G 000004          11$:    CMP     SB.CODE,#4                      ;                       6321
000366  101046                                  BHI     16$
000370  013700  000000G                         MOV     SB.CODE,R0                      ;                       6323
000374  006300                                  ASL     R0
000376  016016  000160'                         MOV     TBL.WST(R0),(SP)
000402  012746  000001                          MOV     #1,-(SP)
000406  010600                                  MOV     SP,R0                           ; SP,*
000410  104415                                  TRAP    15
000412  000433                                  BR      15$
000414  023727  000000G 000003          12$:    CMP     SB.CODE,#3                      ;                       6325
000422  101030                                  BHI     16$
000424  013700  000000G                         MOV     SB.CODE,R0                      ;                       6327
000430  006300                                  ASL     R0
```

```
000432  016016  000172'                        MOV     TBL.CNT(R0),(SP)
000436  012746  000001                        MOV     #1,-(SP)
000442  010600                                MOV     SP,R0                   ; SP,*
000444  104415                                TRAP    15
000446  000415                                BR      15$
000450  023727  000000G 000010        13$:    CMP     SB.CODE,#10             ;                       6329
000456  101012                        14$:    BHI     16$
000460  013700  000000G                        MOV     SB.CODE,R0              ;                       6331
000464  006300                                ASL     R0
000466  016016  000000'                        MOV     TBL.DRV(R0),(SP)
000472  012746  000001                        MOV     #1,-(SP)
000476  010600                                MOV     SP,R0                   ; SP,*
000500  104415                                TRAP    15
000502  005726                        15$:    TST     (SP)+
000504  022626                        16$:    CMP     (SP)+,(SP)+             ;                       6282
000506  000207                                RTS     PC                      ;                       6274

; Routine Size:  164 words,    Routine Base:  $CODE$ + 10010
; Maximum stack depth per invocation:  7 words


000000                                          .PSECT  $PLIT$,  RO ,  D

                                P.AAA:
000000  000024                 2$:             .WORD   24                      ; CASE Table for EMS.SBC+0044   6290
000002  000060                                 .WORD   60                      ; [4$]
000004  000434                                 .WORD   434                     ; [5$]
000006  000120                                 .WORD   120                     ; [16$]
000010  000434                                 .WORD   434                     ; [7$]
000012  000154                                 .WORD   154                     ; [16$]
000014  000210                                 .WORD   210                     ; [8$]
000016  000434                                 .WORD   434                     ; [9$]
000020  000254                                 .WORD   254                     ; [16$]
000022  000310                                 .WORD   310                     ; [10$]
000024  000344                                 .WORD   344                     ; [11$]
000026  000400                                 .WORD   400                     ; [12$]
                                                                                ; [13$]
```

```
routine EMS_CMD : novalue =

!+
!       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
!       PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
!       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
!
!       IMPLICIT INPUTS:
!               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
!-

    begin
    PRINTX (EX_CMD);                                        ! "COMMAND: "

    selectoneu (.RP_ADDR [ENDCOD] and OP_MSK) of
        set

        [OP_ONL]:       PRINTX (EX_ONL);                    ! ONLINE

        [OP_ACC]:       PRINTX (EX_ACC);                    ! ACCESS

        [OP_RD]:        begin
                        PRINTX (EX_RD);                     ! READ

                        if .RP_ADDR [CMDMOD] neq 0
                        then
                            PRINTX (EX_CMP);                ! COMPARE

                        end;

        [OP_WRT]:       begin
                        PRINTX (EX_WRT);                    ! WRITE

                        if .RP_ADDR [CMDMOD] neq 0
                        then
                            PRINTX (EX_CMP);                ! COMPARE

                        end;

        [otherwise]:    PRINTX (EX_OP, .RP_ADDR [ENDCOD]);  ! ENDCODE VALUE IF NO MATCH
        tes;

    end;                                                    ! ROUTINE EMS_CMD
```

```
010450                          .SBTTL  EMS_CMD ERROR MESSAGE SUBROUTINES
                                .PSECT  $CODE$, RO

000000  004157  000000G EMS_CMD:JSR     R1,$SAVE2                       ; 6551
000004  012746  000000G         MOV     #EX_CMD,-(SP)           ;       ; 6551
000010  012746  000001          MOV     #1,-(SP)
000014  010600                  MOV     SP,R0                   ; SP,*
000016  104415                  TRAP    15
```

```
000000  013700  000000G        MOV     RP,ADDR,R0              ;                       6353
000004  116001  000013          MOVB    14(R0),R1
000010  042701  177600          BIC     #177600,R1
000014  020127  000011          CMP     R1,#11                  ;                       6356
000020  001007                  BNE     1$
000022  013746  000000G         MOV     #EX.ONL,-(SP)
000026  012746  000001          MOV     #1,-(SP)
000032  010600                  MOV     SP,R0                   ; SP,*
000034  104415                  TRAP    15
000036  000464                  BR      5$
000040  020127  000000      1$: CMP     R1,#0                   ;                       6358
000044  001007                  BNE     2$
000046  013746  000000G         MOV     #EX.ACC,-(SP)
000052  012746  000001          MOV     #1,-(SP)
000056  010600                  MOV     SP,R0                   ; SP,*
000060  104415                  TRAP    15
000062  000445                  BR      5$
000064  020127  000041      2$: CMP     R1,#41                  ;                       6360
000070  001023                  BNE     3$
000072  013746  000000G         MOV     #EX.RD,-(SP)            ;                       6361
000076  012746  000001          MOV     #1,-(SP)
000102  010600                  MOV     SP,R0                   ; SP,*
000104  104415                  TRAP    15
000106  013700  000000G         MOV     RP,ADDR,R0              ;                       6363
000112  005760  000012          TST     12.(R0)
000116  001434                  BEQ     5$
000120  013746  000000G         MOV     #EX.CMP,-(SP)           ;                       6365
000124  012746  000001          MOV     #1,-(SP)
000130  010600                  MOV     SP,R0                   ; SP,*
000132  104415                  TRAP    15
000134  000424                  BR      4$
000136  020127  000042      3$: CMP     R1,#42                  ;                       6366
000142  001024                  BNE     6$
000144  013746  000000G         MOV     #EX.WRT,-(SP)           ;                       6370
000150  012746  000001          MOV     #1,-(SP)
000154  010600                  MOV     SP,R0                   ; SP,*
000156  104415                  TRAP    15
000160  013700  000000G         MOV     RP,ADDR,R0              ;                       6371
000164  005760  000012          TST     12.(R0)
000170  001407                  BEQ     5$
000172  013746  000000G         MOV     #EX.CMP,-(SP)           ;                       6374
000176  012746  000001          MOV     #1,-(SP)
000202  010600                  MOV     SP,R0                   ; SP,*
000204  104415                  TRAP    15
000206  005726                  TST     (SP)+           ;
000210  005726            4$:   TST     (SP)+           ;
000212  000413                  BR      7$              ;                       6376
000214  005726            5$:   TST     (SP)+           ;                       6378
000216  116001  000014          MOVB    14(R0),-(SP)
000222  013746  000000G         MOV     #EX.CMP,-(SP)
000226  012746  000002          MOV     #2,-(SP)
000232  010600                  MOV     SP,R0
000234  104415                  TRAP    15              ; SP,*
```

```
000000  0C0006                          CMP     (SP)+,(SP)+
000002  0C0006                  7$:     CMP     (SP)+,(SP)+                     ;                       6350
000004  000200                          RTS     PC                              ;                       6339

; Routine Size:  90 words,    Routine Base:  $CODE$ + 10520
; Maximum stack depth per invocation:  9 words
```

```
;    6382  1      GLOBAL ROUTINE EMS_DBN : NOVALUE =                                   !ZZZ
;    6383  1      !+                                                                  !ZZZ
;    6384  1      !       THIS ROUTINE PRINTS THE PRESENT DBN                         !ZZZ
;    6385  1      !                                                                   !ZZZ
;    6386  1      !       IMPLICIT INPUTS:                                            !ZZZ
;    6387  1      !               CST_ADDR - ADDRESS OF CONTROLLER STATUS TABLE       !ZZZ
;    6388  1      !                                                                   !ZZZ
;    6389  1                                                                          !ZZZ
;    6390  2      BEGIN                                                               !ZZZ
;    6391  2      PRINTB (%X15, .CDISK);                  !"DISK %X"                  !ZZZ
; P  6392  2      PRINTB (%X23, .CST_ADDR [.COUNT * 8 + DBN, 0,8], .CST_ADDR          !ZZZ
;    6393  2                             [.COUNT * 8 + DBN, 8,8]);    !"DBN: %XXXXXX,"  !ZZZ
;    6394  2      PRINTB (%X52, .S_DUPPKT - .P);          !PRINT BYTE COUNT           !ZZZ
;    6395  2      PRINTB (%X55, .P_PATTERN);              !PRINT THE PATTERN          !ZZZ
;    6396  2      PRINTB (%X54, .(DUPPKT + .S_DUPPKT), .(DUPPKT + .S_DUPPKT));  !PRINT THE WORD READ  !ZZZ
;    6397  2      EMS_BLK (DUPPKT+2, 256);                !PRINT WHOLE BLOCK READ     !ZZZ
;    6398  1      END;                                    !IN OCTAL                   !ZZZ


                                .SBTTL  EMS.DBN ERROR MESSAGE SUBROUTINES
000000  013746  000000G    EMS.DBN::
                            MOV     CDISK,-(SP)                     ;               ;6391
000004  012746  000000G            MOV     #X15,-(SP)
000010  012746  000000'            MOV     #2,-(SP)
000014  010600                     MOV     SP,R0                   ; SP,*
000016  104414                     TRAP    14
000020  013700  000000G            MOV     CODE,R0                 ;               ;6393
000024  006300                     ASL     R0
000026  063700  000000G            ADD     CST_ADDR,R0
000032  005046                     CLR     -(SP)
000034  116016  000000            MOVB    X0(R0),-(SP)
000040  005046                     CLR     -(SP)
000042  116016  000000            MOVB    X8(R0),-(SP)
000046  012746  000000G            MOV     #X23,-(SP)
000052  012746  000005            MOV     #5,-(SP)
000056  010600                     MOV     SP,R0                   ; SP,*
000060  104414                     TRAP    14
000062  013716  000000G            MOV     S_DUPPKT,-(SP)          ;               ;6394
000066  162716  000000'            SUB     #2,(SP)
000072  012746  000000G            MOV     #X52,-(SP)
000076  012746  000000'            MOV     #2,-(SP)
000102  010600                     MOV     SP,R0                   ; SP,*
000104  104414                     TRAP    14
000106  013716  000000G            MOV     P_PATTERN,-(SP)         ;               ;6395
000112  012746  000000G            MOV     #X55,-(SP)
000116  012746  000000'            MOV     #2,-(SP)
000122  010600                     MOV     SP,R0                   ; SP,*
000124  104414                     TRAP    14
000126  013700  000000G            MOV     S_DUPPKT,R0             ;               ;6396
000132  016046  000000G            MOV     DUPPKT(R0),-(SP)
000136  011646                     MOV     (SP),-(SP)
000140  012746  000000G            MOV     #X54,-(SP)
000144  012746  000005            MOV     #5,-(SP)
```

```
000150  010600                  MOV     SP,R0                   ; SP,*
000152  104414                  TRAP    14
000154  012716  000000'G        MOV     #DUPPKT+2,(SP)          ;                       6397
000160  012746  000400           MOV     #400, -(SP)
000164  004737  000000V          JSR     PC,EMS$BLK
000170  062706  000034           ADD     #34,SP                  ;                       6390
000174  000207                  RTS     PC                      ;                       6382

; Routine Size:  62 words,    Routine Base:  $CODE$ + 11004
; Maximum stack depth per invocation:  15 words


;  .ASSU  :
;  6400  1
```

```
;    6401  1
;    6402  1      GLOBAL ROUTINE EMS_BLK (ADDR, LENGTH) : NOVALUE =                                    !///
;    6403  1                                                                                          !///
;    6404  1      !+                                                                                   !///
;    6405  1      !       THIS ROUTINE WILL PRINTX A BLOCK OF MEMORY, WHICH IS 'LENGTH'                !///
;    6406  1      !       WORDS LONG STARTING AT ADDRESS 'ADDR'. PRINTING IS DONE IN OCTAL.            !///
;    6407  1      !       8 WDS TO A LINE.                                                             !///
;    6408  1      !                                                                                    !///
;    6409  1                                                                                          !///
;    6410  2      BEGIN                                                                               !///
;    6411  2      LITERAL                                                                             !///
;    6412  2              MASK = %o'7';                                                               !///
;    6413  2                                                                                          !///
;    6414  2      PRINTX (CRLF);                                                                      !///
;    6415  2              INCR COUNT FROM 1 TO .LENGTH DO          !FOR EACH WD TO PRINT              !///
;    6416  3                      BEGIN                                                               !///
;    6417  3                      IF ((.COUNT - 1) AND MASK) EQL 0 !IF START OF NEW LINE              !///
;    6418  3                      THEN                                                                !///
;    6419  3                              PRINTX (SPACE4);         !PRINT 4 BLANKS                    !///
;    6420  3                                                                                          !///
;    6421  3                      PRINTX (EX_WRD, .ADDR);          !PRINTX A WORD                     !///
;    6422  3                      ADDR = .ADDR +2;                 !TO NEXT ADDRESS                   !///
;    6423  3                                                                                          !///
;    6424  4                      IF ((.COUNT AND MASK) EQL 0) OR  !END OF LINE OR                    !///
;    6425  4                         (.COUNT EQL .LENGTH))         !WHEN DONE                         !///
;    6426  5                      THEN                                                                !///
;    6427  5                              PRINTX (CRLF);           !PRINT CR LF                       !///
;    6428  4                      END;                                                                !///
;    6429  1      END;


                        .SBTTL  EMS,BLK ERROR MESSAGE SUBROUTINES
000000  010146      EMS,BLK::
                        MOV     R1, -(SP)                       ;                               6400
000002  012746  000006  MOV     #CRLF, -(SP)                    ;                               6414
000006  012746  000001  MOV     #1, -(SP)
000012  010600          MOV     SP,R0                           ; SP,*
000014  104415          TRAP    15
000016  005001          CLR     R1                              ; COUNT                         6415
000020  000445          BR      5$
000022  010100      1$: MOV     R1,R0                           ; COUNT,*                       6417
000024  005300          DEC     R0
000026  032700  000007  BIT     #7,R0
000032  001007          BNE     2$
000034  012716  000006  MOV     #SPACE4,(SP)                    ;                               6419
000040  012746  000001  MOV     #1, -(SP)
000044  010600          MOV     SP,R0                           ; SP,*
000046  104415          TRAP    15
000050  005726          TST     (SP)+
000052  012601      2$: MOV     @1*2(SP),-(SP)                  ; ADDR,*                        6421
000056  012746  000000  MOV     #EX_WRD, -(SP)
000062  012746  000002  MOV     #2, -(SP)
000066  010600          MOV     SP,R0                           ; SP,*
```

```
000070  104415                          TRAP    15
000072  062766  000000'  000016         ADD     2$,16(SP)           ; *,ADDR                    6422
000100  032701  000007                  BIT     #7,R1               ; *,COUNT                   6424
000104  001405                          BEQ     3$
000106  020166  000014                  CMP     R1,14(SP)           ; COUNT,LENGTH              6425
000112  001002                          BNE     4$
000114  012716  000000G           5$:    MOV     #CRLF,(SP)          ;                           6427
000120  012746  000001                  MOV     #1,-(SP)
000124  010600                          MOV     SP,R0               ; SP,*
000126  104415                          TRAP    15
000130  005726                          TST     (SP)+
000132  026626  000000'           4$:    CMP     (SP)+,@(SP)+        ;                           6415
000136  005201                    5$:   INC     R1                  ; COUNT                     6415
000140  020166  000010                  CMP     R1,10(SP)           ; COUNT,LENGTH
000144  003757                          BLE     1$
000146  026626  000000'                 CMP     (SP)+,@(SP)+        ;                           6416
000152  012601                          MOV     (SP)+,R1            ;                           6402
000154  000207                          RTS     PC

; Routine Size:  53 words,    Routine Base:  $CODE$ + 11202
; Maximum stack depth per invocation:  8 words


;   6422  1
;   6431  1
```

```
;        1    routine EMS_LBN : novalue =
;        1
;        1    !+
;        1    !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;        1    !       THE CURRENT RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;        1    !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;        1    !       PACKET. HOWEVER, IF THE "FLAGS" FIELD OF THE CURRENT RETURN PACKET
;        1    !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;        1    !       PRINTED.
;        1    !
;        1    !       IMPLICIT INPUTS:
;        1    !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;        1    !-
;        1
;        2        begin
;        2
;        2        if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and             ! IF NO BAD BLOCK FOUND
;        3           (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;        2        then
;        2           PRINTX (EX_LBN, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);
;        2
;        2        if (not BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and             ! IF BAD BLOCKS FOUND AND REPLACED
;        3           (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;        2        then
;        2           PRINTX (EX_BBU, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;        2
;        2        if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF HOST REPLACEABLE BAD BLOCK FOUND
;        3           (not BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;        2        then
;        2           PRINTX (EX_BB, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;        2
;        2        if (BIT_TST (RP_ADDR [FLAGS], EF_BBR)) and                 ! IF MORE THAN 1 HOST REPLACEABLE BAD BLOCK FOUND
;        3           (BIT_TST (RP_ADDR [FLAGS], EF_BBU))
;        2        then
;        2           PRINTX (EX_BB1, .RP_ADDR [BBLK_LO], .RP_ADDR [BBLK_LO]);
;        1        end;

                                        .SBTTL  EMS_LBN ERROR MESSAGE SUBROUTINES
000000  016700  000000G        EMS.LBN:MOV     RP.ADDR,R0                      ;               6443
000004  105760  000015                 TSTB    15(R0)
000010  100417                         BMI     1$
000012  132760  000100  000015         BITB    #100,15(R0)                     ;               6449
000020  001013                         BNE     1$
000022  016046  000050                 MOV     50(R0),-(SP)                    ;               6451
000026  011646                         MOV     (SP),-(SP)
000030  012746  000000G                MOV     #EX.LBN,-(SP)
000034  012746  000003                 MOV     #3,-(SP)
000040  010600                         MOV     SP,R0
000042  104415                         TRAP    15                              ; SP,*
000044  062706  000010                 ADD     #10,SP
000050  016700  000000G        1$:     MOV     RP.ADDR,R0                      ;               6453
000054  105760  000015                 TSTB    15(R0)
```

```
000060  100417                          BMI     2$
000062  132760  000100  000015          BITB    #100,15(R0)             ;                       6454
000070  001413                          BEQ     2$
000072  016046  000040                  MOV     40(R0), -(SP)           ;                       6456
000076  011646                          MOV     (SP), -(SP)
000100  012746  0000006                 MOV     #EX.BBU,-(SP)
000104  012746  000003                  MOV     #3, -(SP)
000110  010600                          MOV     SP,R0                   ; SP,*
000112  104415                          TRAP    15
000114  062706  000010                  ADD     #10,SP
000120  013700  0000006         2$:     MOV     RP.ADDR,R0              ;                       6458
000124  105760  000015                  TSTB    15(R0)
000130  100017                          BPL     3$
000132  132760  000100  000015          BITB    #100,15(R0)             ;                       6459
000140  001013                          BNE     3$
000142  016046  000040                  MOV     40(R0), -(SP)           ;                       6461
000146  011646                          MOV     (SP), -(SP)
000150  012746  0000006                 MOV     #EX.BB,-(SP)
000154  012746  000003                  MOV     #3, -(SP)
000160  010600                          MOV     SP,R0                   ; SP,*
000162  104415                          TRAP    15
000164  062706  000010                  ADD     #10,SP
000170  013700  0000006         3$:     MOV     RP.ADDR,R0              ;                       6463
000174  105760  000015                  TSTB    15(R0)
000200  100017                          BPL     4$
000202  132760  000100  000015          BITB    #100,15(R0)             ;                       6464
000210  001413                          BEQ     4$
000212  016046  000040                  MOV     40(R0), -(SP)           ;                       6466
000216  011646                          MOV     (SP), -(SP)
000220  012746  0000006                 MOV     #EX.BB1,-(SP)
000224  012746  000003                  MOV     #3, -(SP)
000230  010600                          MOV     SP,R0                   ; SP,*
000232  104415                          TRAP    15
000234  062706  000010                  ADD     #10,SP
000240  000207                  4$:     RTS     PC                      ;                       6432

; Routine Size:  81 words,    Routine Base:  $CODE$ + 11354
; Maximum stack depth per invocation:  6 words
```

```
;    6468  1      routine EMS_BC : novalue =
;    6469  1
;    6470  1      !+
;    6471  1      !       THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE CURRENT
;    6472  1      !       RETURN PACKET: THE BYTE COUNT FROM THE COMMAND PACKET AND THE
;    6473  1      !       ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE PACKET).
;    6474  1      !
;    6475  1      !       IMPLICIT INPUTS:
;    6476  1      !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6477  1      !-
;    6478  1
;    6479  2          begin
;    6480  2          PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]);       ! "BYTE COUNT IN COMMAND: XXXXX."
;    6481  2          PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);         ! "ACTUAL # OF BYTES TRANSFERRED: XXXXX."
;    6482  1          end;                                        ! ROUTINE EMS_BC

                                .SBTTL  EMS.BC ERROR MESSAGE SUBROUTINES
000000  013700  000000G  EMS.BC: MOV     RP.ADDR,R0                      ;                       6480
000004  016046  000044           MOV     44(R0),-(SP)
000010  012746  000000G          MOV     #EX.CBC,-(SP)
000014  012746  000002           MOV     #2,-(SP)
000020  010600                   MOV     SP,R0                           ; SP,*
000022  104415                   TRAP    15
000024  013700  000000G          MOV     RP.ADDR,R0                      ;                       6481
000030  016016  000020           MOV     20(R0),(SP)
000034  012746  000000G          MOV     #EX.BC,-(SP)
000040  012746  000002           MOV     #2,-(SP)
000044  010600                   MOV     SP,R0                           ; SP,*
000046  104415                   TRAP    15
000050  062706  000012           ADD     #12,SP                          ;                       6479
000054  000207                   RTS     PC                              ;                       6468

; Routine Size:  23 words,    Routine Base:  $CODE$ + 11616
; Maximum stack depth per invocation:  7 words
```

```
;    6483  1      routine EMS_BD : novalue =
;    6484  1
;    6485  1      !*
;    6486  1      !       THIS ROUTINE PRINTS (EXTENDED) THE TWO WORD I/O BUFFER DESCRIPTOR
;    6487  1      !       APPEARING IN THE CURRENT RETURN PACKET.
;    6488  1      !
;    6489  1      !       IMPLICIT INPUTS:
;    6490  1      !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    6491  1      !-
;    6492  1
;    6493  1          PRINTX (EX_BD, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);       ! "I/O BUFFER DESCRIPTOR: XXXXXX XXXXXX"


                                        .SBTTL  EMS.BD ERROR MESSAGE SUBROUTINES
000000  013700  000006          EMS.BD: MOV     RP.ADDR,R0                      ;                                       6493
000004  016046  000024                  MOV     24(R0),-(SP)
000010  016046  000026                  MOV     26(R0),(SP)
000014  012746  000006                  MOV     #EX.BD,-(SP)
000020  012746  000003                  MOV     #3,(SP)
000024  010600                          MOV     SP,R0                           ; SP,*
000026  104415                          TRAP    15
000030  062706  000010                  ADD     #10,SP
000034  000207                          RTS     PC                              ;                                       6483

; Routine Size:  15 words,    Routine Base:  $CODE$ + 11674
; Maximum stack depth per invocation:  6 words
```

```
;    6494  1
;    6495  1
;    6496  1     routine EMS_RP : novalue =
;    6497  1
;    6498  1
;    6499  1     !+
;    6500  1     !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;    6501  1     !       OF THE CURRENT RETURN PACKET.
;    6502  1     !-
;    6503  1
;    6504  1
;    6505  2         begin
;    6506  2         EMS_SBC ();                                     ! SUB-CODE
;    6507  2         EMS_CMD ();                                     ! COMMAND (AND MODIFIER)
;    6508  2
;    6509  2         if (.RP_ADDR [ENDCOD] and OP_MSK) neq OP_ONL
;    6510  2
;    6511  2         then
;    6512  2             EMS_LBN ();                                 ! LBN OR BAD BLOCK NUMBER
;    6513  2
;    6514  2         if ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD) or
;    6515  3            ((.RP_ADDR [ENDCOD] and OP_MSK) eql OP_WRT)
;    6516  3
;    6517  2         then
;    6518  3             begin
;    6519  3             EMS_BC ();                                  ! BYTE COUNTS
;    6520  3             EMS_BD ();                                  ! I/O BUFFER DESCRIPTOR
;    6521  2             end;
;    6522  2
;    6523  2         EMS_TIM ();                                     ! TIME
;    6524  1         end;                                            ! ROUTINE EMS_RP


                        .SBTTL  EMS.RP ERROR MESSAGE SUBROUTINES
000000  010146          EMS.RP: MOV     R1,-(SP)                    ;                   6496
000002  004737  010010'         JSR     PC,EMS.SBC                  ;                   6506
000006  004737  010520'         JSR     PC,EMS.CMD                  ;                   6507
000012  013700  000006          MOV     RP.ADDR,R0                  ;                   6509
000016  116000  000014          MOVB    14(R0),R0
000022  042700  177600          BIC     #177600,R0
000026  020027  000011          CMP     R0,#11
000032  001402                  BEQ     1$
000034  004737  011354'         JSR     PC,EMS.LBN                  ;                   6512
000040  013700  000006      1$: MOV     RP.ADDR,R0                  ;                   6514
000044  116001  000014          MOVB    14(R0),R1
000050  042701  177600          BIC     #177600,R1
000054  020127  000041          CMP     R1,#41
000060  001407                  BEQ     2$
000062  116000  000014          MOVB    14(R0),R0                   ;                   6515
000066  042700  177600          BIC     #177600,R0
000072  020027  000042          CMP     R0,#42
000076  001004                  BNE     3$
000100  004737  011616'     2$: JSR     PC,EMS.BC                   ;                   6519
```

ZRQAM1		ZRQAM1	RD/R...B1	ZRQAM1	RD/R...B9	ZRQAM2	RD/R...B13
ZRQAM1	RD/R...C1	ZRQAM1	RD/R...C5	ZRQAM1	RD/R...C9	ZRQAM2	RD/R...C13
ZRQAM1	RD/R...D1	ZRQAM1	RD/R...D5	ZRQAM1	RD/R...D9	ZRQAM2	RD/R...D13
ZRQAM1	RD/R...E1	ZRQAM1	RD/R...E5	ZRQAM1	RD/R...E9	ZRQAM2	RD/R...E13
ZRQAM1	RD/R...F1	ZRQAM1	RD/R...F5	ZRQAM1	RD/R...F9	ZRQAM2	RD/R...F13
ZRQAM1	RD/R...G1	ZRQAM1	RD/R...G5	ZRQAM1	RD/R...G9	ZRQAM2	RD/R...G13
ZRQAM1	RD/R...H1	ZRQAM1	RD/R...H5	ZRQAM1	RD/R...H9	ZRQAM2	RD/R...H13
ZRQAM1	RD/R...I1	ZRQAM1	RD/R...I5	ZRQAM1	RD/R...I9	ZRQAM2	RD/R...I13
ZRQAM1	RD/R...J1	ZRQAM1	RD/R...J5	ZRQAM2	RD/R...J9	ZRQAM2	RD/R...J13
ZRQAM1	RD/R...K1	ZRQAM1	RD/R...K5	ZRQAM2	RD/R...K9	ZRQAM2	RD/R...K13
ZRQAM1	RD/R...L1	ZRQAM1	RD/R...L5	ZRQAM2	RD/R...L9	ZRQAM2	RD/R...L13
ZRQAM1	RD/R...M1	ZRQAM1	RD/R...M5	ZRQAM2	RD/R...M9	ZRQAM2	RD/R...M13
ZRQAM1	RD/R...N1	ZRQAM1	RD/R...N5	ZRQAM2	RD/R...N9	ZRQAM2	RD/R...N13
ZRQAM1	RD/R...B2	ZRQAM1	RD/R...B6	ZRQAM2	RD/R...B10	ZRQAM2	RD/R...B14
ZRQAM1	RD/R...C2	ZRQAM1	RD/R...C6	ZRQAM2	RD/R...C10	ZRQAM2	RD/R...C14
ZRQAM1	RD/R...D2	ZRQAM1	RD/R...D6	ZRQAM2	RD/R...D10	ZRQAM2	RD/R...D14
ZRQAM1	RD/R...E2	ZRQAM1	RD/R...E6	ZRQAM2	RD/R...E10	ZRQAM2	RD/R...E14
ZRQAM1	RD/R...F2	ZRQAM1	RD/R...F6	ZRQAM2	RD/R...F10	ZRQAM2	RD/R...F14
ZRQAM1	RD/R...G2	ZRQAM1	RD/R...G6	ZRQAM2	RD/R...G10	ZRQAM2	RD/R...G14
ZRQAM1	RD/R...H2	ZRQAM1	RD/R...H6	ZRQAM2	RD/R...H10	ZRQAM2	RD/R...H14
ZRQAM1	RD/R...I2	ZRQAM1	RD/R...I6	ZRQAM2	RD/R...I10	ZRQAM2	RD/R...I14
ZRQAM1	RD/R...J2	ZRQAM1	RD/R...J6	ZRQAM2	RD/R...J10	ZRQAM2	RD/R...J14
ZRQAM1	RD/R...K2	ZRQAM1	RD/R...K6	ZRQAM2	RD/R...K10	ZRQAM2	RD/R...K14
ZRQAM1	RD/R...L2	ZRQAM1	RD/R...L6	ZRQAM2	RD/R...L10	ZRQAM2	RD/R...L14
ZRQAM1	RD/R...M2	ZRQAM1	RD/R...M6	ZRQAM2	RD/R...M10	ZRQAM2	RD/R...M14
ZRQAM1	RD/R...N2	ZRQAM1	RD/R...N6	ZRQAM2	RD/R...N10	ZRQAM2	RD/R...N14
ZRQAM1	RD/R...B3	ZRQAM1	RD/R...B7	ZRQAM2	RD/R...B11	ZRQAM2	RD/R...B15
ZRQAM1	RD/R...C3	ZRQAM1	RD/R...C7	ZRQAM2	RD/R...C11	ZRQAM2	RD/R...C15
ZRQAM1	RD/R...D3	ZRQAM1	RD/R...D7	ZRQAM2	RD/R...D11	ZRQAM2	RD/R...D15
ZRQAM1	RD/R...E3	ZRQAM1	RD/R...E7	ZRQAM2	RD/R...E11	ZRQAM2	RD/R...E15
ZRQAM1	RD/R...F3	ZRQAM1	RD/R...F7	ZRQAM2	RD/R...F11	ZRQAM2	RD/R...F15
ZRQAM1	RD/R...G3	ZRQAM1	RD/R...G7	ZRQAM2	RD/R...G11	ZRQAM2	RD/R...G15
ZRQAM1	RD/R...H3	ZRQAM1	RD/R...H7	ZRQAM2	RD/R...H11	ZRQAM2	RD/R...H15
ZRQAM1	RD/R...I3	ZRQAM1	RD/R...I7	ZRQAM2	RD/R...I11	ZRQAM2	RD/R...I15
ZRQAM1	RD/R...J3	ZRQAM1	RD/R...J7	ZRQAM2	RD/R...J11	ZRQAM2	RD/R...J15
ZRQAM1	RD/R...K3	ZRQAM1	RD/R...K7	ZRQAM2	RD/R...K11	ZRQAM2	RD/R...K15
ZRQAM1	RD/R...L3	ZRQAM1	RD/R...L7	ZRQAM2	RD/R...L11	ZRQAM2	RD/R...L15
ZRQAM1	RD/R...M3	ZRQAM1	RD/R...M7	ZRQAM2	RD/R...M11	ZRQAM2	RD/R...M15
ZRQAM1	RD/R...N3	ZRQAM1	RD/R...N7	ZRQAM2	RD/R...N11	ZRQAM2	RD/R...N15
ZRQAM1	RD/R...B4	ZRQAM1	RD/R...B8	ZRQAM2	RD/R...B12	ZRQAM2	RD/R...B16
ZRQAM1	RD/R...C4	ZRQAM1	RD/R...C8	ZRQAM2	RD/R...C12	ZRQAM2	RD/R...C16
ZRQAM1	RD/R...D4	ZRQAM1	RD/R...D8	ZRQAM2	RD/R...D12	ZRQAM2	RD/R...D16
ZRQAM1	RD/R...E4	ZRQAM1	RD/R...E8	ZRQAM2	RD/R...E12	ZRQAM2	RD/R...E16
ZRQAM1	RD/R...F4	ZRQAM1	RD/R...F8	ZRQAM2	RD/R...F12	ZRQAM2	RD/R...F16
ZRQAM1	RD/R...G4	ZRQAM1	RD/R...G8	ZRQAM2	RD/R...G12	ZRQAM2	RD/R...G16
ZRQAM1	RD/R...H4	ZRQAM1	RD/R...H8	ZRQAM2	RD/R...H12	ZRQAM2	RD/R...H16
ZRQAM1	RD/R...I4	ZRQAM1	RD/R...I8	ZRQAM2	RD/R...I12	ZRQAM2	RD/R...I16
ZRQAM1	RD/R...J4	ZRQAM1	RD/R...J8	ZRQAM2	RD/R...J12	ZRQAM2	RD/R...J16
ZRQAM1	RD/R...K4	ZRQAM1	RD/R...K8	ZRQAM2	RD/R...K12	ZRQAM2	RD/R...K16
ZRQAM1	RD/R...L4	ZRQAM1	RD/R...L8	ZRQAM2	RD/R...L12	ZRQAM2	RD/R...L16
ZRQAM1	RD/R...M4	ZRQAM1	RD/R...M8	ZRQAM2	RD/R...M12	ZRQAM2	RD/R...M16
ZRQAM1	RD/R...N4	ZRQAM1	RD/R...N8	ZRQAM2	RD/R...N12	ZRQAM2	RD/R...N16

```
000104  004737  011674'                        JSR     PC,EMS.dD                       ;                6520
000110  004737  00C000V             3$:        JSR     PC,EMS,TIM                      ;                6523
000114  012601                               MOV     (SP)+,R1                        ;                6496
000116  000207                               RTS     PC

; Routine Size:  40 words,      Routine Base:  $CODE$ + 11732
; Maximum stack depth per invocation:  2 words


;   6525  1
;   6526  1
```

```
;    6527  1      global routine EMS_RP1 : novalue =
;    6528  1
;    6529  1      !+
;    6530  1      !       THIS  ROUTINE  IS  CALLED  TO  PRINT  THE  ENTIRE CONTENTS OF THE
;    6531  1      !       RETURN PACKET DESIGNATED BY THE GLOBAL DATUM "RP_ADDR". HOWEVER, THE
;    6532  1      !       PRINTING WILL ONLY OCCUR IF EXTENDED ERROR PRINTING IS ENABLED.
;    6533  1      !-
;    6534  1
;    6535  2          begin
;    6536  2          PRINTX (EX_RP);                         ! "CONTENTS OF RETURN PACKET:"
;    6537  2          EMS_BLK (.RP_ADDR, RP_LEN);             ! PRINT BLOCK OF WORDS
;    6538  1          end;


                                        .SBTTL  EMS.RP1 ERROR MESSAGE SUBROUTINES
000000  012746  000000G        EMS.RP1::
                                MOV     #EX.RP, -(SP)           ;                       6536
000004  012746  000001          MOV     #1, -(SP)
000010  010600                  MOV     SP,R0                   ; SP,*
000012  104415                  TRAP    15
000014  013716  000000G        MOV     RP.ADDR,-(SP)           ;                       6537
000020  012746  000026          MOV     #26,-(SP)
000024  004737  011202'         JSR     PC,EMS.BLK
000030  062706  000006          ADD     #6,SP                   ;                       6535
000034  000207                  RTS     PC                      ;                       6527

; Routine Size:  15 words,      Routine Base:  $CODE$ + 12052
; Maximum stack depth per invocation:  4 words
```

```
;    6539  1     global routine EMS_EL (index) : novalue =
;    6540  1
;    6541  1     !+
;    6542  1     !       THIS ROUTINE IS CALLED FROM 'SEQUEN' AND 'DATAGM' AND PRINTS THE CONTENTS OF THE
;    6543  1     !       ERROR-LOG PACKET
;    6544  1     !-
;    6545  1
;    6546  2         begin
;    6547  2
;    6548  2         local
;    6549  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;    6550  2             REASON : word,
;    6551  2             DISK_NUM : byte,
;    6552  2             ELOG_CODE : byte,
;    6553  2             ELOG_SUB : word;
;    6554  2
;    6555  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                   ! ERROR LOG PACKET'S ADDRESS
;    6556  2         REASON = .ELOG_ADDR [EL_FORMAT];                                ! FORMAT
;    6557  2         DISK_NUM = .ELOG_ADDR [EL_DK_NUM];                              ! DISK NUMBER
;    6558  2         ELOG_CODE = .ELOG_ADDR [EL_CODE];                               ! CODE
;    6559  2         ELOG_SUB = .ELOG_ADDR [EL_SUBCODE];                             ! SUBCODE
;    6560  2         PRINTB (ELG_00);                                                ! ERROR-LOG MESSAGE RECEIVED
;    6561  2
;    6562  2         if (.REASON eql FORMAT_CNTR) or
;    6563  3            (.REASON eql FORMAT_HOST)
;    6564  2         then
;    6565  3             PRINTB (.ELG_FMT [.REASON])                                 ! PRINT BASIC REASON
;    6566  2         else
;    6567  2             PRINTB (.ELG_FMT [.REASON], .DISK_NUM);                     ! PRINT BASIC REASON WITH DISK NUMBER
;    6568  2
;    6569  2         if (.ELOG_CODE gtru 0) and
;    6570  3            (.ELOG_CODE lequ 11)
;    6571  2         then
;    6572  3             begin
;    6573  3             PRINTX (ASTERISK);
;    6574  3             PRINTX (.ERR_COD [.ELOG_CODE - 1]);                         ! CODE
;    6575  3             end
;    6576  2         else
;    6577  2
;    6578  2             if .ELOG_CODE eql ST_DIA                                    ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    6579  2             then
;    6580  3                 begin
;    6581  3                 PRINTX (ASTERISK);
;    6582  3                 PRINTX (.ERR_COD [12]);
;    6583  2                 end;
;    6584  2
;    6585  2         if (.ELOG_CODE eql ST_MFE) and
;    6586  3            (.ELOG_SUB lequ 10)
;    6587  2         then
;    6588  3             begin
;    6589  3             PRINTX (CRLF);
;    6590  3             PRINTX (ASTERISK);
;    6591  3             PRINTX (.TBL_MFE [.ELOG_SUB]);                              ! MEDIA FORMAT ERROR
```

```
;    6592  2              end;
;    6593  2
;    6594  2          if (.ELOG_CODE eql ST_DAT) and
;    6595  3             (.ELOG_SUB lequ 15)
;    6596  2          then
;    6597  3              begin
;    6598  3              PRINTX (CRLF);
;    6599  3              PRINTX (ASTERISK);
;    6600  3              PRINTX (.TBL_DAT [.ELOG_SUB]);                          ! DATA ERROR
;    6601  2              end;
;    6602  2
;    6603  2          if (.ELOG_CODE eql ST_HST) and
;    6604  3             (.ELOG_SUB lequ 4)
;    6605  2          then
;    6606  3              begin
;    6607  3              PRINTX (CRLF);
;    6608  3              PRINTX (ASTERISK);
;    6609  3              PRINTX (.TBL_HST [.ELOG_SUB]);                          ! HOST ACCESS ERROR
;    6610  2              end;
;    6611  2
;    6612  2          if (.ELOG_CODE eql ST_CNT) and
;    6613  3             (.ELOG_SUB lequ 3)
;    6614  2          then
;    6615  3              begin
;    6616  3              PRINTX (CRLF);
;    6617  3              PRINTX (ASTERISK);
;    6618  3              PRINTX (.TBL_CNT [.ELOG_SUB]);                          ! CONTROLLER ERROR
;    6619  2              end;
;    6620  2
;    6621  2          if (.ELOG_CODE eql ST_DRV) and
;    6622  3             (.ELOG_SUB lequ 8)
;    6623  2          then
;    6624  3              begin
;    6625  3              PRINTX (CRLF);
;    6626  3              PRINTX (ASTERISK);
;    6627  3              PRINTX (.TBL_DRV [.ELOG_SUB]);                          ! DRIVE ERROR
;    6628  2              end;
;    6629  2
;    6630  2          if .REASON eql FORMAT_XFER                                  ! IF DISK XFER INVOLVED
;    6631  2          then
;    6632  2
;    6633  2              if .ELOG_ADDR [EL_BLOCK_TYPE] eql TYPE_LBN              ! PRINT PBN OR RBN
;    6634  2              then
;    6635  3                  PRINTX (EX_PBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK])
;    6636  2              else
;    6637  2                  PRINTX (EX_RBN, .ELOG_ADDR [EL_BLOCK], .ELOG_ADDR [EL_BLOCK]);
;    6638  2
;    6639  2          EMS_TIM ();                                                 ! TIME
;    6640  2          EMS_BLK ((.ELOG_ADDR + 2), ((.ELOG_ADDR [EL_MSGLEN] + 1) / 2) + 2); ! PRINTX CONTENTS OF PACKET
;    6641  2          ELOG_ADDR [EL_CONTENTS] = EMPTY;                            ! DECLARE SAVE AREA FREE
;    6642  2
;    6643  1          end;
```

```
                                        .SBTTL  EMS,EL ERROR MESSAGE SUBROUTINES
000000  004137  000000G           EMS.EL::JSR     R1,$SAVE5                       ;                                       6539
000004  005746                            TST     -(SP)
000006  016646  000020                    MOV     20(SP),-(SP)                    ; INDEX,*                               6555
000012  012746  000102                    MOV     #102,-(SP)
000016  004737  000000G                   JSR     PC,BL$MUL
000022  062700  000000G                   ADD     #ELOG.PKT,R0
000026  010001                            MOV     R0,R1                           ; *,ELOG.ADDR
000030  116166  000016  000004            MOVB    16(R1),4(SP)                    ; *(ELOG.ADDR),REASON                   6556
000036  105066  000005                    CLRB    5(SP)                           ; REASON
000042  116105  000012                    MOVB    12(R1),R5                       ; *(ELOG.ADDR),DISK.NUM                 6557
000046  116100  000020                    MOVB    20(R1),R0                       ; *(ELOG.ADDR),*                        6558
000052  042700  177740                    BIC     #177740,R0
000056  105004                            CLRB    R4                              ; ELOG.CODE
000060  050004                            BIS     R0,R4                           ; *,ELOG.CODE
000062  016103  000020                    MOV     20(R1),R3                       ; *(ELOG.ADDR),ELOG.SUB                 6559
000066  006203                            ASR     R3                              ; ELOG.SUB
000070  006203                            ASR     R3                              ; ELOG.SUB
000072  006203                            ASR     R3                              ; ELOG.SUB
000074  006203                            ASR     R3                              ; ELOG.SUB
000076  006203                            ASR     R3                              ; ELOG.SUB
000100  042703  174000                    BIC     #174000,R3                      ; *,ELOG.SUB
000104  012716  000000G                   MOV     #ELG.00,(SP)                    ;                                       6560
000110  012746  000001                    MOV     #1,-(SP)
000114  010600                            MOV     SP,R0                           ; SP,*
000116  104414                            TRAP    14
000120  016602  000006                    MOV     6(SP),R2                        ; REASON,*                              6565
000124  006302                            ASL     R2
000126  005766  000006                    TST     6(SP)                           ; REASON                                6562
000132  001404                            BEQ     1$
000134  026627  000006  000001            CMP     6(SP),#1                        ; REASON,*                              6563
000142  001007                            BNE     2$
000144  016216  000000G           1$:     MOV     ELG.FMT(R2),(SP)                ;                                       6565
000150  012746  000001                    MOV     #1,-(SP)
000154  010600                            MOV     SP,R0                           ; SP,*
000156  104414                            TRAP    14
000160  000411                            BR      3$                              ;                                       6566
000162  005016                    2$:     CLR     (SP)                            ;                                       6567
000164  110516                            MOVB    R5,(SP)                         ; DISK.NUM,*
000166  016246  000000G                   MOV     ELG.FMT(R2),-(SP)
000172  012746  000002                    MOV     #2,-(SP)
000176  010600                            MOV     SP,R0                           ; SP,*
000200  104414                            TRAP    14
000202  005726                            TST     (SP)+
000204  105704                    3$:     TSTB    R4                              ; ELOG.CODE                             6569
000206  001423                            BEQ     4$
000210  120427  000013                    CMPB    R4,#13                          ; ELOG.CODE,*                           6570
000214  101020                            BHI     4$
000216  012716  000000G                   MOV     #ASTERISK,(SP)                  ;                                       6573
000222  012746  000001                    MOV     #1,-(SP)
000226  010600                            MOV     SP,R0                           ; SP,*
000230  104415                            TRAP    15
```

```
000232  005000                          CLR     R0                              ;                               6574
000234  150400                          BISB    R4,R0                           ; ELOG.CODE,*
000236  006300                          ASL     R0
000240  016016  177776G                 MOV     ERR.COD-2(R0),(SP)
000244  012746  000001                  MOV     #1,-(SP)
000250  010600                          MOV     SP,R0                           ; SP,*
000252  104415                          TRAP    15
000254  000417                          BR      5$                              ;                               6572
000256  120427  000037          4$:     CMPB    R4,#37                          ; ELOG.CODE,*                   6578
000262  001015                          BNE     6$
000264  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6581
000270  012746  000001                  MOV     #1,-(SP)
000274  010600                          MOV     SP,R0                           ; SP,*
000276  104415                          TRAP    15
000300  013716  000030G                 MOV     ERR.COD+30,(SP)                 ;                               6582
000304  012746  000001                  MOV     #1,-(SP)
000310  010600                          MOV     SP,R0                           ; SP,*
000312  104415                          TRAP    15
000314  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               6580
000316  120427  000005          6$:     CMPB    R4,#5                           ; ELOG.CODE,*                   6585
000322  001031                          BNE     7$
000324  020327  000012                  CMP     R3,#12                          ; ELOG.SUB,*                    6586
000330  101026                          BHI     7$
000332  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6589
000336  012746  000001                  MOV     #1,-(SP)
000342  010600                          MOV     SP,R0                           ; SP,*
000344  104415                          TRAP    15
000346  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6590
000352  012746  000001                  MOV     #1,-(SP)
000356  010600                          MOV     SP,R0                           ; SP,*
000360  104415                          TRAP    15
000362  010300                          MOV     R3,R0                           ; ELOG.SUB,*                    6591
000364  006300                          ASL     R0
000366  016016  000064'                 MOV     TBL.MFE(R0),(SP)
000372  012746  000001                  MOV     #1,-(SP)
000376  010600                          MOV     SP,R0                           ; SP,*
000400  104415                          TRAP    15
000402  062706  000006                  ADD     #6,SP                           ;                               6583
000406  120427  000010          7$:     CMPB    R4,#10                          ; ELOG.CODE,*                   6594
000412  001031                          BNL     8$
000414  020327  000017                  CMP     R3,#17                          ; ELOG.SUB,*                    6595
000420  101026                          BHI     8$
000422  012716  000000G                 MOV     #CRLF,(SP)                      ;                               6598
000426  012746  000001                  MOV     #1,-(SP)
000432  010600                          MOV     SP,R0                           ; SP,*
000434  104415                          TRAP    15
000436  012716  000000G                 MOV     #ASTERISK,(SP)                  ;                               6599
000442  012746  000001                  MOV     #1,-(SP)
000446  010600                          MOV     SP,R0                           ; SP,*
000450  104415                          TRAP    15
000452  010300                          MOV     R3,R0                           ; ELOG.SUB,*                    6600
000454  006300                          ASL     R0
000456  016016  000120'                 MOV     TBL.DAT(R0),(SP)
```

```
000462  012746  000001                    MOV     #1,-(SP)
000466  010600                            MOV     SP,R0                   ; SP,*
000470  104415                            TRAP    15
000472  062706  000006                    ADD     #6,SP                   ;                               6597
000476  120427  000011          8$:       CMPB    R4,#11                  ; ELOG.CODE,*                   6603
000502  001031                            BNE     9$
000504  020327  000004                    CMP     R3,#4                   ; ELOG.SUB,*                    6604
000510  101026                            BHI     9$
000512  012716  000000G                   MOV     #CRLF,(SP)              ;                               6607
000516  012746  000001                    MOV     #1,-(SP)
000522  010600                            MOV     SP,R0                   ; SP,*
000524  104415                            TRAP    15
000526  012716  000000G                   MOV     #ASTERISK,(SP)          ;                               6608
000532  012746  000001                    MOV     #1,-(SP)
000536  010600                            MOV     SP,R0                   ; SP,*
000540  104415                            TRAP    15
000542  010300                            MOV     R3,R0                   ; ELOG.SUB,*                    6609
000544  006300                            ASL     R0
000546  016016  000160'                   MOV     TBL.MSG(R0),(SP)
000552  012746  000001                    MOV     #1,-(SP)
000556  010600                            MOV     SP,R0                   ; SP,*
000560  104415                            TRAP    15
000562  062706  000006                    ADD     #6,SP                   ;                               6606
000566  120427  000012          9$:       CMPB    R4,#12                  ; ELOG.CODE,*                   6612
000572  001031                            BNE     10$
000574  020327  000003                    CMP     R3,#3                   ; ELOG.SUB,*                    6613
000600  101026                            BHI     10$
000602  012716  000000G                   MOV     #CRLF,(SP)              ;                               6616
000606  012746  000001                    MOV     #1,-(SP)
000612  010600                            MOV     SP,R0                   ; SP,*
000614  104415                            TRAP    15
000616  012716  000000G                   MOV     #ASTERISK,(SP)          ;                               6617
000622  012746  000001                    MOV     #1,-(SP)
000626  010600                            MOV     SP,R0                   ; SP,*
000630  104415                            TRAP    15
000632  010300                            MOV     R3,R0                   ; ELOG.SUB,*                    6618
000634  006300                            ASL     R0
000636  016016  000172'                   MOV     TBL.CNT(R0),(SP)
000642  012746  000001                    MOV     #1,-(SP)
000646  010600                            MOV     SP,R0                   ; SP,*
000650  104415                            TRAP    15
000652  062706  000006                    ADD     #6,SP                   ;                               6615
000656  120427  000013          10$:      CMPB    R4,#13                  ; ELOG.CODE,*                   6621
000662  001031                            BNE     11$
000664  020327  000010                    CMP     R3,#10                  ; ELOG.SUB,*                    6622
000670  101026                            BHI     11$
000672  012716  000000G                   MOV     #CRLF,(SP)              ;                               6625
000676  012746  000001                    MOV     #1,-(SP)
000702  010600                            MOV     SP,R0                   ; SP,*
000704  104415                            TRAP    15
000706  012716  000000G                   MOV     #ASTERISK,(SP)          ;                               6626
000712  012746  000001                    MOV     #1,-(SP)
000716  010600                            MOV     SP,R0                   ; SP,*
```

```
000720  104415                          TRAP    15
000722  010300                          MOV     R3,R0                   ; EL.OG.SUB,*                   6627
000724  006300                          ASL     R0
000726  016016  000020'                  MOV     TBL.DRV(R0),(SP)
000732  012746  000001                  MOV     #1,-(SP)
000736  010600                          MOV     SP,R0                   ; SP,*
000740  104415                          TRAP    15
000742  062706  000006                  ADD     #6,SP                   ;                               6624
000746  026627  000010  000002  11$:    CMP     10(SP),#2               ; REASON,*                      6630
000754  001031                          BNE     14$
000756  032761  170000  000060          BIT     #170000,60(R1)          ; *,*(EL.OG.ADDR)               6633
000764  001013                          BNE     12$
000766  016116  000056                  MOV     56(R1),(SP)             ; *(EL.OG.ADDR),*               6635
000772  011646                          MOV     (SP), (SP)
000774  012746  000000G                 MOV     #EX.PBN,-(SP)
001000  012746  000003                  MOV     #3,-(SP)
001004  010600                          MOV     SP,R0                   ; SP,*
001006  104415                          TRAP    15
001010  000411                          BR      13$                     ;                               6633
001012  016116  000056          12$:    MOV     56(R1),(SP)             ; *(EL.OG.ADDR),*               6637
001016  011646                          MOV     (SP),-(SP)
001020  012746  000000G                 MOV     #EX.RBN,-(SP)
001024  012746  000003                  MOV     #3, (SP)
001030  010600                          MOV     SP,R0                   ; SP,*
001032  104415                          TRAP    15
001034  062706  000006          13$:    ADD     #6,SP                   ;                               6633
001040  004737  000000V         14$:    JSR     PC,EMS.TIM              ;                               6639
001044  012716  000002                  MOV     #2,(SP)                 ;                               6640
001050  060116                          ADD     R1,(SP)                 ; .L OG.ADDR,*
001052  016146  000002                  MOV     2(R1),-(SP)             ; *(EL.OG.ADDR),*
001056  005216                          INC     (SP)
001060  012746  000002                  MOV     #2,-(SP)
001064  004737  000000G                 JSR     PC,BL$DIV
001070  010066  000002                  MOV     R0,2(SP)
001074  062766  000002  000002          ADD     #2,2(SP)
001102  005726                          TST     (SP)+
001104  004737  011202'                 JSR     PC,EMS.BLK
001110  105061  000001                  CLRB    1(R1)                   ; *(EL.OG.ADDR)                 6641
001114  062706  000014                  ADD     #14,SP                  ;                               6539
001120  000207                          RTS     PC

; Routine Size:  297 words,    Routine Base:  $CODE$ + 12110
; Maximum stack depth per invocation:  16 words
```

```
;    6644  1     global routine EMS_CMP (ADDR) : novalue =
;    6645  1
;    6646  1     !+
;    6647  1     ! THIS ROUTINE IS CALLED FROM 'HOST_WRT_CHK' AND PRINTS RELEVENT DATA ON A HOST
;    6648  1     ! COMPARE ERROR
;    6649  1     !-
;    6650  1
;    6651  2         begin
;    6652  2
;    6653  2         local
;    6654  2             ORIG_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    6655  2
;    6656  2         ORIG_ADDR = .ADDR;                                          ! ADDRESS OF THE WRITE RETPKT
;    6657  2         PRINTB (ERR_00, .CDISK);                                    ! "DISK XXX"
;    6658  2         PRINTB (DASH);                                              !
;    6659  2         PRINTB (.ERR_COD [12]);                                     ! " - HOST COMPARE ERROR"
;    6660  2         PRINTX (EX_LBW, .ORIG_ADDR [LBN_LO], .ORIG_ADDR [LBN_LO]);  ! LBN (WRITE)
;    6661  2         PRINTX (EX_LBR, .RP_ADDR [LBN_LO], .RP_ADDR [LBN_LO]);      ! LBN (READ)
;    6662  2         PRINTX (EX_CBW, .ORIG_ADDR [CBCNT_LO]);                     ! BYTE COUNT (WRITE)
;    6663  2         PRINTX (EX_BC, .ORIG_ADDR [BCNT_LO]);                       ! BYTE COUNT XMITTED (WRITE)
;    6664  2         PRINTX (EX_CBR, .RP_ADDR [CBCNT_LO]);                       ! BYTE COUNT (READ);
;    6665  2         PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);                         ! BYTE COUNT XMITTED (READ)
;    6666  2         PRINTX (EX_BDW, .ORIG_ADDR [BUFF_1], .ORIG_ADDR [BUFF_0]);  ! BUFFER ADDRESS (WRITE)
;    6667  2         PRINTX (EX_BDR, .RP_ADDR [BUFF_1], .RP_ADDR [BUFF_0]);      ! BUFFER ADDRESS (READ)
;    6668  2         EMS_TIM ();                                                 ! TIME
;    6669  1         end;


                                        .SBTTL  EMS.CMP ERROR MESSAGE SUBROUTINES
000000  010146                    EMS.CMP::
                                    MOV     R1,-(SP)                ;                               6644
000002  016601  000004              MOV     4(SP),R1                ; ADDR,ORIG.ADDR                6656
000006  013746  000000G             MOV     CDISK,-(SP)             ;                               6657
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)              ;                               6658
000032  012746  000001              MOV     #1,-(SP)
000036  010600                      MOV     SP,R0                   ; SP,*
000040  104414                      TRAP    14
000042  013716  000030G             MOV     ERR.COD+30,(SP)         ;                               6659
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),*                6660
000062  011646                      MOV     (SP),-(SP)
000064  012746  000000G             MOV     #EX.LBW,-(SP)
000070  012746  000003              MOV     #3, (SP)
000074  010600                      MOV     SP,R0                   ; SP,*
000076  104415                      TRAP    15
000100  013700  000000G             MOV     RP.ADDR,R0              ;                               6661
000104  016016  000050              MOV     50(R0),(SP)
```

```
000004  005003                          CLR     R3                              ; RESULT                        2884
000006  012702  000001                  MOV     #1,R2                           ; *,PASS
000012  005037  000000G                 CLR     SB.CODE                         ;                               2891
000016  005037  000000G                 CLR     ST.CODE
000022  013700  000000G                 MOV     L$LUN,R0                        ;                               2892
000026  006300                          ASL     R0
000030  016000  000054'                  MOV     MAX.LBN(R0),R0
000034  060200                          ADD     R2,R0
000036  006200                          ASR     R0
000040  010004                          MOV     R0,R4                           ; *,LBN
000042  042704  100000                  BIC     #100000,R4                      ; *,LBN
000046  005304                          DEC     R4                              ; LBN
000050  013746  000000G         1$:     MOV     CCTLR,-(SP)                     ;                               2897
000054  004737  000000G                 JSR     PC,GET.PKT
000060  010037  000000G                 MOV     R0,P.INDEX
000064  010016                          MOV     R0,(SP)                         ; P.INDEX,*                     2898
000066  012746  000106                  MOV     #106,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  013760  000000G  000016G        MOV     CDISK,MSCP.PKT+16(R0)
000104  112760  000020  000022G        MOVB    #20,MSCP.PKT+22(R0)             ;                               2899
000112  012760  001000  000026G        MOV     #1000,MSCP.PKT+26(R0)           ;                               2900
000120  010460  000046G                 MOV     R4,MSCP.PKT+46(R0)              ; LBN,*                         2901
000124  112760  000002  000004G        MOVB    #2,MSCP.PKT+4(R0)               ;                               2902
000132  013716  000000G                 MOV     P.INDEX,(SP)                    ;                               2904
000136  004737  000000G                 JSR     PC,SEND
000142  005700                          TST     R0
000144  001007                          BNE     2$
000146  013716  000000G                 MOV     P.INDEX,(SP)                    ;                               2907
000152  004737  000000G                 JSR     PC,PUT.PKT
000156  012702  000002                  MOV     #2,R2                           ; *,PASS                        2908
000162  000522                          BR      9$                              ;                               2904
000164  004737  000000G         2$:     JSR     PC,WAIT                         ;                               2915
000170  004737  000000G                 JSR     PC,OUT.IODQ                     ;                               2916
000174  010037  000000G                 MOV     R0,RP.INDX
000200  010016                          MOV     R0,(SP)                         ; RP.INDX,*                     2917
000202  012746  000054                  MOV     #54,-(SP)
000206  004737  000000G                 JSR     PC,BL$MUL
000212  062700  000000G                 ADD     #RETPKT,R0
000216  010037  000000G                 MOV     R0,RP.ADDR
000222  132760  000360  000002         BITB    #360,2(R0)                      ;                               2919
000230  001404                          BEQ     3$
000232  013716  000000G                 MOV     RP.INDX,(SP)                    ;                               2921
000236  004737  000000G                 JSR     PC,PUT.RETPKT
000242  005726                  3$:     TST     (SP)+                           ;                               2914
000244  013701  000000G                 MOV     RP.ADDR,R1                      ;                               2924
000250  005000                          CLR     R0
000252  126127  000003  000003         CMPB    3(R1),#3
000260  001002                          BNE     4$
000262  005200                          INC     R0
000264  000407                          BR      5$
000266  132761  000360  000002         BITB    #360,2(R1)                      ;                               2925
000274  001333                          BNE     2$
000276  105761  000014                  TSTB    14(R1)                          ;                               2926
```

```
;    6670  1     global routine EMS.ERR : novalue =
;    6671  1
;    6672  2         begin
;    6673  2
;    6674  2     ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;    6675  2     !
;    6676  2         PRINTB (ERR_00, .CDISK);                        ! "DISK XXX"
;    6677  2         PRINTB (DASH);                                  !
;    6678  2
;    6679  2         if (.ST_CODE gtru 0) and                        ! IF STATUS CODE IS WITHIN RANGE
;    6680  3            (.ST_CODE lequ 11)
;    6681  2         then
;    6682  3             PRINTB (.ERR_COD [.ST_CODE - 1])            ! PRINTB APPROPRIATE MESSAGE
;    6683  2         else
;    6684  2
;    6685  2             if .ST_CODE eql ST_DIA
;    6686  2             then
;    6687  3                 PRINTB (.ERR_COD [11])                  ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    6688  2             else
;    6689  2                 PRINTB (EX_SC, .ST_CODE);               ! JUST PRINT STATUS CODE WHEN NO MATCH
;    6690  2
;    6691  2         EMS_RP ();                                      ! PRINTX OTHER RETPKT FIELDS
;    6692  2
;    6693  1         end;


                                        .SBTTL  EMS.ERR ERROR MESSAGE SUBROUTINES
000000  013746  000000G           EMS.ERR::
000004  012746  000000G                   MOV     CDISK,-(SP)             ;                       6676
000010  012746  000002                    MOV     #ERR.00,-(SP)
000014  010600                            MOV     #2,-(SP)
000016  104414                            MOV     SP,R0                   ; SP,*
000020  012716  000000G                   TRAP    14
000024  012746  000001                    MOV     #DASH,(SP)              ;                       6677
000030  010600                            MOV     #1,-(SP)
000032  104414                            MOV     SP,R0                   ; SP,*
000034  013700  000000G                   TRAP    14
000040  001413                            MOV     ST.CODE,R0              ;                       6679
000042  020027  000013                    BEQ     1$
00004   101010                            CMP     R0,#13                  ;                       6680
000050  006300                            BHI     1$
000052  016016  177776G                   ASL     R0                      ;                       6682
000056  012746  000001                    MOV     ERR.COD-2(R0),(SP)
000062  010600                            MOV     #1,(SP)
000064  104414                            MOV     SP,R0                   ; SP,*
000066  000422                            TRAP    14
000070  020027  000037                    BR      3$                      ;                       6679
000074  001007                    1$:     CMP     R0,#37                  ;                       6685
000076  013716  000026G                   BNE     2$
000102  012746  000001                    MOV     ERR.COD+26,(SP)         ;                       6687
000106  010600                            MOV     #1,(SP)
000110  104414                            MOV     SP,R0                   ; SP,*
                                          TRAP    14
```

M1


```
000112  000410                          BR      3$                              ;                       6685
000114  010016                  2$:     MOV     R0,(SP)                         ;                       6689
000116  012746  000000G                 MOV     #EX.SC,-(SP)
000122  012746  000002                  MOV     #2,-(SP)
000126  010600                          MOV     SP,R0                           ; SP,*
000130  104414                          TRAP    14
000132  005726                          TST     (SP)+
000134  004737  011732'          3$:     JSR     PC,EMS.RP                       ;                       6691
000140  062706  000012                  ADD     #12,SP                          ;                       6672
000144  000207                          RTS     PC                              ;                       6670

; Routine Size:  51 words,      Routine Base:  $CODE$ + 13560
; Maximum stack depth per invocation:  8 words
```

```
;    6694  1     routine EMS_TIM : novalue =
;    6695  1
;    6696  1     !+
;    6697  1     ! THIS ROUTINE PRINTS THE TIME-OF-DAY MESSAGE
;    6698  1     !-
;    6699  1
;    6700  1     PRINTX (EX_TIM, .HOURS, .MINUTES);


                                        .SBTTL  EMS.TIM ERROR MESSAGE SUBROUTINES
000000  005046                  EMS.TIM:CLR     -(SP)                           ;                       6700
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  062700  000010                  ADD     #10,SP
000034  000207                          RTS     PC                              ;                       6694

; Routine Size:  15 words,      Routine Base:  $CODE$ + 13726
; Maximum stack depth per invocation:  6 words
```

```
;   6701  1     BGNMSG (EMS_01);


                                        .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.01::JSR     PC,M$EMS.01                     ;                       6701
000004  104423                         TRAP    23
000006  000207                         RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 13764
; Maximum stack depth per invocation:  2 words

;   6702  2     PRINTB (EBS_01, MAX_UNITS);             ! "MORE THAN XX UNITS SPECIFIED"
;   6703  1     ENDMSG;


                                        .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  012746  000004                M$EMS.01:
                                        MOV     #4,-(SP)                        ;                       6702
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                           ;                       6701
000024  000207                          RTS     PC

; Routine Size:  11 words,      Routine Base:  $CODE$ + 13774
; Maximum stack depth per invocation:  5 words
```

```
;   6704  1     BGNMSG (EMS_10);


                                        .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.10::JSR     PC,M$EMS.10                     ;                       6704
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14022
; Maximum stack depth per invocation:  2 words


;   6705  2     PRINTB (EBD_10, .RDRX_ADDR + .OF_RC);           ! "NO RESPONSE AT ADDRESS XXXXXX"
;   6706  1     ENDMSG;


                                        .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.10:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       6705
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                           ;                       6704
000030  000207                            RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14032
; Maximum stack depth per invocation:  5 words
```

```
;   6707  1     BGNMSG (EMS_12);


                                        .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.12::JSR     PC,@#EMS.12                     ;                       6707
000004  104423                           TRAP    23
000006  000207                           RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14064
; Maximum stack depth per invocation:  2 words


;   6708  2     PRINTB (EBD_12, .RDRX_ADDR);            ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX"
;   6709  1     ENDMSG;


                                        .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.12:
                                        MOV     RDRX.ADDR,-(SP)                 ;                       6708
000004  012746  000000G                 MOV     #EBD.12,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,R
000016  104414                          TRAP    14
000020  062706  000006                  ADD     #6,SP                           ;                       6707
000024  000207                          RTS     PC

; Routine Size:  11 words,      Routine Base:  $CODE$ + 14074
; Maximum stack depth per invocation:  5 words
```

```
;   6710  1     BGNMSG (EMS_13);


                                        .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.13::JSR     PC,M$EMS.13                     ;                       6710
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14122
; Maximum stack depth per invocation:  2 words


;   6711  2     PRINTB (EBD_13, .STEP);                 ! "STEP X READ ERROR"
;   6712  2     EMS_SA ();                              ! PRINTX SA CONTENTS
;   6713  1     ENDMSG;


                                        .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                M$EMS.13:
                                        MOV     STEP,-(SP)                      ;                       6711
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  007476'                JSR     PC,EMS.SA                       ;                       6712
000024  062706  000006                 ADD     #6,SP                           ;                       6710
000030  000207                         RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14132
; Maximum stack depth per invocation:  5 words
```

```
;   6714  1     BGNMSG (EMS_14);


                                          .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.14::JSR     PC,M$EMS.14                     ;                       6714
000004  104423                                 TRAP    23
000006  000207                                 RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14164
; Maximum stack depth per invocation:  2 words


;   6715  2     PRINTB (EBD_14, .IRDRX_ADDR);                 ! "BAD SA CODE FROM DEVICE XXXXXX"
;   6716  2     EMS_SA ();                                    ! PRINTX SA REGISTER CONTENTS
;   6717  1     ENDMSG;


                                          .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  013746  000000G                M$EMS.14:
                                         MOV     IRDRX.ADDR,-(SP)                ;                       6715
000004  012746  000000G                         MOV     #EBD.14,-(SP)
000010  012746  000002                          MOV     #2,-(SP)
000014  010600                                  MOV     SP,R0                           ; SP,A
000016  104414                                  TRAP    14
000020  004737  007476'                         JSR     PC,EMS.SA                       ;                       6716
000024  062706  000006                          ADD     #6,SP                           ;                       6714
000030  000207                                  RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14174
; Maximum stack depth per invocation:  5 words
```

```
;   6718  1     BGNMSG (EMS_18);


                                        .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.18::JSR     PC,M$EMS.18                     ;                       6718
000004  104423                           TRAP    23
000006  000207                           RTS     PC

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


;   6719  2     PRINTB (EBD_18, .CDISK);                ! "DISK XXX WENT OFFLINE"
;   6720  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   6721  1     ENDMSG;


                                        .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  013746  000000G           M$EMS.18:
                                        MOV     CDISK,-(SP)                     ;                       6719
000004  012746  000000G                 MOV     #EBD.18,-(SP)
000010  012746  000002                  MOV     #2,-(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  011732'                 JSR     PC,EMS.RP                       ;                       6720
000024  062706  000006                  ADD     #6,SP                           ;                       6718
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14236
; Maximum stack depth per invocation:  5 words
```

```
;   6722  1     BGNMSG (EMS_21);


                                          .SBTTL  EMS.2  ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.21::JSR     PC,M$EMS.21                     ;                6722
000004  104423                                 TRAP    23
000006  000207                                 RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 14270
; Maximum stack depth per invocation:  2 words


;   6723  2     EMS_RP1 ();                                     ! CONTENTS OF RETURN PACKET
;   6724  1     ENDMSG;


                                          .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  012052'                M$EMS.21:
                                         JSR     PC,EMS.RP1                      ;                6723
000004  000207                                 RTS     PC                              ;                6722

; Routine Size:  3 words,      Routine Base:  $CODE$ + 14300
; Maximum stack depth per invocation:  1 word
```

```
;   6725  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                     ;                                6725
000004  104423                                    TRAP    23
000006  000207                                    RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14306
; Maximum stack depth per invocation:  2 words


;   6726  2     EMS_DBN ();                                                             !ZZZ
;   6727  1     ENDMSG;                                                                 !ZZZ


                                          .SBTTL  M$EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  011004'                   M$EMS.22:
                                                  JSR     PC,EMS.DBN                  ;                                6726
000004  000207                                    RTS     PC                          ;                                6725

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14316
; Maximum stack depth per invocation:  1 word
```

```
;   6728  1     BGNMSG (EMS_24);


                                        .SBTTL  EMS.24 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.24::JSR     PC,M$EMS.24                     ;                       6728
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14324
; Maximum stack depth per invocation:  2 words


;   6729  2     PRINTB (EBD_24, .CDISK);                ! "DISK XXX WENT TO THE AVAILABLE STATE"
;   6730  2     EMS_RP ();                              ! PRINTX RELEVANT RETPKT FIELDS
;   6731  1     ENDMSG;


                                        .SBTTL  M$EMS.24 ERROR MESSAGE SUBROUTINES
000000  013746  000000G            M$EMS.24:
                                        MOV     CDISK,-(SP)                     ;                       6729
000004  012746  000000G                 MOV     #EBD.24,-(SP)
000010  012746  000002                  MOV     #2,(SP)
000014  010600                          MOV     SP,R0                           ; SP,*
000016  104414                          TRAP    14
000020  004737  011732'                 JSR     PC,EMS.RP                       ;                       6730
000024  062706  000006                  ADD     #6,SP                           ;                       6728
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 14334
; Maximum stack depth per invocation:  5 words
```

```
;   6732  1       BGNMSG (EMS_30);


                                         .SBTTL  EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.30::JSR     PC,M$EMS.30                            ;                              6732
000004  104423                           TRAP    23
000006  000207                           RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 14366
; Maximum stack depth per invocation:  2 words


;   6733  2       EMS_ERR ();                                        ! PRINT ALL RELEVANT DATA ON DETECTING AN ERROR
;   6734  1       ENDMSG;


                                         .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  013560'          M$EMS.30:
                                         JSR     PC,EMS.ERR                             ;                              6733
000004  000207                           RTS     PC                                     ;                              6732

; Routine Size:  3 words,       Routine Base:  $CODE$ + 14376
; Maximum stack depth per invocation:  1 word


;   6735  1
;   6736  1       end
;   6737  1
;   6738  0       eludom



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


;                                PSECT SUMMARY
;
;       Psect Name                       Words     Attributes
;        $OWN$                             74        RW ,  D  ,  LCL,  REL,  CON
;        $CODE$                          3202        RO ,  I  ,  LCL,  REL,  CON
;        $PLIT$                            12        RO ,  D  ,  LCL,  REL,  CON



;                          Library Statistics
;
;                                            -------- Symbols --------      Pages       Processing
;       File                                 Total    Loaded   Percent      Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQADO.L16;3            404       294        72          21         00:00.2
```

```
;                                       COMMAND QUALIFIERS

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

; Size:          3032 code + 6503 data words
; Run Time:        02:48.2
; Elapsed Time:    05:13.5
; Lines/CPU Min:    2403
; Lexemes/CPU-Min: 23961
; Memory Used:  677 pages
; Compilation Complete
```

M2


```
;    0001  0     module ZRQAM3 (
;    0002  0
;    0003  0     %title 'RD/RX EXERCISER'
;    0004  0                 ident = 'V01.6',
;    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 'ZRQAD0.L16';                   ! RDRX EXERCISER GLOBAL LIBRARY
;    0014  1
;    0015  1     require 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
;    1506  1
;    1507  1     EQUALS;
;    1508  1
;    1509  1     forward routine                         ! ROUTINES APPEAR IN THIS ORDER
;    1510  1         INIT_TEST : novalue,                !    INDENTATION IMPLIES CALLED SUBROUTINE
;    1511  1             DRIVER_INIT : novalue,
;    1512  1                 CTLR_INIT : novalue,
;    1513  1                     INI_CTLR_DAT : novalue,
;    1514  1                     REG_EXIST,
;    1515  1                     VEC_BR_TEST,
;    1516  1                     INT_GEN,
;    1517  1                     HARD_INIT,
;    1518  1                     INI_RRING : novalue,
;    1519  1                     SET_CTLR_CHAR,
;    1520  1                 UNIT_INIT : novalue,
;    1521  1                     DR_ERR : novalue,
;    1522  1             ACCESS : novalue,
;    1523  1         MULTI_DRIVE : novalue,
;    1524  1             MD_INIT : novalue,
;    1525  1                 INIT_IO_BUFF : novalue,
;    1526  1             FATAL_ERROR : novalue,
;    1527  1             QIO_OK,
;    1528  1                 QIO_OUT,
;    1529  1                 QIO_GEN : novalue,
;    1530  1                     GET_RANDOM : novalue,
;    1531  1                     QIO_UNIT : novalue,
;    1532  1                     QIO_FUNC : novalue,
;    1533  1                        DUP : NOVALUE,                          !ZZZ
;    1534  1                              DUPWRTDBN : NOVALUE,              !ZZZ
;    1535  1                              DUPREDDBN : NOVALUE,              !ZZZ
;    1536  1                                   DUPCOMMAND : NOVALUE,        !ZZZ
;    1537  1                                   DUPIDLE : NOVALUE,           !ZZZ
;    1538  1                     QIO_LBN : novalue,
;    1539  1                     QIO_SIZE : novalue,
;    1540  1                     FILL_BUFF : novalue,
;    1541  1                 PROC_RETPKT : novalue,
;    1542  1                     DIO_RETPKT : NOVALUE,                   !ZZZ
;    1543  1                        DUP_COMPARE : NOVALUE,               !ZZZ
```

```
;    1544  1                      IO_RETPKT : novalue,
;    1545  1                          FSET_UPAR : novalue,
;    1546  1                          HARD_ERROR : novalue,
;    154   1                              ERR_HRD_RTNE : novalue,
;    157   1                              ERR_HRD_RTNE_APT : novalue,
;    1549  1                          UPD_IO_TALLY : novalue,
;    1550  1                              OVF_CHK : novalue,
;    1551  1                              ROUND_OUTPUT : novalue,
;    1552  1                          HOST_WRT_CHK,
;    1553  1      !                       ERR_HRD_RTNE : novalue,
;    1554  1      !                       ERR_HRD_RTNE_APT : novalue,
;    1555  1                          SWEEP : novalue,
;    1556  1                              RPS_REM,
;    1557  1                      DR_RETPKT : novalue,
;    1558  1          AZINTO : L$ISR novalue,
;    1559  1              AZINT : novalue,
;    1560  1      !           FATAL_ERROR : novalue,
;    1561  1                  POLL_CPING : novalue,
;    1562  1                  POLL_RRING : novalue,
;    1563  1                      DUP_RSP  : NOVALUE,
 !ZZZ
;    1564  1                      DISK_RSP : novalue,
;    1565  1                          SEQUEN : novalue,
;    1566  1                              SCAN_ERRLOG : novalue,
;    1567  1                                  ERR_SOFT_RTNE : novalue,
;    1568  1                                  ERR_SOFT_RTNE_APT : novalue,
;    1569  1                                  SOFT_ERROR : novalue,
;    1570  1                          DATAGM : novalue;
;    1571  1      !                       ERR_SOFT_RTNE : novalue,
;    1572  1      !                       ERR_SOFT_RTNE_APT : novalue,
;    1573  1      !                       SOFT_ERROR : novalue;
;    1574  1
;    1575  1      external
;    1576  1          CST : blockvector [MAX_CTLR, CST_LEN, word] field (CST_FIELDS),
;    1577  1                              ! RUN-TIME CONTROLLER STATUS TABLES
;    1578  1          CST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1579  1                              ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;    1580  1          DCT : blockvector [MAX_CTLR, DCT_LEN, word] field (DCT_FIELDS),
;    1581  1                              ! DRIVER CONTROLLER TABLES
;    1582  1          DCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1583  1                              ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;    1584  1          RDRX_ADDR : ref rdrx field (RC_REG),
;    1585  1                              ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
;    1586  1          IRDRX_ADDR : ref rdrx field (RC_REG),
;    1587  1                              ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;    1588  1          BST : BLOCKVECTOR [MAX_UNITS, 2, WORD],                          !ZZZ
;    1589  1                              !BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS     !ZZZ
;    1590  1                              !RANDOM SEEK) MODE                               !ZZZ
;    1591  1          TALLY : vector [MAX_UNITS * TALLY_LEN, word] field (T_FIELDS),
;    1592  1                              ! STATISTICS TABLES
;    1593  1          T_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;    1594  1                              ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;    1595  1          DUPPKT : BLOCK [257, WORD] FIELD (DP_FIELDS),       !BUFFER FOR DUP  ZZZ
;    1596  1                              !INFO FROM RECEIVE AND SEND COMMANDS              ZZZ
```

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

```
;    1650  1          FER_LBN : word,                    ! LBN OF THE "FORCED ERROR" BLOCK
;    1651  1          FER_BC : word,                     ! BYTE COUNT OF THE "FORCED ERROR" BLOCK
;    1652  1          INIT_OCCURED : byte,               ! EXERCISER INITIALIZATION COMPLETE
;    1653  1          ADDR_VECT_OK : byte,               ! FLAG INDICATES IF ADDRESS/VECTOR TEST PASSED
;    1654  1          S_PATTERN : WORD,                  !PATTERN WRITTEN TO DBNS          ZZZ
;    1655  1          S_DUPPKT : WORD,                   !DBN BYTE COUNTER                 ZZZ
;    1656  1          P_INDEX : SIGNED WORD,             !CURRENT MESSAGE PACKET INDEX     ZZZ
;    1657  1          DBM12,
;    1658  1          DBM18,
;    1659  1          DBM19,
;    1660  1          DBM20,
;    1661  1          DBM21,
;    1662  1          DBM22,
;    1663  1          DBM23,
;    1664  1          DBM25,
;    1665  1          DBM26,
;    1666  1          DBM27,
;    1667  1          DBM29,
;    1668  1          DBM108,
;    1669  1          DBM109,
;    1670  1          DBM111,
;    1671  1          DBM112,
;    1672  1          DBM120,
;    1673  1          DBM121,
;    1674  1          EH_0,                              !ZZZ
;    1675  1          EH_1,                              !ZZZ
;    1676  1          EH_2,                              !ZZZ
;    1677  1          EH_3,                              !ZZZ
;    1678  1          EH_4,                              !ZZZ
;    1679  1          EH_5,                              !ZZZ
;    1680  1          EH_6,                              !ZZZ
;    1681  1          EH_7,                              !ZZZ
;    1682  1          EH_8,                              !ZZZ
;    1683  1          EH_9,                              !ZZZ
;    1684  1          EH_10,                             !ZZZ
;    1685  1          EH_12,                             !ZZZ
;    1686  1          EH_13,                             !ZZZ
;    1687  1          MSG_02,
;    1688  1          MSG_03,
;    1689  1          EGS_02,
;    1690  1          EGD_10,
;    1691  1          EGD_11,
;    1692  1          EGD_12,
;    1693  1          EGD_13,
;    1694  1          EGD_14,
;    1695  1          EGD_15,
;    1696  1          EGD_16,
;    1697  1          EGD_17,
;    1698  1          EGD_18,
;    1699  1          EGD_19,
;    1700  1          EGD_20,
;    1701  1          EGD_21,
;    1702  1          EGD_22,
```

```
;    1703  1          EGD_23,
;    1704  1          EGD_24,
;    1705  1          EGH_30,
;    1706  1          DF_MSG,
;    1707  1          HRD_MSG,
;    1708  1          SFT_MSG,
;    1709  1          HRD_SUB,
;    1710  1          CRLF,
;    1711  1          SWP_ERROR : word,                          ! HARD ERROR LIMIT FOR DROPPING UNIT
;    1712  1          SWP_XFER : word,                           ! TRANSFER LIMIT FOR DROPPING UNIT
;    1713  1          SWP_FLAGS : word,                          ! FLAGS (SEE DOCUMENTATION)
;    1714  1          DUPROUND  : WORD,                          !DUP TESTING RATIO                    ZZZ
;    1715  1          SWP_RAT : word,                            ! RD51/52 OPERATION RATIO
;    1716  1          SWP_DPAT : word,                           ! DATA PATTERN NUMBER
;    1717  1          SWP_UCNT : word,                           ! USER DATA PATTERN COUNT
;    1718  1          SWP_TIME : word,                           ! TIME OF DAY
;    1719  1          SWP_UDPAT : vector [MAX_UDP_CNT, word],    ! USER DATA PATTERN
;    1720  1          L$LUN,
;    1721  1          L$UNIT;
;    1722  1
;    1723  1      psect
;    1724  1          own = $GGG$(read, nowrite, execute, local, concatenate);
;    1725  1
;    1726  1      own
;    1727  1          COMM_AREA : blockvector [MAX_CTLR, COMM_LEN, word] field (COM_FIELDS),
;    1728  1                                  ! COMMUNICATIONS AREA BETWEEN HOST AND AZTEC CONTROLLERS
;    1729  1      !!ZZZ    BST : vector [MAX_UNITS, word, signed],
;    1730  1                                  ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
;    1731  1          DPST : vector [MAX_UNITS, byte],           ! DATA PATTERN SEQUENCE TABLE
;    1732  1          MAX_LBN : vector [MAX_UNITS, word],        ! LARGEST LBN ALLOWED
;    1733  1          STORAGE : vector [MAX_UNITS, word],        ! DUMMY STORAGE
;    1734  1          ICOM_ADDR : ref block [COMM_LEN, word] field (COM_FIELDS),
;    1735  1                                  ! ADDRESS OF INTERRUPTING CONTROLLER'S COMMUNICATION AREA
;    1736  1          ICST_ADDR : ref block [CST_LEN, word] field (CST_FIELDS),
;    1737  1                                  ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    1738  1          IDCT_ADDR : ref block [DCT_LEN, word] field (DCT_FIELDS),
;    1739  1                                  ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    1740  1          INT_ADDR : vector [MAX_CTLR] initial (AZINTO %(, AZINT1, AZINT2, AZINT3)%),
;    1741  1                                  ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
;    1742  1      !!ZZZ    RDM_CNT : word initial (RDM_LEN),          ! NUMBER OF RANDOM NUMBERS   \ KEEP
;    1743  1      !!ZZZ    RANDOM : vector [RDM_LEN, word],           ! RANDOM NUMBER TABLE        / TOGETHER
;    1744  1          ICTLR : word,                          ! INTERRUPTING CONTROLLING NUMBER
;    1745  1          MX1 : signed word,                     ! MSCP PKT INDEX FOR FIRST QIO
;    1746  1          MX2 : signed word,                     ! MSCP PKT INDEX FOR SECOND QIO
;    1747  1          MAD1 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1748  1                                  ! ADDRESS OF MSCP PACKET FOR FIRST QIO
;    1749  1          MAD2 : ref block [PKT_LEN, word] field (PKT_FIELDS),
;    1750  1                                  ! ADDRESS OF MSCP PACKET FOR SECOND QIO
;    1751  1          LAST_PKT : blockvector [MAX_CTLR, LAST_PKT_LEN, word] field (LAST_PKT_FIELDS),
;    1752  1                                  ! SAVE AREA FOR INFO ABOUT LAST RESPONSE PACKET
;    1753  1          PAT02 : vector [2] initial (1,                 ! PATTERN 2
;    1754  1                  %o'000000'),
;    1755  1          PAT03 : vector [2] initial (1,                 ! PATTERN 3
```

```
;  1756  1                 %o'177777'),
;  1757  1          PAT04 : vector [2] initial (1,                          ! PATTERN 4
;  1758  1                  %o'105613'),
;  1759  1          PAT05 : vector [2] initial (1,                          ! PATTERN 5
;  1760  1                  %o'031463'),
;  1761  1          PAT06 : vector [2] initial (1,                          ! PATTERN 6
;  1762  1                  %o'030221'),
;  1763  1          PAT07 : vector [17] initial (16,                        ! PATTERN 7
;  1764  1                  %o'000001', %o'000003', %o'000007', %o'000017',
;  1765  1                  %o'000037', %o'000077', %o'000177', %o'000377',
;  1766  1                  %o'000777', %o'001777', %o'003777', %o'007777',
;  1767  1                  %o'017777', %o'037777', %o'077777', %o'177777'),
;  1768  1          PAT08 : vector [17] initial (16,                        ! PATTERN 8
;  1769  1                  %o'177776', %o'177774', %o'177770', %o'177760',
;  1770  1                  %o'177740', %o'177700', %o'177600', %o'177400',
;  1771  1                  %o'177000', %o'176000', %o'174000', %o'170000',
;  1772  1                  %o'160000', %o'140000', %o'100000', %o'000000'),
;  1773  1          PAT09 : vector [17] initial (16,                        ! PATTERN 9
;  1774  1                  rep 3 of (%o'000000'), rep 3 of (%o'177777'),
;  1775  1                  rep 2 of (%o'000000'), rep 2 of (%o'177777'),
;  1776  1                  %o'000000', %o'177777', %o'000000', %o'177777',
;  1777  1                  %o'000000', %o'177777'),
;  1778  1          PAT10 : vector [2] initial (1,                          ! PATTERN 10
;  1779  1                  %o'133331'),
;  1780  1          PAT11 : vector [17] initial (16,                        ! PATTERN 11
;  1781  1                  rep 3 of (%o'052525'), rep 3 of      '25252'),
;  1782  1                  rep 2 of (%o'052525'), rep 2 of (%   5252'),
;  1783  1                  %o'052525', %o'125252', %o'052525', %o'  5252',
;  1784  1                  %o'052525', %o'125252'),
;  1785  1          PAT12 : vector [21] initial (20,                        ! PATTERN 12
;  1786  1                  rep 3 of (%o'026455'), rep 3 of (%o'151322'),
;  1787  1                  rep 2 of (%o'026455'), rep 2 of (%o'151322'),
;  1788  1                  rep 2 of (%o'026455'),
;  1789  1                  %o'151322', %o'026455', %o'151322', %o'026455',
;  1790  1                  %o'151322', %o'026455', %o'151322', %o'026455'),
;  1791  1          PAT13 : vector [2] initial (1,                          ! PATTERN 13
;  1792  1                  %o'066666'),
;  1793  1          PAT14 : vector [17] initial (16,                        ! PATTERN 14
;  1794  1                  %o'000001', %o'000002', %o'000004', %o'000010',
;  1795  1                  %o'000020', %o'000040', %o'000100', %o'000200',
;  1796  1                  %o'000400', %o'001000', %o'002000', %o'004000',
;  1797  1                  %o'010000', %o'020000', %o'040000', %o'100000'),
;  1798  1          PAT15 : vector [17] initial (16,                        ! PATTERN 15
;  1799  1                  %o'177776', %o'177775', %o'177773', %o'177767',
;  1800  1                  %o'177757', %o'177737', %o'177677', %o'177577',
;  1801  1                  %o'177377', %o'176777', %o'175777', %o'173777',
;  1802  1                  %o'167777', %o'157777', %o'137777', %o'077777'),
;  1803  1          PAT16 : vector [17] initial (16,                        ! PATTERN 16
;  1804  1                  rep 3 of (%o'133331'), rep 3 of (%o'155554'),
;  1805  1                  rep 2 of (%o'133331'), rep 2 of (%o'155554'),
;  1806  1                  %o'133331', %o'155554', %o'133331', %o'155554',
;  1807  1                  %o'133331', %o'155554'),
;  1808  1          PAT17 : vector [22] initial (21,                        ! PATTERN 17
```

F3


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

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

```
;    1877  1      %sbttl 'TEST SECTION'
;    1878  1
;    1879  1      !+
;    1880  1      !       THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RDRX EXERCISER.
;    1881  1      !       THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO A NUMBER OF
;    1882  1      !       SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;    1883  1      !-
;    1884  1
;    1885  3      BGNTST;
;    1886  3
;    1887  3      local
;    1888  3          DUMMY_0 : word,
;    1889  3          DUMMY_1 : word;
;    1890  3
;    1891  3      EOP_FLAG = TRUE;                                    ! ASSUME NO UNIT AVAILABLE
;    1892  3      COMPARE_DATA = TRUE;                                ! ALLOW HOST COMAPRES IF ASKED FOR
;    1893  3      DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);          !CLEAR DUP INIT FLAG     ZZZ
;    1894  3      HOE_FLAG = FALSE;                                   ! ASSUME 'HOE' FLAG NOT SET
;    1895  3      FORCED_ERROR = FALSE;                               ! INITIALIZE "FORCED ERROR" FLAG
;    1896  3
;    1897  3      incr I from 0 to PKT_CNT - 1 do                     ! INITIALIZE PACKET AREA
;    1898  4          begin
;    1899  4
;    1900  4          incr J from 0 to PKT_LEN - 1 do
;    1901  4              MSCP_PKT [.I, .J, 0, 16, 0] = 0;
;    1902  4
;    1903  4          MSCP_PKT [.I, RSP_RECEIVED] = FALSE;
;    1904  3          end;
;    1905  3
;    1906  3      incr I from 0 to RP_CNT - 1 do                      ! INITIALIZE RESPONSE SAVE AREA
;    1907  3          incr J from 0 to RP_LEN - 1 do
;    1908  3              RETPKT [.I, .J, 0, 16, 0] = 0;
;    1909  3
;    1910  3      incr I from 0 to EP_CNT do                          ! INITIALIZE ERROR-LOG SAVE AREA
;    1911  4          begin
;    1912  4
;    1913  4          incr J from 0 to EP_LEN - 1 do
;    1914  4              ELOG_PKT [.I, .J, 0, 16, 0] = 0;
;    1915  4
;    1916  4          ELOG_PKT [.I, EL_CONTENTS] = EMPTY;
;    1917  3          end;
;    1918  3
;    1919  4      if BIT_TST (SWP_FLAGS, SWF_CWC)                     ! NO SIMULTANEOUS CNTR/HOST WRIE CHECKS
;    1920  3      then
;    1921  3          SWP_FLAGS = .SWP_FLAGS and (not SWF_HWC);
;    1922  3
;    1923  4      if BIT_TST (SWP_FLAGS, SWF_RDM)                     ! NO SIMULTANEOUS RANDOM/SEQUENTIAL SELECTS
;    1924  3      then
;    1925  3          SWP_FLAGS = .SWP_FLAGS and (not SWF_SEQ);
;    1926  3
;    1927  3      if not .INIT_OCCURED
;    1928  3      then
;    1929  4          begin
```

```
;    1930  4          DRS_START = .FREE_MEM_ADDR + 2 + (..FREE_MEM_ADDR * 2);          ! START OF SUPERVISOR
;    1931  4      !+
;    1932  4      ! CAUTION...     THE FOLLOWING CODE IS 'KLUGED' TO DETERMINE IF THE EXERCISER IS RUNNING UNDER THE
;    1933  4      !                APT SUPERVISOR, AND IS DEPENDENT FOR IT'S SUCCESS ON THE KNOWLEDGE OF THE ACTUAL
;    1934  4      !                APT SUPERVISOR AND THE ADDRESS OF THE MAIL-BOX WITHIN THE SUPERVISOR.
;    1935  4      !-
;    1936  4
;    1937  4          if (..DRS_START eql %o'167') and                              ! APT DRS STARTS WITH A JMP INSTRUCTION
;    1938  5             (not MANUAL)
;    1939  4          then
;    1940  5              begin
;    1941  5              APT_MODE = TRUE;
;    1942  5              MAIL_BOX_TESTNUM = .DRS_START + %o'62' + %o'6';           ! APT MAIL-BOX IS OFFSET AT OCTAL 62 FROM
;    1943  5              MAIL_BOX_SUBTST = .DRS_START + %o'62' + %o'4';            ! BEGINNING OF SUPERVISOR
;    1944  4              end;
;    1945  4
;    1946  4          NEX = FALSE;                                                  ! CHECK IF LINE-CLOCK PRESENT
;    1947  4          CLK_PRESENT = FALSE;                                          !
;    1948  4          SETVEC (4, NEX_TRAP, PRI07);                                  ! SET TRAP CATCHER ADDRESS
;    1949  4          DUMMY_0 = .LINE_CLOCK;                                        ! TRY TO ADDRESS THE CLOCK
;    1950  4          DUMMY_1 = 0;                                                  ! DUMMY INSTRUCTION
;    1951  4          CLRVEC (4);                                                   ! RETURN LOC 4 TO THE SUPERVISOR
;    1952  4
;    1953  4          if not .NEX
;    1954  4          then
;    1955  5              begin
;    1956  5              CLK_PRESENT = TRUE;                                       ! SET FLAG IF CLOCK PRESENT
;    1957  5              CLK_TICKS = 0;                                            ! INITIALIZE THE LINE-CLOCK TICK COUNT
;    1958  5              HOURS = .SWP_TIME / 100;                                  ! TIME OF DAY (HOURS)
;    1959  5              MINUTES = (.SWP_TIME mod 100) + 1;                        ! TIME OF DAY (MINUTES)
;    1960  5
;    1961  5              while .MINUTES gequ 60 do                                 ! NORMALIZE MINUTES
;    1962  6                  begin
;    1963  6                  MINUTES = .MINUTES - 60;
;    1964  6                  HOURS = .HOURS + 1;
;    1965  5                  end;
;    1966  5
;    1967  5              HOURS = .HOURS mod 24;                                    ! NORMALIZE HOURS
;    1968  4              end;
;    1969  4
;    1970  3          end;
;    1971  3
;    1972  3      if .CLK_PRESENT
;    1973  3      then
;    1974  4          begin
;    1975  4          SETVEC (%o'100', TIME, PRI06);                                ! LINE CLOCK VECTOR
;    1976  4          LINE_CLOCK = BIT6;                                            ! START THE CLOCK
;    1977  3          end;
;    1978  3
;    1979  3      RFLAGS (DRS_FLAGS);                                               ! READ DRS FLAGS INTO LOC DRS_FLAGS
;    1980  3
;    1981  3      if BIT_TST (DRS_FLAGS, HOE) eql HOE
;    1982  3      then
```

```
;    1983  3          HOE_FLAG = TRUE;                                          ! SET FLAG IF 'HOE' SET
;    1984  3
;    1985  3      INIT_TEST ();                                                 ! INITIALIZE TEST ENVIRONMENT
;    1986  3
;    1987  3      incr CTLR from 0 to (MAX_CTLR - 1) do                         ! FOR EVERY CONTROLLER
;    1988  3
;    1989  3          if (.CST [.CTLR, STATE] eql ONLINE) and                   ! IF CONTROLLER ONLINE
;    1990  3             (.DCT [.CTLR, STAT] eql ONLINE) and
;    1991  4             (.CST [.CTLR, U_CNT] gequ 0)
;    1992  3          then
;    1993  3              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + 4) by UNIT_SIZE do
;    1994  3
;    1995  3                  if .CST [.CTLR, .OFFSET + OF_DATA, D_STAT] eql ONLINE     ! IF AT LEAST ONE UNIT ALIVE
;    1996  3                  then
;    1997  4                      begin
;    1998  4                      EOP_FLAG = FALSE;                             ! NOT END OF PASS
;    1999  4                      exitloop;
;    2000  3                      end;
;    2001  3
;    2002  3      if not .EOP_FLAG
;    2003  3      then
;    2004  3          MULTI_DRIVE ();                                           ! RUN MULTI-DRIVE TEST
;    2005  1      ENDTST;
```

```
                                .TITLE  ZRQAM3 RD/RX EXERCISER
                                .IDENT  /V01.6/
                                .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   AZINIO
000104                          ICTLR:  .BLKW   1
000106                          MX1:    .BLKW   1
000110                          MX2:    .BLKW   1
000112                          MAD1:   .BLKW   1
000114                          MAD2:   .BLKW   1
000116                          LAST.PKT:
                                        .BLKW   3
000124  000001                  PAT02:  .WORD   1
000126  000000                          .WORD   0
```

```
000130  000001          PAT03:  .WORD   1
000132  177777                  .WORD   -1
000134  000001          PAT04:  .WORD   1
000136  105613                  .WORD   -72165
000140  000001          PAT05:  .WORD   1
000142  031463                  .WORD   31463
000144  000001          PAT06:  .WORD   1
000146  030221                  .WORD   30221
000150  000020          PAT07:  .WORD   20
000152  000001                  .WORD   1
000154  000003                  .WORD   3
000156  000007                  .WORD   7
000160  000017                  .WORD   17
000162  000037                  .WORD   37
000164  000077                  .WORD   77
000166  000177                  .WORD   177
000170  000377                  .WORD   377
000172  000777                  .WORD   777
000174  001777                  .WORD   1777
000176  003777                  .WORD   3777
000200  007777                  .WORD   7777
000202  017777                  .WORD   17777
000204  037777                  .WORD   37777
000206  077777                  .WORD   77777
000210  177777                  .WORD   -1
000212  000020          PAT08:  .WORD   20
000214  177776                  .WORD   -2
000216  177774                  .WORD   -4
000220  177770                  .WORD   -10
000222  177760                  .WORD   -20
000224  177740                  .WORD   -40
000226  177700                  .WORD   -100
000230  177600                  .WORD   -200
000232  177400                  .WORD   -400
000234  177000                  .WORD   -1000
000236  176000                  .WORD   -2000
000240  174000                  .WORD   -4000
000242  170000                  .WORD   -10000
000244  160000                  .WORD   -20000
000246  140000                  .WORD   -40000
000250  100000                  .WORD   -100000
000252  000000                  .WORD   0
000254  000020          PAT09:  .WORD   20
000256  000000                  .WORD   0
000260  000000                  .WORD   0
000262  000000                  .WORD   0
000264  177777                  .WORD   -1
000266  177777                  .WORD   -1
000270  177777                  .WORD   -1
000272  000000                  .WORD   0
000274  000000                  .WORD   0
000276  177777                  .WORD   -1
000300  177777                  .WORD   -1
```

```
000302  000000                          .WORD   0
000304  177777                          .WORD   -1
000306  000000                          .WORD   0
000310  177777                          .WORD   -1
000312  000000                          .WORD   0
000314  177777                          .WORD   -1
000316  000001                  PAT10:  .WORD   1
000320  133331                          .WORD   -44447
000322  000020                  PAT11:  .WORD   20
000324  052525                          .WORD   52525
000326  052525                          .WORD   52525
000330  052525                          .WORD   52525
000332  125252                          .WORD   -52526
000334  125252                          .WORD   -52526
000336  125252                          .WORD   -52526
000340  052525                          .WORD   52525
000342  052525                          .WORD   52525
000344  125252                          .WORD   -52526
000346  125252                          .WORD   -52526
000350  052525                          .WORD   52525
000352  125252                          .WORD   -52526
000354  052525                          .WORD   52525
000356  125252                          .WORD   -52526
000360  052525                          .WORD   52525
000362  125252                          .WORD   -52526
000364  000024                  PAT12:  .WORD   24
000366  026455                          .WORD   26455
000370  026455                          .WORD   26455
000372  026455                          .WORD   26455
000374  151322                          .WORD   -26456
000376  151322                          .WORD   -26456
000400  151322                          .WORD   -26456
000402  026455                          .WORD   26455
000404  026455                          .WORD   26455
000406  151322                          .WORD   -26456
000410  151322                          .WORD   -26456
000412  026455                          .WORD   26455
000414  026455                          .WORD   26455
000416  151322                          .WORD   -26456
000420  026455                          .WORD   26455
000422  151322                          .WORD   -26456
000424  026455                          .WORD   26455
000426  151322                          .WORD   -26456
000430  026455                          .WORD   26455
000432  151322                          .WORD   -26456
000434  026455                          .WORD   26455
000436  000001                  PAT13:  .WORD   1
000440  066666                          .WORD   66666
000442  000020                  PAT14:  .WORD   20
000444  000001                          .WORD   1
000446  000002                          .WORD   2
000450  000004                          .WORD   4
000452  000010                          .WORD   10
```

```
000454  000020                  .WORD   20
000456  000040                  .WORD   40
000460  000100                  .WORD   100
000462  000200                  .WORD   200
000464  000400                  .WORD   400
000466  001000                  .WORD   1000
000470  002000                  .WORD   2000
000472  004000                  .WORD   4000
000474  010000                  .WORD   10000
000476  020000                  .WORD   20000
000500  040000                  .WORD   40000
000502  100000                  .WORD   -100000
000504  000020          PAT15:  .WORD   20
000506  177776                  .WORD   -2
000510  177775                  .WORD   -3
000512  177773                  .WORD   -5
000514  177767                  .WORD   -11
000516  177757                  .WORD   -21
000520  177737                  .WORD   -41
000522  177677                  .WORD   -101
000524  177577                  .WORD   -201
000526  177377                  .WORD   -401
000530  176777                  .WORD   -1001
000532  175777                  .WORD   -2001
000534  173777                  .WORD   -4001
000536  167777                  .WORD   -10001
000540  157777                  .WORD   -20001
000542  137777                  .WORD   -40001
000544  077777                  .WORD   77777
000546  000020          PAT16:  .WORD   20
000550  133331                  .WORD   -44447
000552  133331                  .WORD   -44447
000554  133331                  .WORD   -44447
000556  155554                  .WORD   -22224
000560  155554                  .WORD   -22224
000562  155554                  .WORD   -22224
000564  133331                  .WORD   -44447
000566  133331                  .WORD   -44447
000570  155554                  .WORD   -22224
000572  155554                  .WORD   -22224
000574  133331                  .WORD   -44447
000576  155554                  .WORD   -22224
000600  133331                  .WORD   -44447
000602  155554                  .WORD   -22224
000604  133331                  .WORD   -44447
000606  155554                  .WORD   -22224
000610  000025          PAT17:  .WORD   25
000612  000000                  .WORD   0
000614  106466                  .WORD   -71312
000616  106466                  .WORD   -71312
000620  071311                  .WORD   71311
000622  071311                  .WORD   71311
000624  071311                  .WORD   71311
```

```
000626  106466                          .WORD   -71312
000630  106466                          .WORD   -71312
000632  106466                          .WORD   -71312
000634  106466                          .WORD   -71312
000636  071311                          .WORD   71311
000640  071311                          .WORD   71311
000642  071311                          .WORD   71311
000644  071311                          .WORD   71311
000646  071311                          .WORD   71311
000650  106466                          .WORD   -71312
000652  106466                          .WORD   -71312
000654  106466                          .WORD   -71312
000656  106466                          .WORD   -71312
000660  106466                          .WORD   -71312
000662  106466                          .WORD   -71312
000664  000025                  PAT18:  .WORD   25
000666  106466                          .WORD   -71312
000670  000000                          .WORD   0
000672  071311                          .WORD   71311
000674  106466                          .WORD   -71312
000676  106466                          .WORD   -71312
000700  106466                          .WORD   -71312
000702  071311                          .WORD   71311
000704  071311                          .WORD   71311
000706  071311                          .WORD   71311
000710  071311                          .WORD   71311
000712  106466                          .WORD   -71312
000714  106466                          .WORD   -71312
000716  106466                          .WORD   -71312
000720  106466                          .WORD   -71312
000722  106466                          .WORD   -71312
000724  071311                          .WORD   71311
000726  071311                          .WORD   71311
000730  071311                          .WORD   71311
000732  071311                          .WORD   71311
000734  071311                          .WORD   71311
000736  071311                          .WORD   71311
000740  000025                  PAT19:  .WORD   25
000742  000000                          .WORD   0
000744  134631                          .WORD   -43147
000746  134631                          .WORD   -43147
000750  043146                          .WORD   43146
000752  043146                          .WORD   43146
000754  043146                          .WORD   43146
000756  134631                          .WORD   -43147
000760  134631                          .WORD   -43147
000762  134631                          .WORD   -43147
000764  134631                          .WORD   -43147
000766  043146                          .WORD   43146
000770  043146                          .WORD   43146
000772  043146                          .WORD   43146
000774  043146                          .WORD   43146
000776  043146                          .WORD   43146
```

```
001000  134631                          .WORD   -43147
001002  134631                          .WORD   -43147
001004  134631                          .WORD   -43147
001006  134631                          .WORD   -43147
001010  134631                          .WORD   -43147
001012  134631                          .WORD   -43147
001014  000025                  PAT20:  .WORD   25
001016  134631                          .WORD   -43147
001020  000000                          .WORD   0
001022  043146                          .WORD   43146
001024  134631                          .WORD   -43147
001026  134631                          .WORD   -43147
001030  134631                          .WORD   -43147
001032  043146                          .WORD   43146
001034  043146                          .WORD   43146
001036  043146                          .WORD   43146
001040  043146                          .WORD   43146
001042  134631                          .WORD   -43147
001044  134631                          .WORD   -43147
001046  134631                          .WORD   -43147
001050  134631                          .WORD   -43147
001052  134631                          .WORD   -43147
001054  043146                          .WORD   43146
001056  043146                          .WORD   43146
001060  043146                          .WORD   43146
001062  043146                          .WORD   43146
001064  043146                          .WORD   43146
001066  043146                          .WORD   43146
001070  000001                  PAT21:  .WORD   1
001072  000000                          .WORD   0
001074  000000G                 DPA.TBL:.WORD   RDM.CNT
001076  000124'                         .WORD   PAT02
001100  000130'                         .WORD   PAT03
001102  000134'                         .WORD   PAT04
001104  000140'                         .WORD   PAT05
001106  000144'                         .WORD   PAT06
001110  000150'                         .WORD   PAT07
001112  000212'                         .WORD   PAT08
001114  000254'                         .WORD   PAT09
001116  000316'                         .WORD   PAT10
001120  000322'                         .WORD   PAT11
001122  000364'                         .WORD   PAT12
001124  000436'                         .WORD   PAT13
001126  000442'                         .WORD   PAT14
001130  000504'                         .WORD   PAT15
001132  000546'                         .WORD   PAT16
001134  000610'                         .WORD   PAT17
001136  000664'                         .WORD   PAT18
001140  000740'                         .WORD   PAT19
001142  001014'                         .WORD   PAT20
001144  001070'                         .WORD   PAT21
001146  000000                  BST.CNT:.WORD   0
001150  000000                  BST.DEV:.WORD   0
```

```
001150                          CURRENT.VECTOR:
                                        .BLKW   1
001154                          BRLEVEL:.BLKW   1
001156                          DUOFF:  .BLKW   1
001160                          DRS.START:
                                        .BLKW   1
001162                          APT.MODE:
001162  000                             .BYTE   0
                                        .EVEN
001164                          MAIL.BOX.TESTNUM:
                                        .BLKW   1
001166                          MAIL.BOX.SUBTST:
                                        .BLKW   1
001170                          COMPARE.DATA:
                                        .BLKB   1
                                        .EVEN
001172                          DRS.FLAGS:
                                        .BLKW   1
001174                          RD.MAX.SEQ.CNT:
                                        .BLKW   1
001176                          RX.MAX.SEQ.CNT:
                                        .BLKW   1

                                .GLOBL  CST, CST.ADDR, DCT, DCT.ADDR, RDRX.ADDR
                                .GLOBL  IRDRX.ADDR, BST, TALLY, T.ADDR
                                .GLOBL  DUPPKT, TRK.SGN, RDM.CNT, RANDOM
                                .GLOBL  C.ERR.TBL, MSCP.PKT, IPKT.ADDR
                                .GLOBL  PKT.USE, RETPKT, RP.USE, RP.INDX
                                .GLOBL  RP.ADDR, ELOG.PKT, BUFF.ADDR, BUFF.OWN
                                .GLOBL  IODQ, IODQ.IN, IODQ.OUT, ENTRY.REASON
                                .GLOBL  EOP.FLAG, DUP.FLAGS, CCTLR, CDISK
                                .GLOBL  CUOFF, CTLR.CNT, DUR, QIO, FREE.MEM.ADDR
                                .GLOBL  BYTS.PER.QIO, ST.CODE, SB.CODE
                                .GLOBL  STEP, OF.RC, SA.REG, CMD.TIME
                                .GLOBL  NEX, CRN.LOW, CRN.HIGH, CREDIT.BAL
                                .GLOBL  NEXT.PKT.USE, HOURS, MINUTES, CLK.TICKS
                                .GLOBL  CLK.PRESENT, MOE.FLAG, FORCED.ERROR
                                .GLOBL  FER.LBN, FER.BC, INIT.OCCURED
                                .GLOBL  ADDR.VECT.OK, S.PATTERN, S.DUPPKT
                                .GLOBL  P.INDEX, DBM12, DBM18, DBM19, DBM20
                                .GLOBL  DBM21, DBM22, DBM23, DBM25, DBM26
                                .GLOBL  DBM27, DBM29, DBM108, DBM109, DBM111
                                .GLOBL  DBM112, DBM120, DBM121, EH.0, EH.1
                                .GLOBL  EH.2, EH.3, EH.4, EH.5, EH.6, EH.7
                                .GLOBL  EH.8, EH.9, EH.10, EH.12, EH.13
                                .GLOBL  MSG.02, MSG.03, EGS.02, EGD.10
                                .GLOBL  EGD.11, EGD.12, EGD.13, EGD.14
                                .GLOBL  EGD.15, EGD.16, EGD.17, EGD.18
                                .GLOBL  EGD.19, EGD.20, EGD.21, EGD.22
                                .GLOBL  EGD.23, EGD.24, EGH.30, DF.MSG
                                .GLOBL  HRD.MSG, SFT.MSG, HRD.SUB, CRLF
                                .GLOBL  SWP.ERROR, SWP.XFER, SWP.FLAGS
```

```
                        .GLOBL  DUPROUND, SWP.RAT, SWP.DPAT, SWP.UCNT
                        .GLOBL  SWP.TIME, SWP.UDPAT, L$LUN, L$UNIT
                        .GLOBL  NEX.TRAP, TIME, SET.CPAR, SET.UPAR
                        .GLOBL  OUT.IODQ, IN.IODQ, GET.PKT, PUT.PKT
                        .GLOBL  GET.RETPKT, PUT.RETPKT, GET.IO.BUFF
                        .GLOBL  PUT.IO.BUFF, PUTA.BUFF, SEND, WAIT
                        .GLOBL  MODULAS, DROP.CTLR, DRV.CTLERR
                        .GLOBL  EMS.RP1, EMS.EL, EMS.CMP, EMS.ERR
                        .GLOBL  EMS.10, EMS.12, EMS.13, EMS.14
                        .GLOBL  EMS.18, EMS.21, EMS.22, EMS.24
                        .GLOBL  EMS.30

100000                  BIT15=          -100000
040000                  BIT14=          40000
020000                  BIT13=          20000
010000                  BIT12=          10000
004000                  BIT11=          4000
002000                  BIT10=          2000
001000                  BIT09=          1000
000400                  BIT08=          400
000200                  BIT07=          200
000100                  BIT06=          100
000040                  BIT05=          40
000020                  BIT04=          20
000010                  BIT03=          10
000004                  BIT02=          4
000002                  BIT01=          2
000001                  BIT00=          1
001000                  BIT9=           1000
000400                  BIT8=           400
000200                  BIT7=           200
000100                  BIT6=           100
000040                  BIT5 =          40
000020                  BIT4=           20
000010                  BIT3=           10
000004                  BIT2=           4
000002                  BIT1=           2
000001                  BIT0=           1
000040                  EF.START=       40
000037                  EF.RESTART=     37
000036                  EF.CONTINUE=    36
000035                  EF.NEW=         35
000034                  EF.PWR=         34
000340                  PRI07=          340
000300                  PRI06=          300
000240                  PRI05=          240
000200                  PRI04=          200
000140                  PRI03=          140
000100                  PRI02=          100
000040                  PRI01=          40
000000                  PRI00=          0
000004                  EVL.=           4
```

```
        000010                  LOT==           10
        000020                  ADR==           20
        000040                  IDU==           40
        000100                  ISR==           100
        000200                  UAM==           200
        000400                  BOE==           400
        001000                  PNI==           1000
        002000                  PRI==           2000
        004000                  IXE==           4000
        010000                  IBE==           10000
        020000                  IER==           20000
        040000                  LOE==           40000
        100000                  HOE==           -100000

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

000000  004137  000000G           $T1:    JSR     R1,$SAVE3                       ;                       1876
000004  112737  000001  000000G           MOVB    #1,ECP.FLAG                     ;                       1891
000012  112737  000001  001170'           MOVB    #1,COMPARE.DATA                 ;                       1892
000020  042737  000002  000000G           BIC     #2,DUP.FLAGS                    ;                       1893
000026  105037  000000G                   CLRB    HOE.FLAG                        ;                       1894
000032  105037  000000G                   CLRB    FORCED.ERROR                    ;                       1895
000036  005002                            CLR     R2                              ; I                     1897
000040  010246                  1$:       MOV     R2,-(SP)                        ; I,*                   1901
000042  012746  000043                    MOV     #43,-(SP)
000046  004737  000000G                   JSR     PC,BL$MUL
000052  005001                            CLR     R1                              ; J                     1900
000054  010003                  2$:       MOV     R0,R3                           ;                       1901
000056  060103                            ADD     R1,R3                           ; J,*
000060  006303                            ASL     R3
000062  005063  000000G                   CLR     MSCP.PKT(R3)
000066  005201                            INC     R1                              ; J                     1900
000070  020127  000042                    CMP     R1,#42                          ; J,*
000074  003767                            BLE     2$
000076  010216                            MOV     R2,(SP)                         ; I,*                   1903
000100  012746  000106                    MOV     #106,-(SP)
000104  004737  000000G                   JSR     PC,BL$MUL
000110  105060  000005G                   CLRB    MSCP.PKT+5(R0)
000114  062706  000006                    ADD     #6,SP                           ;                       1898
000120  005202                            INC     R2                              ; I                     1899
000122  020227  000013                    CMP     R2,#13                          ; I,*
000126  003744                            BLE     1$
000130  005002                            CLR     R2                              ; I                     1906
000132  005001                  3$:       CLR     R1                              ; J                     1907
000134  010200                  4$:       MOV     R2,R0                           ; I,*                   1908
000136  060100                            ADD     R1,R0                           ; J,*
000140  006300                            ASL     R0
000142  005060  000000G                   CLR     RETPKT(R0)
000146  005201                            INC     R1                              ; J                     1907
000150  020127  000025                    CMP     R1,#25                          ; J,*
000154  003767                            BLE     4$
```

```
000156  062702  000026                  ADD     #26,R2                  ; *,I                   1906
000162  020227  000232                  CMP     R2,#232                 ; I,*
000166  003761                          BLE     3$
000170  005002                          CLR     R2                      ; I                     1910
000172  010246                  5$:     MOV     R2,-(SP)                ; I,*                   1914
000174  012746  000041                  MOV     #41,-(SP)
000200  004737  000000G                 JSR     PC,BL$MUL
000204  005001                          CLR     R1                      ; J                     1913
000206  010003                  6$:     MOV     R0,R3                   ;                       1914
000210  060103                          ADD     R1,R3                   ; J,*
000212  006303                          ASL     R3
000214  005063  000000G                 CLR     ELOG.PKT(R3)
000220  005201                          INC     R1                      ; J                     1913
000222  020127  000040                  CMP     R1,#40                  ; J,*
000226  003767                          BLE     6$
000230  010216                          MOV     R2,(SP)                 ; I,*                   1916
000232  012746  000102                  MOV     #102,-(SP)
000236  004737  000000G                 JSR     PC,BL$MUL
000242  105060  000001G                 CLRB    ELOG.PKT+1(R0)
000246  062706  000006                  ADD     #6,SP                   ;                       1911
000252  005202                          INC     R2                      ; I                     1910
000254  020227  000014                  CMP     R2,#14                  ; I,*
000260  003744                          BLE     5$
000262  032737  000020  000000G         BIT     #20,SWP.FLAGS           ;                       1919
000270  001403                          BEQ     7$
000272  042737  000040  000000G         BIC     #40,SWP.FLAGS           ;                       1921
000300  032737  000002  000000G 7$:     BIT     #2,SWP.FLAGS            ;                       1923
000306  001403                          BEQ     8$
000310  042737  001000  000000G         BIC     #1000,SWP.FLAGS         ;                       1925
000316  132737  000001  000000G 8$:     BITB    #1,INIT.OCCURED         ;                       1927
000324  001145                          BNE     15$
000326  017700  000000G                 MOV     @FREE.MEM.ADDR,R0       ;                       1930
000332  006300                          ASL     R0
000334  063700  000000G                 ADD     FREE.MEM.ADDR,R0
000340  010037  001160'                 MOV     R0,DRS.START
000344  062737  000002  001160'         ADD     #2,DRS.START
000352  027727  001160'  000167         CMP     @DRS.START,#167         ;                       1937
000360  001021                          BNE     9$
000362  104450                          TRAP    50                      ;                       1938
000364  103417                          BCS     9$
000366  112737  000001  001162'         MOVB    #1,ABT.MODE             ;                       1941
000374  013737  001160'  001164'         MOV     DRS.START,MAIL.BOX.TESTNUM ;                    1942
000402  062737  000070  001164'         ADD     #70,MAIL.BOX.TESTNUM
000410  013737  001160'  001166'         MOV     DRS.START,MAIL.BOX.SUBTST ;                     1943
000416  062737  000066  001166'         ADD     #66,MAIL.BOX.SUBTST
000424  005037  000000G         9$:     CLR     NEXT                    ;                       1946
000430  105037  000000G                 CLRB    CLK.PRESENT             ;                       1947
000434  012746  000340                  MOV     #340,-(SP)              ;                       1949
000440  012746  000000G                 MOV     #SPEX.TRAP,-(SP)
000444  012746  000004                  MOV     #4,-(SP)
000450  012746  000003                  MOV     #3,-(SP)
000454  104457                          TRAP    57
000456  012700  000004                  MOV     #4,R0                   ;                       1951
```

```
000462  104436                          TRAP    36
000464  032737  000001  000000G         BIT     #1,NEX                  ;                       1953
000472  001060                          BNE     1??
000474  112737  000001  000000G         MOVB    #1,CLK.PRESENT          ;                       1956
000502  005037  000000G                 CLR     CLK.TICKS               ;                       1957
000506  013716  000000G                 MOV     SWP.TIME,(SP)           ;                       1958
000512  012746  000144                  MOV     #144,-(SP)
000516  004737  000000G                 JSR     PC,@#L$DIV
000522  110037  000000G                 MOVB    R0,HOURS
000526  013716  000000G                 MOV     SWP.TIME,(SP)           ;                       1959
000532  012746  000144                  MOV     #144,-(SP)
000536  004737  000000G                 JSR     PC,@#L$MOD
000542  010001                          MOV     R0,R1
000544  005201                          INC     R1
000546  110137  000000G                 MOVB    R1,MINUTES
000552  123727  000000G  000074  10$:   CMPB    MINUTES,#74             ;                       1961
000560  103412                          BLO     11$
000562  005000                          CLR     R0                      ;                       1963
000564  153700  000000G                 BISB    MINUTES,R0
000570  162700  000074                  SUB     #74,R0
000574  110037  000000G                 MOVB    R0,MINUTES
000600  105237  000000G                 INCB    HOURS                   ;                       1964
000604  000762                          BR      10$                     ;                       1961
000606  005016                  11$:    CLR     (SP)                    ;                       1967
000610  113716  000000G                 MOVB    HOURS,(SP)
000614  012746  000030                  MOV     #30,-(SP)
000620  004737  000000G                 JSR     PC,@#L$MOD
000624  110037  000000G                 MOVB    R0,HOURS
000630  062706  000006                  ADD     #6,SP                   ;                       1955
000634  062706  000010          12$:    ADD     #10,SP                  ;                       1929
000640  132737  000001  000000G 13$:    BITB    #1,CLK.PRESENT          ;                       1972
000646  001416                          BEQ     14$
000650  012746  000300                  MOV     #300,-(SP)              ;                       1975
000654  012746  000000G                 MOV     #TT...,-(SP)
000660  012746  000100                  MOV     #100,-(SP)
000664  012746  000003                  MOV     #3,-(SP)
000670  104437                          TRAP    37
000672  012737  000100  177546          MOV     #100,@#177546           ;                       1976
000700  062706  000010                  ADD     #10,SP                  ;                       1974
000704  104421                  14$:    TRAP    21                      ;                       1979
000706  010037  001172'                 MOV     R0,DRS.FLAGS
000712  042700  077777                  BIC     #77777,R0               ;                       1981
000716  020027  100000                  CMP     R0,#-100000
000722  001003                          BNE     15$
000724  012700  000001                  MOV     #1,R0
000730  000401                          BR      16$
000732  005000                  15$:    CLR     R0
000734  020027  100000          16$:    CMP     R0,#-100000
000740  001003                          BNE     17$
000742  112737  000001  000000G         MOVB    #1,TOE.FLAG             ;                       1983
000750  004737  000000V         17$:    JSR     PC,INIT.TEST            ;                       1985
000754  005002                          CLR     R2                      ; CTLR                  1987
000756  010246                  18$:    MOV     R2,-(SP)                ; CTLR,*                1989
```

```
000760  012746  000126                        MOV     #126,-(SP)
000764  004737  000000G                       JSR     PC,BL$MUL
000770  022626                                CMP     (SP)+,(SP)+
000772  005760  000002G                       TST     CST+2(R0)
000776  100040                                BPL     22$
001000  010246                                MOV     R2,-(SP)                ; CTLR,*                        1990
001002  012746  000022                        MOV     #22,-(SP)
001006  004737  000000G                       JSR     PC,BL$MUL
001012  022626                                CMP     (SP)+,(SP)+
001014  005760  000000G                       TST     NCT(R0)
001020  100027                                BPL     22$
001022  010246                                MOV     R2,-(SP)                ; CTLR,*                        1995
001024  012746  000053                        MOV     #53,-(SP)
001030  004737  000000G                       JSR     PC,BL$MUL
001034  012701  000003                        MOV     #3,R1                   ; *,OFFSET                      1993
001040  010003                      19$:      MOV     R0,R3                   ;                               1995
001042  060103                                ADD     R1,R3                   ; OFFSET,*
001044  006303                                ASL     R3
001046  032763  020000  000000G               BIT     #20000,CST(R3)
001054  001403                                BEQ     20$
001056  105037  000000G                       CLRB    EOP.FLAG                ;                               1998
001062  000405                                BR      21$                     ;                               1997
001064  062701  000012                20$:     ADD     #12,R1                  ; *,OFFSET                      1993
001070  020127  000042                        CMP     R1,#42                  ; OFFSET,*
001074  003761                                BLE     19$
001076  022626                      21$:      CMP     (SP)+,(SP)+
001100  005202                      22$:      INC     R2                      ; CTLR                          1987
001102  000243                                .WORD CLV!CLC
001104  003724                                BLE     18$
001106  132737  000001  000000G               BITB    #1,EOP.FLAG             ;                               2002
001114  001002                                BNE     23$
001116  004737  000000V                       JSR     PC,MULTI.DRIVE          ;                               2004
001122  000207                      23$:      RTS     PC                      ;                               1876

; Routine Size:  298 words,      Routine Base:  $CODE$ + 0000
; Maximum stack depth per invocation:  12 words



                                              .SBTTL  T1 TEST SECTION
000000  004737  000000'                T1::
000000                                  1$:     JSR     PC,$T1                  ;                               2004
000004  104466                                  TRAP    66
000006  006000                                  ROR     R0
000010  103773                                  BLO     1$
000012  000207                                  RTS     PC

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

```
;    2006  1     %sbttl 'INITIALIZATION TEST ROUTINES'
;    2007  1
;    2008  1     GLOBAL routine INIT_TEST : novalue =
;    2009  1
;    2010  1     !+
;    2011  1     !       THE INITIALIZATION TEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
;    2012  1     !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
;    2013  1     !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE TEST
;    2014  1     !       OR THE DM EXERCISER.
;    2015  1     !
;    2016  1     !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
;    2017  1     !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
;    2018  1     !       MARKED OFFLINE IN THEIR DCT AND CST.  FOR THOSE DEVICES WHICH SURVIVE
;    2019  1     !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
;    2020  1     !       EACH DISK VIA ROUTINE "ACCESS". THE INITIALIZATION TEST IS DEEMED A
;    2021  1     !       SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH DISK CAN BE ACCESSED.
;    2022  1     !-
;    2023  1
;    2024  2         begin
;    2025  2         DRIVER_INIT ();                             ! INIT DRIVER DATA AND DEVICES
;    2026  2
;    2027  2         incr CTLR from 0 to (MAX_CTLR - 1) do       ! FOR EACH CONTROLLER
;    2028  3             begin
;    2029  3             SET_CPAR (.CTLR);                       ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;    2030  3
;    2031  3             if .CST_ADDR [STATF] eql ONLINE         ! IF CONTROLLER IS STILL ALIVE
;    2032  3             then                                    ! FOR EACH DISK
;    2033  3
;    2034  3                 incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    2035  3
;    2036  3                 if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    2037  3                    (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    2038  4                    (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2039  3                 then
;    2040  4                     begin
;    2041  4                     SET_UPAR (.OFFSET);             ! SET UP UNIT-RELATED DATA ITEMS
;    2042  5                     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)          !ZZZ
;    2043  4                        THEN ACCESS ();                                 !ZZZ
;    2044  4                                                !SKIP IF DUP CAUSED INIT ZZZ
;    2045  4
;    2046  3                     end;                            ! IF UNIT IS PRESENT AND ONLINE
;    2047  3
;    2048  2             end;                                    ! CONTROLLER LOOP
;    2049  2
;    2050  1         end;                                        ! ROUTINE INIT_TEST


                                .SBTTL  INIT_TEST INITIALIZATION TEST ROUTINES
000000  004137  000000G      INIT_TEST::
                                JSR     R1,$SAVE2                       ;                       2008
000004  004737  000000V                 JSR     PC,DRIVER_INIT          ;                       2025
000010  005002                          CLR     R2                      ; CTLR                  2027
000012  010246              1$:         MOV     R2,-(SP)                ; CTLR,*                2029
```

```
000014  004737  000000G                        JSR     PC,SET.CFAR
000020  013700  000000G                        MOV     CST.ADDR,R0             ;                       2031
000024  005760  000002                         TST     2(R0)
000030  100035                                 BPL     4$
000032  012701  000003                         MOV     #3,R1                   ; *,OFFSET              2034
000036  010100                        2$:      MOV     R1,R0                   ; OFFSET,*              2036
000040  006300                                 ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  032710  040000                         BIT     #40000,(R0)
000052  001417                                 BEQ     3$
000054  032710  020000                         BIT     #20000,(R0)             ;                       2037
000060  001414                                 BEQ     3$
000062  032710  010000                         BIT     #10000,(R0)             ;                       2038
000066  001011                                 BNE     3$
000070  010116                                 MOV     R1,(SP)                 ; OFFSET,*              2041
000072  004737  000000G                        JSR     PC,SET.UPAR
000076  032737  000002  000000G                BIT     #2,DDB$FLAGS            ;                       2042
000104  001002                                 BNE     3$
000106  004737  000000V                        JSR     PC,ACCESS               ;                       2043
000112  062701  000012                3$:      ADD     #12,R1                  ; *,OFFSET              2034
000116  020127  000041                         CMP     R1,#41                  ; OFFSET,*
000122  003745                                 BLE     2$
000124  005726                        4$:      TST     (SP)+                   ;                       2028
000126  005202                                 INC     R2                      ; CTLR                  2027
000130  000243                                 .WORD CLV!CLC

000132  003727                                 BLE     1$
000134  000207                                 RTS     PC                      ;                       2008
```

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

```
;    2051  1     GLOBAL routine DRIVER_INIT : novalue =
;    2052  1
;    2053  1     !+
;    2054  1     !       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
;    2055  1     !       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
;    2056  1     !       DRIVER DATA, AND TO BRING EACH RDRX CONTROLLER AND UNIT (DISK)
;    2057  1     !       ONLINE.
;    2058  1     !-
;    2059  1
;    2060  2         begin
;    2061  2
;    2062  2         local
;    2063  2             PKT_ADDR;
;    2064  2
;    2065  2         PKT_ADDR = MSCP_PKT + 10;                                   ! ADDR (TEXT + 0) OF 1ST MSCP PKT
;    2066  2         NEXT_PKT_USE = 0;                                           ! NEXT PACKET TO ALLOCATE
;    2067  2
;    2068  2         incr COUNT from 0 to (PKT_CNT - 1) do                       ! FOR EACH MSCP PACKET
;    2069  3             begin
;    2070  3             PKT_USE [.COUNT] = -1;                                  ! MARK PACKET FREE
;    2071  3             MSCP_PKT [.COUNT, PKT_LO] = .PKT_ADDR;                  ! LOAD ADDR INTO BUFFER DESCRIPTOR
;    2072  3             MSCP_PKT [.COUNT, PKT_HI] = 0;
;    2073  3             MSCP_PKT [.COUNT, CONNID] = CID_DISK;                   ! SET CONNECTION ID TO MSCP ID
;    2074  3             PKT_ADDR = .PKT_ADDR + (PKT_LEN * 2);                   ! ADVANCE ADDR TO NEXT PACKET
;    2075  2             end;
;    2076  2
;    2077  2         incr CTLR from 0 to (MAX_CTLR - 1) do                       ! FOR EACH CONTROLLER
;    2078  2
;    2079  2             if .CST [.CTLR, IP_ADDR] neqa 0                         ! IF CONTROLLER IS PRESENT
;    2080  2             then
;    2081  3             begin
;    2082  3             SET_CPAR (.CTLR);                                       ! CURRENT CONTROLLER PARAMETERS
;    2083  3             CURRENT_VECTOR = .CST_ADDR [VEC_ADDR];                  ! CURRENT CONTROLLER'S VECTOR
;    2084  3             BRLEVEL = .CST_ADDR [BR_LEV] + 5;                       ! SET CURRENT CONTROLLER'S BR LEVEL
;    2085  3             CTLR_INIT ();                                           ! INIT DEVICE AND CTLR DATA
;    2086  3
;    2087  3             if .DCT_ADDR [STAT] eql ONLINE                          ! IF CONTROLLER IS STILL ALIVE
;    2088  3             then                                                    ! FOR EACH DISK UNIT
;    2089  3
;    2090  3                 incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    2091  3
;    2092  3                     if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and  ! IF UNIT EXISTS
;    2093  4                        (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    2094  3                     then
;    2095  4                         begin
;    2096  4                         CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_0] = %o'40';      ! BLANK DEVICE NAME
;    2097  4                         CST_ADDR [.OFFSET + OF_NAME_0, D_NAME_1] = %o'40';      !
;    2098  4                         CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_2] = %o'40';      !
;    2099  4                         CST_ADDR [.OFFSET + OF_NAME_2, D_NAME_3] = %o'40';      !
;    2100  4                         SET_UPAR (.OFFSET);                                     ! SET UP UNIT RELATED DATA ITEMS
;    2101  4                         UNIT_INIT ();                                           ! BRING UNIT ONLINE
;    2102  3                         end;                                                    ! IF UNIT EXISTS
;    2103  3
```

```
;    2104  2                end;                                        ! IF CONTROLLER IS PRESENT
;    2105  2
;    2106  1            end;                                            ! ROUTINE DRIVER_INIT


                                        .SBTTL  DRIVER.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G           DRIVER.INIT::
                                        JSR     R1,$SAVE3               ;                                   2051
000004  012702  000012G                 MOV     #MSCP.PKT+12,R2         ; *,PKT.ADDR                        2065
000010  105037  000000G                 CLRB    NEXT.PKT.USE            ;                                   2066
000014  005001                          CLR     R1                      ; COUNT                             2068
000016  112761  000377  000000G 1$:     MOVB    #377,PKT.USE(R1)        ; *,*(COUNT)                        2070
000024  010146                          MOV     R1,-(SP)                ; COUNT,*                           2071
000026  012746  000106                 MOV     #106,-(SP)
000032  004737  000000                 JSR     PC,BL$MUL
000036  010260  0000??                 MOV     R2,MSCP.PKT(R0)         ; PKT.ADDR,*
000042  005060  0000??                 CLR     MSCP.PKT+2(R0)          ;                                   2072
000046  105060  000011G                CLRB    MSCP.PKT+11(R0)         ;                                   2073
000052  062702  000106                 ADD     #106,R2                 ; *,PKT.ADDR                        2074
000056  022626                        CMP     (SP)+,(SP)+             ;                                   2069
000060  005201                        INC     R1                      ; COUNT                             2068
000062  020127  000013                CMP     R1,#13                  ; COUNT,*
000066  003753                        BLE     1$
000070  005003                        CLR     R3                      ; CTLR                              2077
000072  010346                  2$:   MOV     R3,-(SP)                ; CTLR,*                            2079
000074  012746  000126                 MOV     #126,-(SP)
000100  004737  000000G               JSR     PC,BL$MUL
000104  022626                        CMP     (SP)+,(SP)+
000106  005760  000000G               TST     CST(R0)
000112  001503                        BEQ     6$
000114  010346                        MOV     R3,-(SP)                ; CTLR,*                            2082
000116  004737  000000G               JSR     PC,SET.CPAR
000122  013700  000000G               MOV     CST.ADDR,R0             ;                                   2083
000126  016037  000002  001152'       MOV     2(R0),CURRENT.VECTOR
000134  042737  177000  001152'       BIC     #177000,CURRENT.VECTOR
000142  005016                        CLR     (SP)                    ;                                   2084
000144  116016  000004                MOVB    4(R0),(SP)
000150  012746  000005                MOV     #5, (SP)
000154  004737  000000G               JSR     PC,BL$SHF
000160  010037  001154'               MOV     R0,BRLEVEL
000164  004737  000000V               JSR     PC,CTLR.INIT            ;                                   2085
000170  005777  000000G               TST     @DCT.ADDR               ;                                   2087
000174  100051                        BPL     5$
000176  012701  000003                MOV     #3,R1                   ; *,OFFSET                          2090
000202  013702  000000G         3$:   MOV     CST.ADDR,R2             ;                                   2092
000206  010100                        MOV     R1,R0                   ; OFFSET,*
000210  006300                        ASL     R0
000212  060200                        ADD     R2,R0
000214  032710  040000                BIT     #40000,(R0)
000220  001432                        BEQ     4$
000222  032710  010000                BIT     #10000,(R0)             ;                                   2095
000226  001027                        BNE     4$
000230  010100                        MOV     R1,R0                   ; OFFSET,*                          2096
```

```
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)                      ;                               2097
000252  010100                          MOV     R1,R0                           ; OFFSET,*                      2098
000254  006300                          ASL     R0
000256  060200                          ADD     R2,R0
000260  112760  000040  000014          MOVB    #40,14(R0)
000266  112760  000040  000015          MOVB    #40,15(R0)                      ;                               2099
000274  010116                          MOV     R1,(SP)                         ; OFFSET,*                      2100
000276  004737  0000005                 JSR     PC,SET.UPAR
000302  004737  000000V                 JSR     PC,UNIT.INIT                    ;                               2101
000306  062701  000012          4$:     ADD     #12,R1                          ; *,OFFSET                      2090
000312  020127  000041                  CMP     R1,#41                          ; OFFSET,*
000316  003731                          BLE     3$
000320  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               2081
000322  005203                  6$:     INC     R3                              ; CTLR                          2077
000324  000243                          .WORD CLV!CLC

000326  003661                          BLE     2$
000330  000207                          RTS     PC                              ;                               2051

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

```
;    2107  1     GLOBAL routine CTLR_INIT : novalue =
;    2108  1
;    2109  1     !+
;    2110  1     !       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;    2111  1     !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RDRX ONLINE
;    2112  1     !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;    2113  1     !
;    2114  1     !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;    2115  1     !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS,
;    2116  .     !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;    2117  1     !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;    2118  1     !               ADDRESS AND INTERRUPT REQUEST LEVEL,
;    2119  1     !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;    2120  1     !
;    2121  1     !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;    2122  1     !       DEVICE ARE DROPPED.
;    2123  1     !-
;    2124  1
;    2125  2         begin
;    2126  2
;    2127  2         local
;    2128  2             RESULT : byte;
;    2129  2
;    2130  2         INI_CTLR_DAT ();                                        ! INITIALIZE CONTROLLER DATA
;    2131  2         SETVEC (.CURRENT_VECTOR, .INT_ADDR [.CCTLR], PRIO4);    ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;    2132  2         DCT_ADDR [IG_INT] = TRUE;                               ! SET "IGNORE INTERRUPT" BIT
;    2133  2         L$LUN = .CST_ADDR [OF_UN + OF_DATA, D_UNIT];            ! GET FIRST UNIT NUMBER OF CONTROLLER
;    2134  2                                                                 ! (USED BY DRS FOR DEVICE-FATAL CTLR ERRORS)
;    2135  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2136  2     THEN                                        !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2137  2
;    2138  2         if REG_EXIST () eql FAILURE                             ! REGISTER EXISTENCE TEST
;    2139  2         then
;    2140  3             begin
;    2141  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2142  3             return;
;    2143  2             end;
;    2144  2
;    2145  3     IF SWP_DINT NEQ (.DUP_FLAGS AND SWP_DINT)                   !IF DUP  ZZZ
;    2146  2     THEN                                        !CAUSED INIT, SKIP THIS CODE   ZZZ
;    2147  2
;    2148  2         if VEC_BR_TEST () eql FAILURE                           ! VECTOR ADDR AND BR LEVEL TEST
;    2149  2         then
;    2150  3             begin
;    2151  3             DROP_CTLR (.CCTLR, DU_INIT);                        ! DROP ALL CONTROLLER'S UNITS
;    2152  3             retur .
;    2153  2             end;
;    2154  2
;    2155  2         RESULT = HARD_INIT ();                                  ! ATTEMPT HARD DEVICE INIT
;    2156  2         DCT_ADDR [IG_INT] = FALSE;                              ! CLAER "IGNORE INTERRUPT" BIT
;    2157  2
;    2158  2         if .RESULT eql SUCCESS                                  ! IF HARD INIT WAS SUCCESSFUL
;    2159  2         then
```

```
;  2160  3          begin
;  2161  3          ADDR_VECT_OK = TRUE;                    ! ADDRESS/VECTOR TEST PASSED
;  2162  3          INI_RRING ();                           ! INITIALIZE RESPONSE RING
;  2163  3          WRT_RDRX (RCSA, RC_ALL, SA_GO);         ! SET "GO" BIT (START CTLR POLLING)
;  2164  3
;  2165  3          if SET_CTLR_CHAR () eql SUCCESS         ! SET CONTROLLER CHARACTERISTICS
;  2166  3          then
;  2167  4              begin
;  2168  4              DCT_ADDR [STAT] = ONLINE;           ! MARK CONTROLLER ONLINE IN "DRIVER"
;  2169  4              CST_ADDR [STATE] = ONLINE;          ! MARK CONTROLLER ONLINE IN "PROGRAM"
;  2170  3              end;
;  2171  3          end
;  2172  3
;  2173  2      else                                        ! HARD INIT FAILED
;  2174  3          begin
;  2175  3          DROP_CTLR (.CCTLR, DU_INIT);            ! DROP ALL CONTROLLER'S UNITS
;  2176  2          end;
;  2177  2
;  2178  1      end;                                        ! ROUTINE CTLR_INIT

                                        .SBTTL  CTLR.INIT INITIALIZATION TEST ROUTINES
000000  010146                 CTLR.INIT::
                                MOV     R1,-(SP)                ;2107
000002  004737  000000V         JSR     PC,INI.CTLR.DAT         ;2130
000006  012746  000200          MOV     #200,-(SP)              ;2131
000012  013700  000000G         MOV     CCTLR,R0
000016  006300                  ASL     R0
000020  016046  000102'         MOV     INT.ADDR(R0),-(SP)
000024  013746  001152'         MOV     CURRENT.VECTOR,-(SP)
000030  012746  000003          MOV     #3,-(SP)
000034  104437                  TRAP    37
000036  052777  040000  000000G BIS     #40000,@DCT.ADDR        ;2132
000044  013700  000000G         MOV     CST.ADDR,R0             ;2133
000050  016001  000006          MOV     6(R0),R1
000054  000301                  SWAB    R1
000056  042701  177760          BIC     #177760,R1
000062  010137  000000G         MOV     R1,@$LUN
000066  032737  000002  000000G BIT     #2,DUP.FLAGS            ;2135
000074  001025                  BNE     2$
000076  004737  000000V         JSR     PC,REG.EXIST            ;2138
000102  005700                  TST     R0
000104  001410                  BEQ     1$                      ;2141
000106  032737  000002  000000G BIT     #2,DUP.FLAGS            ;2145
000114  001015                  BNE     2$
000116  004737  000000V         JSR     PC,VEC.BR.TEST          ;2148
000122  005700                  TST     R0
000124  001011                  BNE     2$
000126  013716  000000G  1$:    MOV     CCTLR,(SP)              ;2151
000132  012746  000002          MOV     #2,(SP)
000136  004737  000000G         JSR     PC,DROP.CTLR
000142  062706  000012          ADD     #12,SP                  ;2155
000146  000453                  BR      5$                      ;2150
```

```
000150  004737  000000V                2$:     JSR     PC,HARD.INIT                    ;                               2155
000154  110001                                  MOVB    R0,R1                           ; #,RESULT
000156  042777  040000  000000G                BIC     #40000,@DCT.ADDR                ;                               2156
000164  120127  000001                          CMPB    R1,#1                           ; RESULT,#                      2158
000170  001031                                  BNE     3$
000172  112737  000001  000000G                MOVB    #1,ADDR.VECT.OK                 ;                               2161
000200  004737  000000V                        JSR     PC,IN1.RRING                    ;                               2162
000204  012701  000001                          MOV     #1,R1                           ; #,RC.REG                      2163
000210  013700  000000G                        MOV     RDRX.ADDR,R0
000214  010160  000002                          MOV     R1,2(R0)                        ; RC.REG,#
000220  004737  000000V                        JSR     PC,SET.CTLR.CHAR                ;                               2165
000224  020027  000001                          CMP     R0,#1
000230  001020                                  BNE     4$
000232  052777  100000  000000G                BIS     #100000,@DCT.ADDR               ;                               2168
000240  013700  000000G                        MOV     CST.ADDR,R0                     ;                               2169
000244  052760  100000  000002                BIS     #100000,2(R0)
000252  000407                                  BR      4$                              ;                               2158
000254  013716  000000G                3$:     MOV     CCTLR,(SP)                      ;                               2175
000260  012746  000002                          MOV     #2,-(SP)
000264  004737  000000G                        JSR     PC,DROP.CTLR
000270  005726                                  TST     (SP)+                           ;                               2174
000272  062706  000010                4$:     ADD     #10,SP                          ;                               2125
000276  012601                        5$:     MOV     (SP)+,R1                        ;                               2107
000300  000207                                RTS     PC

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

```
;    2179  1     GLOBAL routine INI_CTLR_DAT : novalue =
;    2180  1
;    2181  1     !+
;    2182  1     !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER-RELATED
;    2183  1     !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;    2184  1     !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;    2185  1     !
;    2186  1     !       IMPLICIT INPUTS:
;    2187  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2188  1     !               DCT_ADDR - ADDRESS OF CURENT CONTROLLER'S DCT
;    2189  1     !-
;    2190  1
;    2191  2         begin
;    2192  2         DCT_ADDR [WORD0] = 0;                                          ! CLEAR FIRST DCT WORD
;    2193  2         DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;    2194  2         DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4);! LAST SLOT IN RESPONSE RING
;    2195  2         DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                    ! START OF COMMAND RING
;    2196  2         DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4);! LAST SLOT IN COMMAND RING
;    2197  2         DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG];                       ! FIRST RRING SLOT TO POLL
;    2198  2         DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;    2199  1         end;


                                        .SBTTL  INI.CTLR.DAT INITIALIZATION TEST ROUTINES
000000  004137  000000G             INI.CTLR.DAT::
                    JSR     R1,$SAVE2                       ;                   2179
000004  013701  000000G             MOV     DCT.ADDR,R1     ;                   2192
000010  005011                      CLR     (R1)
000012  012702  000004              MOV     #4,R2           ;                   2193
000016  060102                      ADD     R1,R2
000020  013746  000000G             MOV     CCTLR,-(SP)
000024  012746  000050              MOV     #50,-(SP)
000030  004737  000000G             JSR     PC,BL$MUL
000034  062700  000010'              ADD     #COMM.AREA+10,R0
000040  010012                      MOV     R0,(R2)
000042  010061  000006              MOV     R0,6(R1)        ;                   2194
000046  062761  000014  000006      ADD     #14,6(R1)
000054  012700  000010              MOV     #10,R0          ;                   2195
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)     ;                   2196
000076  062761  000014  000012      ADD     #14,12(R1)
000104  011261  000014              MOV     (R2),14(R1)     ;                   2197
000110  011061  000020              MOV     (R0),20(R1)     ;                   2198
000114  011061  000016              MOV     (R0),16(R1)
000120  022626                      CMP     (SP)+,(SP)+     ;                   2191
000122  000207                      RTS     PC              ;                   2179

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

```
;    2200  1     GLOBAL routine REG_EXIST =
;    2201  1     !+
;    2202  1     !       THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
;    2203  1     !       THE PRESENCE OF AN RDRX DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
;    2204  1     !       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
;    2205  1     !       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
;    2206  1     !       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
;    2207  1     !       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
;    2208  1     !       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
;    2209  1     !-
;    2210  2         begin
;    2211  2
;    2212  2         local
;    2213  2             DUMMY_0 : word,                         ! TEMP FOR READING SA AND IP
;    2214  2             DUMMY_1 : word;                         !
;    2215  2
;    2216  2         if .ENTRY_REASON eql NEW_PASS
;    2217  2         then
;    2218  2             return SUCCESS;                         ! SKIP TEST FOR NEXT PASS
;    2219  2
;    2220  2         OF_RC = 2;                                  ! SET UP TO READ SA FIRST
;    2221  2
;    2222  2         do
;    2223  3             begin
;    2224  3             NEX = FALSE;                            ! SET TO "TRAP NOT RECEIVED"
;    2225  3             SETVEC (4, NEX_TRAP, PRIO7);            ! SET LOCATION 4 TRAP VECTOR ADDRESS
;    2226  3             DUMMY_0 = .(.RDRX_ADDR + .OF_RC);       ! READ REGISTER (THEN TRAP OR CONTINUE)
;    2227  3             DUMMY_1 = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
;    2228  3                                                     !       (TRAP RETURNS TO NEXT INSTRUCTION)
;    2229  3             CLRVEC (4);                             ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
;    2230  3
;    2231  3             if .NEX                                 ! IF NEX TRAP OCCURRED
;    2232  3             then
;    2233  4                 begin
;    2234  4                 C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2235  4
;    2236  4                 if .APT_MODE
;    2237  4                 then
;    2238  5                     begin
;    2239  5                     .MAIL_BOX_TESTNUM = 1;
;    2240  5                     .MAIL_BOX_SUBTST = 0;
;    2241  4                     end;
;    2242  4
;    2243  4                 ERRDF (10, EGD_10, EMS_10);         ! REGISTER EXISTENCE TEST FAILED
;    2244  4                 SETPRI (PRI00);                     ! LOWER PRIORITY
;    2245  4                 return FAILURE;
;    2246  4                 end
;    2247  3             else
;    2248  3                 OF_RC = .OF_RC - 2;                 ! SET UP FOR IP REG OR QUIT
;    2249  3
;    2250  3             end
;    2251  2         until .OF_RC lss 0;
;    2252  2
```

```
;  2253  2          return SUCCESS;
;  2254  1          end;


                                        .SBTTL  REG.EXIST INITIALIZATION TEST ROUTINES
000000  004137  000000G                REG.EXIST::
                                        JSR     R1,$SAVE2                       ;                       2200
000004  123727  000000G 000005          CMPB    ENTRY.REASON,#5                 ;                       2216
000012  001472                          BEQ     4$                              ;                       2218
000014  012737  000002  000000G         MOV     #2,OF.RC                        ;                       2220
000022  005037  000000G         1$:     CLR     NEX                             ;                       2224
000026  012746  000340                  MOV     #340,-(SP)                      ;                       2225
000032  012746  000000G                 MOV     #NEX.TRAP,-(SP)
000036  012746  000004                  MOV     #4,-(SP)
000042  012746  000003                  MOV     #3,-(SP)
000046  104437                          TRAP    37
000050  013700  000000G                 MOV     RDRX.ADDR,R0                    ;                       2226
000054  063700  000000G                 ADD     OF.RC,R0
000060  011001                          MOV     (R0),R1                         ; *,DUMMY.0
000062  005002                          CLR     R2                              ; DUMMY.1               2227
000064  012700  000004                  MOV     #4,R0                           ;                       2229
000070  104436                          TRAP    36
000072  032737  000001  000000G         BIT     #1,NEX                          ;                       2231
000100  001427                          BEQ     3$
000102  013700  000000G                 MOV     CCTLR,R0                        ;                       2234
000106  006300                          ASL     R0
000110  105260  000000G                 INCB    C.ERR.TBL(R0)
000114  032737  000001  001162'         BIT     #1,APT.MODE                     ;                       2236
000122  001405                          BEQ     2$
000124  012777  000001  001164'         MOV     #1,@MAIL.BOX.TESTNUM            ;                       2239
000132  005077  001166'                 CLR     @MAIL.BOX.SUBTST                ;                       2240
000136  104455                  2$:     TRAP    55                              ;                       2243
000140  000012                          .WORD   12
000142  000000G                         .WORD   EGD.10
000144  000000G                         .WORD   EMS.10
000146  005000                          CLR     R0                              ;                       2244
000150  104441                          TRAP    41
000152  062706  000010                  ADD     #10,SP                          ;                       2245
000156  000413                          BR      5$                              ;                       2233
000160  162737  000002  000000G 3$:     SUB     #2,OF.RC                        ;                       2248
000166  062706  000010                  ADD     #10,SP                          ;                       2225
000172  005737  000000G                 TST     OF.RC                           ;                       2251
000176  002311                          BGE     1$
000200  012700  000001          4$:     MOV     #1,R0                           ;                       2210
000204  000207                          RTS     PC
000206  005000                  5$:     CLR     R0                              ;                       2200
000210  000207                          RTS     PC

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

```
;    2255  1     GLOBAL routine VEC_BR_TEST =
;    2256  1
;    2257  1     !+
;    2258  1     !       THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RDRX VECTOR ADDRESS GIVEN
;    2259  1     !       BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;    2260  1     !       USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;    2261  1     !       OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;    2262  1     !       AN RDRX INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;    2263  1     !       THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;    2264  1     !       "FUNCTIONAL TEST STARTED" IS PRINTED BEFORE THE TEST, AND
;    2265  1     !       "EXERCISER STARTED" IS PRINTED AT ITS SUCCESSFUL CONCLUSION. IF
;    2266  1     !       EITHER "FUNCTIONAL TEST ..." OR "EXERCISER ..." DOES NOT APPEAR, THEN
;    2267  1     !       PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP IS LIKELY TO OCCUR. AT
;    2268  1     !       THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;    2269  1     !
;    2270  1     !       IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;    2271  1     !       PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;    2272  1     !       USER. A FORCED INTERRUPT SHOULD NOT OCCURR. THEN, BY LOWERING THE
;    2273  1     !       PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;    2274  1     !-
;    2275  1
;    2276  2         begin
;    2277  2
;    2278  2         if .ENTRY_REASON eql NEW_PASS
;    2279  2         then
;    2280  3             begin
;    2281  3             SETPRI (PRI00);                         ! LOWER PRIORITY
;    2282  3             return SUCCESS;                         ! SKIP TEST IF NEXT PASS
;    2283  2             end;
;    2284  2
;    2285  2         PRINTF (MSG_02);                            ! "FUNCTIONAL TEST STARTED"
;    2286  2
;    2287  2         if INT_GEN () eql FALSE                     ! FORCE AN INTERRUPT
;    2288  2         then
;    2289  3             begin                                   ! IF INTERRUPT DID NOT OCCUR
;    2290  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2291  3
;    2292  3             if .APT_MODE
;    2293  3             then
;    2294  4                 begin
;    2295  4                 .MAIL_BOX_TESTNUM = 1;
;    2296  4                 .MAIL_BOX_SUBTST = 0;
;    2297  3                 end;
;    2298  3
;    2299  3             ERRDF (11, EGD_11, 0);                  ! VECTOR TEST FAILED
;    2300  3             return FAILURE;
;    2301  3             end
;    2302  2         else
;    2303  3             begin                                   ! INTERRUPT DID OCCUR
;    2304  3             PRINTF (MSG_03);                        ! "EXERCISER STARTED"
;    2305  3             SETPRI (.BRLEVEL);                      ! SET PRIORITY TO ASSUMED BR LEVEL
;    2306  3
;    2307  3             if INT_GEN () eql FALSE                 ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
```

```
;    2308  3              then
;    2309  4                  begin                           ! IF INTERRUPT DID NOT OCCUR
;    2310  4                  SETPRI (.BRLEVEL - %o'40');     ! LOWER PRIORITY BY 1
;    2311  4                  DELAY (1);                      ! WAIT
;    2312  4
;    2313  4                  if .DCT_ADDR [SA_SAVE] neq 0    ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON-ZERO)
;    2314  4                  then
;    2315  5                      begin
;    2316  5                      SETPRI (PRI00);             ! RESTORE PROCESSOR PRIORITY TO 0
;    2317  5                      return SUCCESS;             ! ONLY SUCCESSFUL EXIT POINT
;    2318  4                      end;
;    2319  4
;    2320  3                  end;
;    2321  3
;    2322  2              end;
;    2323  2
;    2324  2          SETPRI (PRI00);                         ! COME HERE ONLY FOR BR TEST FAILURE
;    2325  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2326  2
;    2327  2          if .APT_MODE
;    2328  2          then
;    2329  3              begin
;    2330  3              .MAIL_BOX_TESTNUM = 1;
;    2331  3              .MAIL_BOX_SUBTST = 0;
;    2332  2              end;
;    2333  2
;    2334  2          ERRDF (12, EGD_12, EMS_12);
;    2335  2          return FAILURE;
;    2336  1          end;

                        .GLOBL  L$DLY

                        .SBTTL  VEC.BR.TEST INITIALIZATION TEST ROUTINES
000000  010146      VEC.BR.TEST::
                        MOV     R1,-(SP)                    ;                       2255
000002  005746          TST     (SP)
000004  123727  000000G 000005  CMPB    ENTRY.REASON,#5     ;                       2278
000012  001003          BNE     1$
000014  005000          CLR     R0                          ;                       2281
000016  104441          TRAP    41
000020  000504          BR      8$                          ;                       2280
000022  012746  000000G 1$:     MOV     #MSG.02,-(SP)       ;                       2285
000026  012746  000001          MOV     #1,-(SP)
000032  010600                  MOV     SP,R0               ; SP,*
000034  104417                  TRAP    17
000036  004737  000000V         JSR     PC,INT.GEN          ;                       2287
000042  005700                  TST     R0
000044  001023                  BNE     3$
000046  013700  000000G         MOV     CCTLR,R0            ;                       2290
000052  006300                  ASL     R0
000054  105260  000000G         INCB    C.ERR.TBL(R0)
```

```
000060  032737  000001  001162'            BIT     #1,APT.MODE                     ;                               2292
000066  001405                             BEQ     2$
000070  012777  000001  001164'            MOV     #1,@MAIL.BOX.TESTNUM            ;                               2295
000076  005077  001166'                     CLR     @MAIL.BOX.SUBTST                ;                               2296
000102  104455                  2$:        TRAP    55                              ;                               2299
000104  000013                             .WORD   13
000106  000000G                            .WORD   EGD.11
000110  000000                             .WORD   0
000112  000477                             BR      11$                             ;                               2300
000114  012716  000000G           3$:      MOV     #MSG.03,(SP)                    ;                               2304
000120  012746  000001                     MOV     #1,-(SP)
000124  010600                             MOV     SP,R0                           ; SP,*
000126  104417                             TRAP    17
000130  013700  001154'                     MOV     BRLEVEL,R0                      ;                               2305
000134  104441                             TRAP    41
000136  004737  000000V                     JSR     PC,INT.GEN                      ;                               2307
000142  005700                             TST     R0
000144  001035                             BNE     9$
000146  013700  001154'                     MOV     BRLEVEL,R0                      ;                               2310
000152  162700  000040                     SUB     #40,R0
000156  104441                             TRAP    41
000160  012701  000001                     MOV     #1,R1                           ; *,$$TMP2                      2311
000164  001411                  4$:        BEQ     7$
000166  013700  000000G                     MOV     L$DLY,R0                        ; *,$$TMP1
000172  001404                             BEQ     6$
000174  005066  000006           5$:      CLR     6(SP)                           ; $$TMP
000200  005300                             DEC     R0                              ; $$TMP1
000202  001374                             BNE     5$
000204  005301                  6$:        DEC     R1                              ; $$TMP2
000206  000766                             BR      4$
000210  013700  000000G           7$:      MOV     DCT.ADDR,R0                     ;                               2313
000214  005760  000002                     TST     2(R0)
000220  001407                             BEQ     8$
000222  005000                             CLR     (R0)                            ;                               2316
000224  104441                             TRAP    41
000226  062706  000006                     ADD     #6,SP                           ;                               2317
000232  012700  000001           8$:      MOV     #1,R0                           ;                               2315
000236  000427                             BR      12$
000240  005726                  9$:        TST     (SP)+                           ;                               2303
000242  005000                             CLR     R0                              ;                               2324
000244  104441                             TRAP    41
000246  013700  000000G                     MOV     CCTLR,R0                        ;                               2325
000252  006300                             ASL     R0
000254  105260  000000G                     INCB    C.ERR.TBL(R0)
000260  032737  000001  001162'            BIT     #1,APT.MODE                     ;                               2327
000266  001405                             BEQ     10$
000270  012777  000001  001164'            MOV     #1,@MAIL.BOX.TESTNUM            ;                               2330
000276  005077  001166'                     CLR     @MAIL.BOX.SUBTST                ;                               2331
000302  104455                  10$:       TRAP    55                              ;                               2334
000304  000014                             .WORD   14
000306  000000G                            .WORD   EGD.12
000310  000000G                            .WORD   EMS.12
000312  022626                  11$:       CMP     (SP)+,(SP)+                     ;                               2335
```

```
000314  005000                          CLR     R0                              ;                               2255
000316  005726                  12$:    TST     (SP)+
000320  012601                          MOV     (SP)+,R1
000322  000207                          RTS     PC

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

```
;    2337  1     GLOBAL routine INT_GEN =
;    2338  1
;    2339  1     !+
;    2340  1     !       THIS ROUTINE BEGINS AN RDRX INITIALIZATION SEQUENCE, BUT ONLY
;    2341  1     !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RDRX
;    2342  1     !       INTERRUPT (AT THE COMPLETEION OF STEP 1) IN ORDER TO HELP VERIFY THE
;    2343  1     !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;    2344  1     !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;    2345  1     !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;    2346  1     !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;    2347  1     !-
;    2348  1
;    2349  2         begin
;    2350  2
;    2351  2         local
;    2352  2             SA : word;                                          ! STORAGE FOR STEP 1 READ AND WRITE
;    2353  2
;    2354  2         DCT_ADDR [SA_SAVE] = 0;                                 ! ZERO OUT SA SAVE WORD IN DCT
;    2355  2         WRT_RDRX (RCIP, RC_ALL, ALL_ONES);                      ! WRITE IP TO START INIT SEQUENCE
;    2356  2         DELAY (2);                                              ! WAIT
;    2357  2         SA = .RDRX_ADDR [RCSA, RC_ALL];                         ! STEP 1 READ
;    2358  2         SA = (WR_RING + 8) or (.CURRENT_VECTOR + -2) or SA_INT; ! STEP 1 WRITE VALUE
;    2359  2         WRT_RDRX (RCSA, RC_ALL, .SA);                           ! STEP 1 WRITE
;    2360  2
;    2361  2         incr COUNT from 1 to 8000 do
;    2362  3             begin
;    2363  3             DELAY (1);                                          ! TOTAL DELAY COUNT OF 8,000
;    2364  3
;    2365  3             if .DCT_ADDR [SA_SAVE] neq 0                        ! IF SA WAS CHANGED
;    2366  3             then
;    2367  3                 return TRUE;                                    ! INTERRUPT OCCURED
;    2368  3
;    2369  3             BREAK;
;    2370  2             end;
;    2371  2
;    2372  2         return FALSE;                                           ! IF INTERRUPT DID NOT OCCUR
;    2373  1         end;


                                .SBTTL  INT.GEN INITIALIZATION TEST ROUTINES
000000  004137  000000G    INT.GEN::
                            JSR     R1,$SAVE2                       ;                       2337
000004  024646              CMP     -(SP),(SP)
000006  013700  000000G     MOV     DCT.ADDR,R0                     ;                       2354
000012  005060  000002      CLR     2(R0)
000016  012700  177777      MOV     #-1,R0                          ; #,RC.REG              2355
000022  010077  000000G     MOV     R0,@RDRX.ADDR                   ; RC.REG,#
000026  012701  000002      MOV     #2,R1                           ; #,$$TMP2              2356
000032  001411              BEQ     4$
000034  013700  000000G 1$:  MOV     L$DLY,R0                        ; #,$$TMP1
000040  001404              BEQ     3$
000042  005066  000002 2$:  CLR     2(SP)                           ; $$TMP
000046  005300              DEC     R0                              ; $$TMP1
```

```
000050  001374                          BNE     2$
000052  005301                  3$:     DEC     R1                      ; $$TMP2
000054  000766                          BR      1$
000056  013700  000000G          4$:     MOV     RDRX.ADDR,R0            ;                               2357
000062  016016  000002                  MOV     2(R0),(SP)              ; *,RC.REG
000066  013701  001152'                  MOV     CURRENT.VECTOR,R1       ;                               2358
000072  006201                          ASR     R1
000074  006201                          ASR     R1
000076  010102                          MOV     R1,R2                   ; *,SA
000100  052702  111200                  BIS     #111200,R2              ; *,SA
000104  010201                          MOV     R2,R1                   ; SA,RC.REG                     2359
000106  010160  000002                  MOV     R1,2(R0)                ; RC.REG,*
000112  012702  017500                  MOV     #17500,R2               ; *,COUNT                       2361
000116  012701  000001          5$:     MOV     #1,R1                   ; *,$$TMP2                      2363
000122  001411                  6$:     BEQ     9$
000124  013700  000000G                  MOV     L$DLY,R0                ; *,$$TMP1
000130  001404                          BEQ     8$
000132  005066  000002          7$:     CLR     2(SP)                   ; $$TMP
000136  005300                          DEC     R0                      ; $$TMP1
000140  001374                          BNE     7$
000142  005301                  8$:     DEC     R1                      ; $$TMP2
000144  000766                          BR      6$
000146  013700  000000G          9$:     MOV     DCT.ADDR,R0             ;                               2365
000152  005760  000002                  TST     2(R0)
000156  001403                          BEQ     10$
000160  012700  000001                  MOV     #1,R0                   ;                               2367
000164  000404                          BR      11$
000166  104422                  10$:    TRAP    22
000170  005302                          DEC     R2                      ; COUNT                         2361
000172  001351                          BNE     5$
000174  005000                          CLR     R0                      ;                               2349
000176  022626                  11$:    CMP     (SP)+,(SP)+             ;                               2337
000200  000207                          RTS     PC

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

```
;    2374  1     GLOBAL routine HARD_INIT =
;    2375  1
;    2376  1     !+
;    2377  1     !       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;    2378  1     !       INITIALIZE AN RDRX DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;    2379  1     !       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;    2380  1     !       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;    2381  1     !       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;    2382  1     !-
;    2383  1
;    2384  2         begin
;    2385  2
;    2386  2         local
;    2387  2             IE_VEC : word;                                  ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
;    2388  2                                                             !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;    2389  2
;    2390  2         IE_VEC = .CURRENT_VECTOR + -2;                      ! GET VECTOR ADDR/4 (IE - 0)
;    2391  2
;    2392  2         incr ATTEMPTS from 1 to INI_ATT do
;    2393  3             begin
;    2394  3
;    2395  3             label
;    2396  3                 STEP_1_READ,
;    2397  3                 STEP_2_READ,
;    2398  3                 STEP_3_READ,
;    2399  3                 STEP_4_READ;
;    2400  3
;    2401  3             WRT_RDRX (RCIP, RC_ALL, ALL_ONES);              ! WRITE IP TO START INIT SEQUENCE
;    2402  3     !
;    2403  3     !       STEP 1 READ
;    2404  3     !
;    2405  3             STEP = 1;
;    2406  3             STEP_1_READ:
;    2407  4                 begin
;    2408  4
;    2409  4                 incr COUNT from 1 to 500 do
;    2410  5                     begin
;    2411  5                     DELAY (1);                              ! TOTAL DELAY COUNT OF 500 FOR STEP 1
;    2412  5                     SA_REG = .RDRX_ADDR [RCSA, RC_ALL];     ! READ SA
;    2413  5
;    2414  5                     if (.SA_REG and S1_MASK) eql SA_S1      ! IF STEP 1 READ IS O.K.
;    2415  5                     then
;    2416  5                         leave STEP_1_READ;
;    2417  5
;    2418  5                     BREAK;
;    2419  4                     end;
;    2420  4
;    2421  4                 exitloop;
;    2422  3                 end;
;    2423  3
;    2424  3     !
;    2425  3     !       STEP 1 WRITE
;    2426  3     !
```

```
;  2427  3              SA_REG = (WR_RING ^ 8) or .IE_VEC;                  ! STEP 1 WRITE VALUE
;  2428  3              WRT_RDRX (RCSA, RC_ALL, .SA_REG);                   ! STEP 1 WRITE
;  2429  3      !
;  2430  3      !       STEP 2 READ
;  2431  3      !
;  2432  3              STEP = .STEP + 1;
;  2433  3              STEP_2_READ:
;  2434  4                  begin
;  2435  4
;  2436  4                  incr COUNT from 1 to 10000 do
;  2437  5                      begin
;  2438  5                      DELAY (1);                                  ! TOTAL DELAY COUNT OF 10,000 FOR STEP 2
;  2439  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];         ! READ SA
;  2440  5
;  2441  6                      if (.SA_REG and S2_MASK) eql (SA_S2 or WR_RING) ! IF STEP 2 READ IS O.K.
;  2442  5                      then
;  2443  5                          leave STEP_2_READ;
;  2444  5
;  2445  5                      BREAK;
;  2446  4                      end;
;  2447  4
;  2448  4                  exitloop;
;  2449  3                  end;
;  2450  3
;  2451  3      !
;  2452  3      !       STEP 2 WRITE
;  2453  3      !
;  2454  3              WRT_RDRX (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]);        ! RINGBASE-LO, PI = 0
;  2455  3      !
;  2456  3      !       STEP 3 READ
;  2457  3      !
;  2458  3              STEP = .STEP + 1;
;  2459  3              STEP_3_READ:
;  2460  4                  begin
;  2461  4
;  2462  4                  incr COUNT from 1 to 10000 do
;  2463  5                      begin
;  2464  5                      DELAY (1);                                  ! TOTAL DELAY COUNT OF 10,000 FOR STEP 3 READ
;  2465  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];         ! READ SA
;  2466  5
;  2467  6                      if (.SA_REG and S3_MASK) eql (SA_S3 or .IE_VEC) ! IF STEP 3 READ IS O.K.
;  2468  5                      then
;  2469  5                          leave STEP_3_READ;
;  2470  5
;  2471  5                      BREAK;
;  2472  4                      end;
;  2473  4
;  2474  4                  exitloop;
;  2475  3                  end;
;  2476  3
;  2477  3      !
;  2478  3      !       STEP 3 WRITE
;  2479  3      !
```

```
;    2480  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! PP, RINGBASE-HI = 0
;    2481  3      !
;    2482  3      !       STEP 4 READ
;    2483  3      !
;    2484  3              STEP = .STEP + 1;
;    2485  3              STEP_4_READ:
;    2486  4                  begin
;    2487  4
;    2488  4                  incr COUNT from 1 to 10000 do
;    2489  5                      begin
;    2490  5                      DELAY (1);                                   ! TOTAL DELAY COUNT OF 10,000 FOR STEP 4 READ
;    2491  5                      SA_REG = .RDRX_ADDR [RCSA, RC_ALL];          ! READ SA
;    2492  5
;    2493  5                      if (.SA_REG and S4_MASK) eql SA_S4           ! IF STEP 4 READ IS O.K.
;    2494  5                      then
;    2495  5                          leave STEP_4_READ;
;    2496  5
;    2497  5                      BREAK;
;    2498  4                      end;
;    2499  4
;    2500  4                  exitloop;
;    2501  3                  end;
;    2502  3      !
;    2503  3      !       STEP 4 WRITE
;    2504  3      !
;    2505  3              CREDIT_BAL = 1;                                      ! START WITH A CREDIT BALANCE = 1
;    2506  3              WRT_RDRX (RCSA, RC_ALL, 0);                          ! BURST, LF, GO = 0
;    2507  3              return SUCCESS;                                      ! SUCCESS EXIT POINT
;    2508  3
;    2509  2              end;                                                 ! TRY AGAIN OR GIVE UP
;    2510  2
;    2511  2          CREDIT_BAL = 0;                                          ! NO CREDIT BALANCE
;    2512  2          C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2513  2
;    2514  2          IF .APT_MODE
;    2515  2          then
;    2516  3              begin
;    2517  3              .MAIL_BOX_TESTNUM = 1;
;    2518  3              .MAIL_BOX_SUBTST = 0;
;    2519  2              end;
;    2520  2
;    2521  2          ERRDF (13, EGD_13, EMS_13);                              ! INIT SEQUENCE FAILED
;    2522  2          return FAILURE;
;    2523  1          end;                                                     ! ROUTINE HARD_INIT

                                        .SBTTL  HARD.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G          HARD.INIT::
                                        JSR     R1,$SAVE5                    ;                                  2374
000004  162706  000012                  SUB     #12,SP
000010  013704  001152'                 MOV     CURRENT.VECTOR,R4            ; *,IE.VEC                         2390
000014  006204                          ASR     R4                           ; IE.VEC
000016  006204                          ASR     R4                           ; IE.VEC
```

```
000020  012705  000002                        MOV     #2,R5                   ; *,ATTEMPTS                    2392
000024  012700  177777                        MOV     #-1,R0                  ; *,RC.REG                      2401
000030  010077  000000G                       MOV     R0,@RDRX.ADDR           ; RC.REG,*
000034  012737  000001  000000G               MOV     #1,STEP                 ;                               2405
000042  012702  000764                        MOV     #764,R2                 ; *,COUNT                       2409
000046  012701  000001              1$:       MOV     #1,R1                   ; *,$$TMP2                      2411
000052  001411                      2$:       BEQ     5$
000054  013700  000000G                       MOV     L$DLY,R0                ; *,$$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                      ; $$TMP2
000074  000766                              BR      2$
000076  013700  000000G             5$:       MOV     RDRX.ADDR,R0            ;                               2412
000102  016016  000002                        MOV     2(R0),(SP)              ; *,RC.REG
000106  011637  000000G                       MOV     (SP),SA.REG             ; RC.REG,*
000112  011600                              MOV     (SP),R0                 ; SA.REG,*                      2414
000114  042700  001777                        BIC     #1777,R0
000120  020027  004000                        CMP     R0,#4000
000124  001404                              BEQ     6$                      ;                               2416
000126  104422                              TRAP    22
000130  005302                              DEC     R2                      ; COUNT                         2409
000132  001345                              BNE     1$
000134  000532                              BR      18$                     ;                               2393
000136  010437  000000G             6$:       MOV     R4,SA.REG               ; IE.VEC,*                      2427
000142  052737  111000  000000G               BIS     #111000,SA.REG
000150  013701  000000G                       MOV     SA.REG,R1               ; *,RC.REG                      2428
000154  013700  000000G                       MOV     RDRX,ADDR,R0
000160  010160  000002                        MOV     R1,2(R0)                ; RC.REG,*
000164  005237  000000G                       INC     STEP                    ;                               2432
000170  012702  023420                        MOV     #23420,R2               ; *,COUNT                       2436
000174  012701  000001              7$:       MOV     #1,R1                   ; *,$$TMP2                      2438
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$:      MOV     RDRX,ADDR,R0            ;                               2439
000230  016066  000002  000002              MOV     2(R0),2(SP)             ; *,RC.REG
000236  016637  000002  000000G               MOV     2(SP),SA.REG            ; RC.REG,*
000244  016600  000002                        MOV     2(SP),R0                ; SA.REG,*                      2441
000250  042700  003400                        BIC     #3400,R0
000254  020027  010222                        CMP     R0,#10222
000260  001404                              BEQ     12$                     ;                               2443
000262  104422                              TRAP    22
000264  005302                              DEC     R2                      ; COUNT                         2436
000266  001342                              BNE     7$
000270  000537                              BR      26$                     ;                               2453
000272  013700  000000G             12$:      MOV     DCT,ADDR,R0             ;                               2454
```

```
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                    ;                               2458
000316  010403                                  MOV     R4,R3                   ; IE.VEC,*                      2467
000320  052703  020000                          BIS     #20000,R3
000324  012702  023420                          MOV     #23420,R2               ; *,COUNT                       2462
000330  012701  000001                  13$:    MOV     #1,R1                   ; *,$$TMP2                      2464
000334  001411                          14$:    BEQ     17$
000336  013700  000000G                         MOV     L$DLY,R0                ; *,$$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                      ; $$TMP2
000356  000766                                  BR      14$
000360  013700  000000G                 17$:    MOV     RDRX.ADDR,R0            ;                               2465
000364  016066  000002  000004                 MOV     2(R0),4(SP)             ; *,RC.REG
000372  016637  000004  000000G                 MOV     4(SP),SA.REG            ; RC.REG,*
000400  016600  000004                          MOV     4(SP),R0                ; SA.REG,*                      2467
000404  042700  003400                          BIC     #3400,R0
000410  020003                                  CMP     R0,R3
000412  001404                                  BEQ     19$                     ;                               2469
000414  104422                                  TRAP    22
000416  005302                                  DEC     R2                      ; COUNT                         2462
000420  001343                                  BNE     13$
000422  000462                          18$:    BR      26$                     ;                               2393
000424  013700  000000G                 19$:    MOV     RDRX.ADDR,R0            ;                               2480
000430  005060  000002                          CLR     2(R0)
000434  005237  000000G                         INC     STEP                    ;                               2484
000440  012703  023420                          MOV     #23420,R3               ; *,COUNT                       2488
000444  012701  000001                  20$:    MOV     #1,R1                   ; *,$$TMP2                      2490
000450  001411                          21$:    BEQ     24$
000452  013700  000000G                         MOV     L$DLY,R0                ; *,$$TMP1
000456  001404                                  BEQ     23$
000460  005066  000010                  22$:    CLR     10(SP)                  ; $$TMP
000464  005300                                  DEC     R0                      ; $$TMP1
000466  001374                                  BNE     22$
000470  005301                          23$:    DEC     R1                      ; $$TMP2
000472  000766                                  BR      21$
000474  013700  000000G                 24$:    MOV     RDRX.ADDR,R0            ;                               2491
000500  016066  000002  000006                 MOV     2(R0),6(SP)             ; *,RC.REG
000506  016637  000006  000000G                 MOV     6(SP),SA.REG            ; RC.REG,*
000514  016600  000006                          MOV     6(SP),R0                ; SA.REG,*                      2493
000520  042700  003777                          BIC     #3777,R0
000524  020027  040000                          CMP     R0,#40000
000530  001404                                  BEQ     25$                     ;                               2495
000532  104422                                  TRAP    22
000534  005303                                  DEC     R3                      ; COUNT                         2488
000536  001342                                  BNE     20$
000540  000413                                  BR      26$                     ;                               2393
000542  012737  000001  000000G         25$:    MOV     #1,CREDIT.BAL           ;                               2505
000550  005001                                  CLR     R1                      ; RC.REG                        2506
```

```
000552  013700  000000G                       MOV     RDRX,ADDR,R0
000556  005060  000002                        CLR     2(R0)
000562  012700  000001                        MOV     #1,R0                   ;                       2393
000566  000425                                BR      28$
000570  005037  000000G             26$:      CLR     CREDIT.BAL              ;                       2511
000574  013700  000000G                       MOV     CCTLR,R0                ;                       2512
000600  006300                                ASL     R0
000602  105260  000000G                       INCB    C.ERR.TBL(R0)
000606  032737  000001  001162'               BIT     #1,APT.MODE             ;                       2514
000614  001405                                BEQ     27$
000616  012777  000001  001164'               MOV     #1,@MAIL.BOX.TESTNUM    ;                       2517
000624  005077  001166'                       CLR     @MAIL.BOX.SUBTST        ;                       2518
000630  104455                      27$:      TRAP    55                      ;                       2521
000632  000015                                .WORD   15
000634  000000G                               .WORD   EGD.13
000636  000000G                               .WORD   EMS.13
000640  005000                                CLR     R0                      ;                       2384
000642  062706  000012              28$:      ADD     #12,SP                  ;                       2374
000646  000207                                RTS     PC

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

```
;    2524  1     GLOBAL routine INI_RRING : novalue =
;    2525  1
;    2526  1     !+
;    2527  1     !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP PACKETS TO
;    2528  1     !       FILL AN RDRX RESPONSE RING. THE BUFFER DESCRIPTOR OF EACH PACKET
;    2529  1     !       (LOCATED IN FRONT OF THE PACKET ITSELF) IS LOADED INTO SUCCESSIVE
;    2530  1     !       RRING SLOTS. NOTE THAT THE BUFFER DESCRIPTORS HAVE BEEN INITIALIZED
;    2531  1     !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
;    2532  1     !       CONTROLLER-OWNED.
;    2533  1     !
;    2534  1     !       IMPLICIT INPUTS:
;    2535  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2536  1     !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;    2537  1     !-
;    2538  1
;    2539  2         begin
;    2540  2
;    2541  2         local
;    2542  2             index : word,
;    2543  2             RRING_ADDR;
;    2544  2
;    2545  2         RRING_ADDR = .DCT_ADDR [RR_BEG];                    ! FIRST RESPONSE RING SLOT
;    2546  2
;    2547  2         incr COUNT from 1 to RRING_LEN do
;    2548  3             begin
;    2549  3             index = GET_PKT (.CCTLR);                       ! GET AN MSCP PACKET
;    2550  3             .RRING_ADDR = .MSCP_PKT [.index, PKT_LO];       ! LOAD LO-ORDER BUFF DESC INTO SLOT
;    2551  3             RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO SECOND WORD
;    2552  3             .RRING_ADDR = .MSCP_PKT [.index, PKT_HI];       ! LOAD HI-ORDER BUFF DESC INTO SLOT
;    2553  3             PKT_USE [.index] = .CCTLR;                      ! PACKET IN USE
;    2554  3             .RRING_ADDR = ..RRING_ADDR or ED_OWN or ED_FLAG;! GIVE OWNERSHIP TO CONTRLLER
;    2555  3             RRING_ADDR = .RRING_ADDR + 2;                   ! ADVANCE TO NEXT SLOT
;    2556  2             end;
;    2557  2
;    2558  1         end;


                                .SBTTL  INI.RRING INITIALIZATION TEST ROUTINES
000000  004137  000000G    INI.RRING::
                            JSR     R1,$SAVE4                           ;                       2524
000004  013700  000000G            MOV     DCT.ADDR,R0                 ;                       2545
000010  016001  000004             MOV     4(R0),R1                    ; *,RRING.ADDR
000014  013703  000000G            MOV     CCTLR,R3                    ;                       2549
000020  012704  000004             MOV     #4,R4                       ; *,COUNT               2547
000024  010346              1$:    MOV     R3,-(SP)                    ;                       2549
000026  004737  000000G            JSR     PC,GET.PKT
000032  010002                     MOV     R0,R2                       ; *,INDEX
000034  010216                     MOV     R2,(SP)                     ; INDEX,*               2550
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          2552
000056  013703  000000G            MOV     CCTLR,R3                    ;                       2553
```

```
000062  110362  000000G                    MOVB    R3,PKT.USE(R2)          ; *,*(INDEX)
000066  052721  140000                      BIS     #140000,(R1)+           ; *.RRING.ADDR                  2554
000072  022626                              CMP     (SP)+,(SP)+             ;                               2548
000074  005304                              DEC     R4                      ; COUNT                         2547
000076  001352                              BNE     1$
000100  000207                              RTS     PC                      ;                               2524

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

```
;    2559  1     GLOBAL routine SET_CTLR_CHAR =
;    2560  1
;    2561  1     !+
;    2562  1     !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RDRX HAS BEEN HARD-
;    2563  1     !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;    2564  1     !       CHARACTERISTICS" COMMAND, AND TO VALIDATE THE RESPONSE (END MESSAGE).
;    2565  1     !
;    2566  1     !       IMPLICIT INPUTS:
;    2567  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2568  1     !-
;    2569  1
;    2570  2         begin
;    2571  2
;    2572  2         local
;    2573  2             P_INDEX : word;
;    2574  2
;    2575  2
;    2576  2     ! MISCELLANEOUS INITIALIZATION                                          !ZZZ
;    2577  2                                                                            !ZZZ
;    2578  2     QIO [.CCTLR] = 0;               !INIT NO OF OUTSTANDING QIOS           !ZZZ
;    2579  2     CST [.CCTLR, U_CNT] = 0;        !CLEAR UNITS IN CST TABLE              !ZZZ
;    2580  2     INCR COUNT FROM 0 TO (RP_CNT - 1) DO    !INIT RETURN PACKET POOL       !ZZZ
;    2581  2          RP_USE [.COUNT] = -1;                                             !ZZZ
;    2582  2                                                                            !ZZZ
;    2583  2     IODQ_IN = IODQ_OUT = 0;         !INIT I/O DONE QUEUE POINTERS          !ZZZ
;    2584  2
;    2585  2
;    2586  2         P_INDEX = GET_PKT (.CCTLR);                                        ! GET AN MSCP PACKET
;    2587  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_SCC;                              ! PACKET SIZE
;    2588  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_SCC;                              ! OPCODE = SET CTLR CHAR
;    2589  2         MSCP_PKT [.P_INDEX, C_FLAGS] = CF_MASK;                            ! CONTROLLER FLAGS
;    2590  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = IMM_CMD;                           ! IMMEDIATE COMMAND
;    2591  2
;    2592  2         if SEND (.P_INDEX) eql FAILURE                                     ! ATTEMPT SEND
;    2593  2         then
;    2594  3             begin                                                          ! IF SEND WAS UNSUCCESSFUL
;    2595  3             C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2596  3
;    2597  3             if .APT_MODE
;    2598  3             then
;    2599  4                 begin
;    2600  4                 .MAIL_BOX_TESTNUM = 1;
;    2601  4                 .MAIL_BOX_SUBTST = 0;
;    2602  3                 end;
;    2603  3
;    2604  3             ERRDF (20, EGD_20, 0);                                     ! FATAL ERROR
;    2605  3             PUT_PKT (.P_INDEX);                                        ! RETURN PACKET TO POOL
;    2606  3             DROP_CTLR (.CCTLR, DU_CFATAL);                             ! DROP CONTROLLER
;    2607  3             return FAILURE;
;    2608  3             end
;    2609  2         else
;    2610  3             begin                                                      ! IF SEND WAS SUCCESSFUL
;    2611  3
```

```
;    2612  3          do
;    2613  4              begin
;    2614  4              WAIT ();                                          ! WAIT FOR RETPKT RESPONSE
;    2615  4              RP_INDX = OUT_IODQ ();                            ! GET INDEX OF RETPKT
;    2616  4              RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);       ! CALCULATE RETPKT ADDRESS
;    2617  4
;    2618  4              if .RP_ADDR [MESTYP] neq MT_SEQ                   ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2619  4              then
;    2620  4                  PUT_RETPKT (.RP_INDX);
;    2621  4
;    2622  4              end
;    2623  3          until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    2624  4                ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    2625  3                 ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2626  3
;    2627  3          if .RP_ADDR [CONID] eql CID_DRIVER                    ! IF RETPKT IS FROM "DRIVER"
;    2628  3          then
;    2629  4              begin
;    2630  4              PRINTF (DBM23);                                   ! "ERROR IN SET_CTLR_CHAR"
;    2631  4              PUT_RETPKT (.RP_INDX);                            ! RELEASE RETURN PACKET
;    2632  4              DR_ERR ();                                        ! DROP CONTROLLER
;    2633  4              return FAILURE;
;    2634  4              end
;    2635  3          else
;    2636  4              begin                                             ! ELSE - RETPKT IS FROM DISK MSCP
;    2637  4
;    2638  4              if (.RP_ADDR [ENDCOD] neq (OP_SCC or OP_END)) or  ! IF WRONG ENDCODE
;    2639  5                 ((.RP_ADDR [C_FLGS] and CF_MASK) neq CF_MASK)  ! OR FLAGS IN ERROR
;    2640  4              then
;    2641  5                  begin
;    2642  5                  C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;    2643  5
;    2644  5                  if .APT_MODE
;    2645  5                  then
;    2646  6                      begin
;    2647  6                      .MAIL_BOX_TESTNUM = 1;
;    2648  6                      .MAIL_BOX_SUBTST = 0;
;    2649  5                      end;
;    2650  5
;    2651  5                  ERRDF (21, EGD_21, EMS_21);                   ! FATAL ERROR
;    2652  5                  DROP_CTLR (.CCTLR, DU_CFATAL);                ! DROP CONTROLLER
;    2653  5                  PUT_RETPKT (.RP_INDX);                        ! RELEASE RETURN PACKET
;    2654  5                  return FAILURE
;    2655  5                  end
;    2656  4              else
;    2657  5                                                                ! RETPKT HAS CORRECT ENDCODE
;    2658  5                  CMD_TIME = .RP_ADDR [C_TIME] * 2;
;    2659  5
;    2660  6                  if BIT_TST (SWP_FLAGS, SWF_TRC)
;    2661  5                  then
;    2662  5                      PRINTF (DBM25, .RP_ADDR [C_TIME]);
;    2663  5
;    2664  4                  end;                                          ! RETPKT HAS CORRECT ENDCODE
```

```
;    2665  4
;    2666  3                end;                                    ! IF RETPKT WAS SENT BY DISK MSCP
;    2667  3
;    2668  3            PUT_RETPKT (.RP_INDX);
;    2669  3            return SUCCESS;                             ! IF SEND WAS SUCCESSFUL
;    2670  2            end;
;    2671  2
;    2672  1        end;                                            ! ROUTINE SET_CTLR_CHAR


                                        .SBTTL  SET.CTLR.CHAR INITIALIZATION TEST ROUTINES
000000  010146                    SET.CTLR.CHAR::
                                        MOV     R1,-(SP)                ;                       2559
000002  013701  000000G                 MOV     CCTLR,R1                ;                       2578
000006  105061  000000G                 CLRB    QIO(R1)
000012  010146                          MOV     R1,-(SP)                ;                       2579
000014  012746  000126                  MOV     #126,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  105060  000005G                 CLRB    CST+5(R0)
000030  005000                          CLR     R0                      ; COUNT                 2580
000032  112760  000377  000000G  1$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)            2581
000040  005200                          INC     R0                      ; COUNT                 2580
000042  020027  000007                  CMP     R0,#7                   ; COUNT,*
000046  003771                          BLE     1$
000050  005037  000000G                 CLR     IODQ.OUT                ;                       2583
000054  005037  000000G                 CLR     IODQ.IN
000060  010116                          MOV     R1,(SP)                 ;                       2586
000062  004737  000000G                 JSR     PC,GET.PKT
000066  010001                          MOV     R0,R1                   ; *,P.INDEX
000070  010116                          MOV     R1,(SP)                 ; P.INDEX,*             2587
000072  012746  000106                  MOV     #106,-(SP)
000076  004737  000000G                 JSR     PC,BL$MUL
000102  012760  000040  000006G         MOV     #40,MSCP.PKT+6(R0)
000110  112760  000004  000022G         MOVB    #4,MSCP.PKT+22(R0)      ;                       2588
000116  012760  000120  000030G         MOV     #120,MSCP.PKT+30(R0)    ;                       2589
000124  105060  000004G                 CLRB    MSCP.PKT+4(R0)          ;                       2590
000130  010116                          MOV     R1,(SP)                 ; P.INDEX,*             2592
000132  004737  000000G                 JSR     PC,SEND
000136  005700                          TST     R0
000140  001036                          BNE     3$
000142  013700  000000G                 MOV     CCTLR,R0                ;                       2595
000146  006300                          ASL     R0
000150  105260  000000G                 INCB    C.ERR.TBL(R0)
000154  032737  000001  001162'         BIT     #1,APT.MODE             ;                       2597
000162  001405                          BEQ     2$
000164  012777  000001  001164'         MOV     #1,@MAIL.BOX.TESTNUM    ;                       2600
000172  005077  001166'                 CLR     @MAIL.BOX.SUBTST        ;                       2601
000176  104455                  2$:     TRAP    55                      ;                       2604
000200  000024                          .WORD   24
000202  000000G                         .WORD   EGD.20
000204  000000                          .WORD   0
000206  010116                          MOV     R1,(SP)                 ; P.INDEX,*             2605
000210  004737  000000G                 JSR     PC,PUT.PKT
```

K6


```
000214  013716  000000G                        MOV     CCTLR,(SP)              ;                       2606
000220  012746  000006                         MOV     #6,-(SP)
000224  004737  000000G                        JSR     PC,DROP.CTLR
000230  005726                                 TST     (SP)+                   ;                       2594
000232  005000                                 CLR     R0                      ;                       2610
000234  000571                                 BR      13$
000236  004737  000000G               3$:     JSR     PC,WAIT                 ;                       2614
000242  004737  000000G                        JSR     PC,OUT.IODQ             ;                       2615
000246  010037  000000G                        MOV     R0,RP.INDX
000252  010016                                 MOV     R0,(SP)                 ; RP.INDX,*             2616
000254  012746  000054                         MOV     #54,-(SP)
000260  004737  000000G                        JSR     PC,BL$MUL
000264  062700  000000G                        ADD     #RETPKT,R0
000270  010037  000000G                        MOV     R0,RP.ADDR
000274  132760  000360  000002                 BITB    #360,2(R0)              ;                       2618
000302  001404                                 BEQ     4$
000304  013716  000000G                        MOV     RP.INDX,(SP)            ;                       2620
000310  004737  000000G                        JSR     PC,PUT.RETPKT
000314  005726                       4$:     TST     (SP)+                   ;                       2613
000316  013701  000000G                        MOV     RP.ADDR,R1              ;                       2623
000322  005000                                 CLR     R0
000324  126127  000003  000003                 CMPB    3(R1),#3
000332  001002                                 BNE     5$
000334  005200                                 INC     R0
000336  000407                                 BR      6$
000340  132761  000360  000002       5$:     BITB    #360,2(R1)              ;                       2624
000346  001333                                 BNE     3$
000350  105761  000014                         TSTB    14(R1)                  ;                       2625
000354  100330                                 BPL     3$
000356  006000                       6$:     ROR     R0                      ;                       2627
000360  103015                                 BCC     7$
000362  012716  000000G                        MOV     #DBM23,(SP)             ;                       2630
000366  012746  000001                         MOV     #1,-(SP)
000372  010600                                 MOV     SP,R0                   ; SP,*
000374  104417                                 TRAP    17
000376  013716  000000G                        MOV     RP.INDX,(SP)            ;                       2631
000402  004737  000000G                        JSR     PC,PUT.RETPKT
000406  004737  000000V                        JSR     PC,DR.ERR               ;                       2632
000412  000447                                 BR      10$                     ;                       2633
000414  126127  000014  000204       7$:     CMPB    14(R1),#204             ;                       2638
000422  001007                                 BNE     8$
000424  016100  000022                         MOV     22(R1),R0               ;                       2639
000430  042700  177657                         BIC     #177657,R0
000434  020027  000120                         CMP     R0,#120
000440  001437                                 BEQ     11$
000442  013700  000000G               8$:     MOV     CCTLR,R0                ;                       2642
000446  006300                                 ASL     R0
000450  105260  000000G                        INCB    C.ERR.TBL(R0)
000454  032737  000001  001162'               BIT     #1,APT.MODE             ;                       2644
000462  001405                                 BEQ     9$
000464  012777  000001  001164'               MOV     #1,@MAIL.BOX,TESTNUM    ;                       2647
000472  005077  001166'                        CLR     @MAIL.BOX,SUBTST        ;                       2648
000476  104455                       9$:     TRAP    55                      ;                       2651
```

```
000500  000025                          .WORD   25
000502  000000G                         .WORD   EGD.21
000504  000000G                         .WORD   EMS.21
000506  013716  000000G                 MOV     CCTLR,(SP)              ;                       2652
000512  012746  000006                  MOV     #6,-(SP)
000516  004737  000000G                 JSR     PC,DROP.CTLR
000522  013716  000000G                 MOV     RP.INDX,(SP)            ;                       2653
000526  004737  000000G                 JSR     PC,PUT.RETPKT
000532  062706  000010          10$:    ADD     #10,SP                  ;                       2654
000536  000433                          BR      14$                     ;                       2641
000540  016137  000024  000000G  11$:    MOV     24(R1),CMD.TIME         ;                       2658
000546  006337  000000G                 ASL     CMD.TIME
000552  032737  000001  000000G         BIT     #1,SWP.FLAGS            ;                       2660
000560  001411                          BEQ     12$
000562  016116  000024                  MOV     24(R1),(SP)             ;                       2662
000566  012746  000000G                 MOV     #DBM25,-(SP)
000572  012746  000002                  MOV     #2,-(SP)
000576  010600                          MOV     SP,R0                   ; SP,+
000600  104417                          TRAP    17
000602  022626                          CMP     (SP)+,(SP)+
000604  013716  000000G          12$:    MOV     RP.INDX,(SP)            ;                       2668
000610  004737  000000G                 JSR     PC,PUT.RETPKT
000614  012700  000001                  MOV     #1,R0                   ;                       2610
000620  062706  000006          13$:    ADD     #6,SP                   ;                       2592
000624  000401                          BR      15$                     ;                       2570
000626  005000                  14$:    CLR     R0                      ;                       2559
000630  012601                  15$:    MOV     (SP)+,R1
000632  000207                          RTS     PC

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

```
;  2673  1     routine UNIT_INIT : novalue =
;  2674  1
;  2675  1     !+
;  2676  1     !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
;  2677  1     !       (DISK) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
;  2678  1     !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
;  2679  1     !       MESSAGE, AND TO VERIFY THE RESPONSE.
;  2680  1     !
;  2681  1     !       IMPLICIT INPUTS:
;  2682  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;  2683  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;  2684  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;  2685  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  2686  1     !-
;  2687  1
;  2688  2         begin
;  2689  2         local
;  2690  2             MAX0_LBNS : WORD UNSIGNED,              ! UNIT'S MAXIMUM LO WORD LBN
;  2691  2             MAX1_LBNS : WORD UNSIGNED;              ! UNIT'S MAXIMUM HI WORD LBN
;  2692  2
;  2693  2         P_INDEX = GET_PKT (.CCTLR);                 ! GET AN MSCP PACKET
;  2694  2         MSCP_PKT [.P_INDEX, MSGLEN] = SZ_ONL;       ! PACKET SIZE
;  2695  2         MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;       ! SET DISK ADDRESS (RD/RX DISK NUMBER)
;  2696  2         MSCP_PKT [.P_INDEX, OPCODE] = OP_ONL;       ! OPCODE FOR "ONLINE"
;  2697  2         MSCP_PKT [.P_INDEX, DDPAR] = BIT00;         ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
;  2698  2         MSCP_PKT [.P_INDEX, CMD_TYPE] = SEQ_CMD;    ! SEQUENTIAL COMMAND
;  2699  2
;  2700  2         IF SEND (.P_INDEX) eql FAILURE              ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
;  2701  2         then
;  2702  3             begin
;  2703  3             T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  2704  3             CST_ADDR [.CUOFF, D_FATAL] = TRUE;      ! FATAL ERROR
;  2705  3             ERRDF (22, EGD_22, 0);                  !
;  2706  3             DUR [.L$LUN] = DU_ONLINE;               ! SETUP REASON TO DROP UNIT
;  2707  3             DODU (.L$LUN);                          ! DROP UNIT
;  2708  3             PUT_PKT (.P_INDEX);                     ! RETURN PACKET TO POOL
;  2709  3             end
;  2710  2         else
;  2711  3             begin                                   ! OTHERWISE (SEND WAS SUCCESSFUL)
;  2712  3
;  2713  3             do
;  2714  4                 begin
;  2715  4                 WAIT ();                            ! WAIT FOR RETPKT RESPONSE
;  2716  4                 RP_INDX = OUT_IODQ ();              ! GET INDEX OF RETPKT
;  2717  4                 RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;  2718  4
;  2719  4                 if .RP_ADDR [MESTYP] neq MT_SEQ     ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;  2720  4                 then
;  2721  4                     PUT_RETPKT (.RP_INDX);
;  2722  4
;  2723  4                 end
;  2724  3             until (.RP_ADDR [CONID] eql CID_DRIVER) or
;  2725  4                   ((.RP_ADDR [MESTYP] eql MT_SEQ) and
```

```
;    2726  3                  ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2727  3
;    2728  3              if .RP_ADDR [CONID] eql CID_DRIVER      ! IF RETPKT IS FROM "DRIVER"
;    2729  3              then
;    2730  4                  begin
;    2731  4                  PRINTF (DBM26);                     ! "ERROR IN UNIT INIT"
;    2732  4                  DR_ERR ();                          ! DROP CONTROLLER
;    2733  4                  end
;    2734  3              else
;    2735  3
;    2736  4                  if .RP_ADDR [ENDCOD] neq (OP_ONL or OP_END) ! IF RETPKT IS FROM DISK MSCP
;    2737  3                  then
;    2738  4                      begin
;    2739  4                      T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2740  4                      CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2741  4                      ERRDF (23, EGD_23, EMS_21);     ! FATAL ERROR
;    2742  4                      DUR [.L$LUN] = DU_ONLINE;       ! SETUP REASON TO DROP UNIT
;    2743  4                      DODU (.L$LUN);                  ! DROP UNIT
;    2744  4                      end
;    2745  3                  else
;    2746  4                      begin                           ! RETPKT HAS GOOD ENDCODE
;    2747  4                      ST_CODE = .RP_ADDR [STSCOD];    ! GET STATUS CODE
;    2748  4                      SB_CODE = .RP_ADDR [SUBCOD];    ! GET SUB-CODE
;    2749  4
;    2750  4                      if .ST_CODE neq ST_SUC          ! IF STATUS CODE IS NOT SUCCESSFUL
;    2751  4                      then
;    2752  5                          begin
;    2753  5                          T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2754  5                          CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2755  5                          ERRDF (15, EGD_15, EMS_30); ! ONLINE FAILED
;    2756  5                          DUR [.L$LUN] = DU_ONLINE;   ! SET UP REASON FOR DROPPING UNIT
;    2757  5                          DODU (.L$LUN);              ! DROP UNIT
;    2758  5                          end
;    2759  4                      else
;    2760  5                          begin                       ! SUCCESFUL OPERATION
;    2761  5
;    2762  5                          MAX0_LBNS = .RP_ADDR [SIZE0];                       ! LOAD LOWER WORD OF UNIT SIZE
;    2763  5                          MAX1_LBNS = .RP_ADDR [SIZE1];                       ! LOAD UPPER WORD OF UNIT SIZE
;    2764  5
;    2765  6                          if (.MAX0_LBNS eql 0)                               ! THIS SUBTRACTS ONE FROM THE TOTAL
;    2766  5                          then                                                ! BECAUSE EVERYTHING STARTS AT 0
;    2767  6                              begin                                           ! THROUGH (MAXIMUM - 1)
;    2768  6                              MAX0_LBNS = %O'177777';
;    2769  6                              MAX1_LBNS = .MAX1_LBNS - 1;
;    2770  6                              end
;    2771  5                          else
;    2772  5                              MAX0_LBNS = .MAX0_LBNS - 1;
;    2773  5
;    2774  5                          if  (.CST_ADDR [.CUOFF + 2, D_BEG1] gtru .MAX1_LBNS) or         ! THIS SECTION CHECKS TO SEE
 IF LBNs LISTED
;    2775  5
;    2776  6                              ((.CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .MAX1_LBNS) and       ! IN SOFTWARE QUESTIONS WERE
 TO LARGE FOR
;    2777  6
;    2778  6                              (.CST_ADDR [.CUOFF + 1, D_BEG0] gtru (.MAX0_LBNS - 1)))     ! DEVICE SPECIFIED
                                                                                            ! note 1 less then max. or diagnosti
c  will error
```

```
;    2779  5                        then                                                ! operator error
;    2780  6                            begin
;    2781  6                            CST_ADDR [.CUOFF + 2, D_BEG1] = 0;
;    2782  6                            CST_ADDR [.CUOFF + 1, D_BEG0] = 0;              ! change beginning lbn to 0
;    2783  5                            end;
;    2784  5
;    2785  5                         if
;    2786  5                            (.CST_ADDR [.CUOFF + 4, D_END1] gtru .MAX1_LBNS) or
;    2787  5
;    2788  6                            ((.CST_ADDR [.CUOFF + 4, D_END1] eqlu .MAX1_LBNS) and
;    2789  6                            (.CST_ADDR [.CUOFF + 3, D_END0] gtru .MAX0_LBNS))
;    2790  5                         then
;    2791  6                            begin
;    2792  6                            CST_ADDR [.CUOFF + 4, D_END1] = .MAX1_LBNS;
;    2793  6                            CST_ADDR [.CUOFF + 3, D_END0] = .MAX0_LBNS;     ! and ending lbn to max_lbn
;    2794  5                            end;
;    2795  5
;    2796  5
;    2797  5
;    2798  7                            if (((.ENTRY_REASON eql RESTART) or             ! if restart or
;    2799  6                                (.ENTRY_REASON eql START)) and              ! if continue
;    2800  6
;    2801  6                                (.CRN_LOW leq 8) and                        ! and
;    2802  6                                (.CRN_HIGH eql 0))                          ! first initialization
;    2803  6
;    2804  5                            THEN                                            ! intialize block numbers
;    2805  6                                begin
;    2806  6                                BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0];      ! LOAD sequential LBN table
;    2807  6                                BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1];      !
;    2808  6                                TRK_SGN [.L$LUN] = 1;                                       ! POSITIVE TRACKING DIRECTIO
N
;    2809  5                                end;
;    2810  5                                                                            ! THIS SECTION LOADS TYPE INTO  CST TABLE
;    2811  5                            selectoneu .R? ADDR [R_MODEL] of                ! MODEL BYTE TELLS WHAT TYPE OF UNIT IN
;    2812  5                                                                            ! INDENTIFICATION BLOCK
;    2813  5                                set
;    2814  5
;    2815  5                                [#0'6'] :   CST_ADDR [.CUOFF, D_TYPE] = RD_51;      ! RD 51
;    2816  5                                [#0'7'] :   CST_ADDR [.CUOFF, D_TYPE] = RX_50;      ! RX 50
;    2817  5                                [#0'10'] :  CST_ADDR [.CUOFF, D_TYPE] = RD_52;      ! RD 52
;    2818  5
;    2819  6                                [otherwise] : BEGIN
;    2820  6                                              ERRDF (25 ,EGD_24 ,EMS_30);           ! ERROR UNKNOWN DEVICE
;    2821  5                                              END;
;    2822  5                                tes;
;    2823  5
;    2824  5                        if ((.RP_ADDR [0,FLGS] and UF_WPH) eql UF_WPH) and          ! STATUS CODE IS O.K.
;    2825  6                           (.CST_ADDR [.CUOFF, D_PROT] eql UNPROTECTED)
;    2826  5                        then
;    2827  6                            begin
;    2828  6                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2829  6                            CST_ADDR [.CUOFF, D_FATAL] = TRUE;
;    2830  6                            ERRDF (16, EGD_16  EMS_30);                             ! WRITE PROTECT CONFLICT
;    2831  6                            DUR [.L$LUN] = DU_PROTECT;                              ! SET REASON TO DROP UNIT
```

```
;  2832  6                          DODU (.L$LUN);                                  ! DROP UNIT
;  2833  6                          end
;  2834  5                      else
;  2835  6                          begin                                           ! WRITE PROTECT SWITCH IS O.K.
;  2836  6                          CST_ADDR [.CUOFF, D_STAT] = ONLINE;             ! SET ONLINE FLAG
;  2837  6                          CST [.CCTLR, U_CNT] = .CST [.CCTLR, U_CNT] + 1; ! ADD UNIT TO CTRL TABLE
;  2838  5                          end;
;  2839  4                      end;
;  2840  3                  end;                                                    ! IF RETPKT HAS CORRECT ENDCODE
;  2841  3
;  2842  3              PUT_RETPKT (.RP_INDX);
;  2843  2              end;                                                        ! IF SEND WAS SUCCESSFUL
;  2844  2
;  2845  1          end;                                                            ! ROUTINE UNIT-INIT


                                .SBTTL  UNIT.INIT INITIALIZATION TEST ROUTINES
000000  004137  000000G        UNIT.INIT:
                                JSR     R1,$SAVE5                       ;                       2673
000004  024646                  CMP     -(SP),-(SP)
000006  013746  000000G         MOV     CCTLR,-(SP)                     ;                       2693
000012  004737  000000G         JSR     PC,GET.PKT
000016  010037  000000G         MOV     R0,P.INDEX
000022  010016                  MOV     R0,(SP)                         ; P.INDEX,*             2694
000024  012746  000106          MOV     #106,-(SP)
000030  004737  000000G         JSR     PC,BLDMUL
000034  012760  000044  000006G MOV     #44,MSCP.PKT+6(R0)
000042  013760  000000G 000016G MOV     CDISK,MSCP.PKT+16(R0)           ;                       2695
000050  112760  000011  000022G MOVB    #11,MSCP.PKT+22(R0)             ;                       2696
000056  012760  000001  000046G MOV     #1,MSCP.PKT+46(R0)              ;                       2697
000064  112760  000001  000004G MOVB    #1,MSCP.PKT+4(R0)               ;                       2698
000072  013716  000000G         MOV     P.INDEX,(SP)                    ;                       2700
000076  004737  000000G         JSR     PC,SEND
000102  005700                  TST     R0
000104  001033                  BNE     1$
000106  013700  000000G         MOV     T.ADDR,R0                       ;                       2703
000112  105260  000051          INCB    51(R0)
000116  013700  000000G         MOV     CUOFF,R0                        ;                       2704
000122  006300                  ASL     R0
000124  063700  000000G         ADD     CST.ADDR,R0
000130  052710  010000          BIS     #10000,(R0)
000134  104455                  TRAP    55                              ;                       2705
000136  000026                  .WORD   26
000140  000000G                 .WORD   EGD.22
000142  000000                  .WORD   0
000144  013700  000000G         MOV     L$LUN,R0                        ;                       2706
000150  112760  000007  000000G MOVB    #7,DUR(R0)
000156  104451                  TRAP    51                              ;                       2707
000160  013716  000000G         MOV     P.INDEX,(SP)                    ;                       2708
000164  004737  000000G         JSR     PC,PUT.PKT
000170  000137  006330'         JMP     2$                              ;                       2709
000174  004737  000000G 1$:     JSR     PC,WAIT                         ;                       2715
000200  004737  000000G         JSR     PC,OUT.IODQ                     ;                       2716
```

```
000204  010037  000000G                        MOV     R0,RP.INDX
000210  010016                                 MOV     R0,(SP)                 ; RP.INDX,*                     2717
000212  012746  000054                         MOV     #54,-(SP)
000216  004737  000000G                        JSR     PC,BL$MUL
000222  062700  000000G                        ADD     #RETPKT,R0
000226  010037  000000G                        MOV     R0,RP.ADDR
000232  132760  000360  000002                 BITB    #360,2(R0)              ;                               2719
000240  001404                                 BEQ     2$
000242  013716  000000G                        MOV     RP.INDX,(SP)            ;                               2721
000246  004737  000000G                        JSR     PC,PUT.RETPKT
000252  005726                         2$:     TST     (SP)+                   ;                               2714
000254  013702  000000G                        MOV     RP.ADDR,R2              ;                               2724
000260  005000                                 CLR     R0
000262  126227  000003  000003                 CMPB    3(R2),#3
000270  001002                                 BNE     3$
000272  005200                                 INC     R0
000274  000407                                 BR      4$
000276  132762  000360  000002         3$:     BITB    #360,2(R2)              ;                               2725
000304  001333                                 BNE     1$
000306  105762  000014                         TSTB    14(R2)                  ;                               2726
000312  100330                                 BPL     1$
000314  006000                         4$:     ROR     R0                      ;                               2728
000316  103012                                 BCC     5$
000320  012716  000000G                        MOV     #DBM26,(SP)             ;                               2731
000324  012746  000001                         MOV     #1,-(SP)
000330  010600                                 MOV     SP,R0                   ; SP,*
000332  104417                                 TRAP    17
000334  004737  000000V                        JSR     PC,DR.ERR               ;                               2732
000340  000137  006316'                        JMP     23$                     ;                               2730
000344  013705  000000G                 5$:    MOV     CST.ADDR,R5             ;                               2740
000350  013766  000000G 000006                MOV     CUOFF,6(SP)
000356  006366  000006                         ASL     6(SP)
000362  060566  000006                         ADD     R5,6(SP)
000366  126227  000014  000211                 CMPB    14(R2),#211             ;                               2736
000374  001422                                 BEQ     6$
000376  013700  000000G                        MOV     T.ADDR,R0               ;                               2739
000402  105260  000050                         INCB    50(R0)
000406  052776  010000  000006                 BIS     #10000,6(SP)            ;                               2740
000414  104455                                 TRAP    55                      ;                               2741
000416  000027                                 .WORD   27
000420  000000G                                .WORD   EGD,23
000422  000000G                                .WORD   EMS,21
000424  013700  000000G                        MOV     L$LUN,R0                ;                               2742
000430  112760  000007  000000G                MOVB    #7,DUR(R0)
000436  104451                                 TRAP    51                      ;                               2743
000440  000445                                 BR      7$                      ;                               2736
000442  116237  000016  000000G         6$:    MOVB    16(R2),ST.CODE          ;                               2744
000450  042737  177740  000000G                BIC     #177740,ST.CODE
000456  016200  000016                         MOV     16(R2),R0               ;                               2748
000462  006200                                 ASR     R0
000464  006200                                 ASR     R0
000466  006200                                 ASR     R0
000470  006200                                 ASR     R0
```

```
000472  006200                          ASR     R0
000474  042700  174000                  BIC     #174000,R0
000500  010037  000000G                 MOV     R0,SB.CODE
000504  005737  000000G                 TST     ST.CODE                 ;                       2750
000510  001423                          BEQ     8$
000512  013700  000000G                 MOV     T.ADDR,R0               ;                       2753
000516  105260  000050                  INCB    50(R0)
000522  052776  010000  000006         BIS     #10000,@6(SP)           ;                       2754
000530  104455                          TRAP    55                      ;                       2755
000532  000017                          .WORD   17
000534  000000G                         .WORD   EGD.15
000536  000000G                         .WORD   EMS.30
000540  013700  000000G                 MOV     L$LUN,R0                ;                       2756
000544  112760  000007  000000G         MOVB    #7,DUR(R0)
000552  104451                          TRAP    51                      ;                       2757
000554  000137  006320'        7$:     JMP     23$                     ;                       2750
000560  016203  000044        8$:     MOV     44(R2),R3               ; *,MAX0.LBNS           2762
000564  016204  000046                  MOV     46(R2),R4               ; *,MAX1.LBNS           2763
000570  005703                          TST     R3                      ; MAX0.LBNS             2765
000572  001004                          BNE     9$
000574  012703  177777                  MOV     #-1,R3                  ; *,MAX0.LBNS           2768
000600  005304                          DEC     R4                      ; MAX1.LBNS             2769
000602  000401                          BR      10$                     ;                       2765
000604  005303                  9$:     DEC     R3                      ; MAX0.LBNS             2772
000606  013700  000000G         10$:    MOV     CUOFF,R0                ;                       2774
000612  006300                          ASL     R0
000614  060500                          ADD     R5,R0
000616  011766  000004                  MOV     (PC),4(SP)
000622  060066  000004                  ADD     R0,4(SP)
000626  027604  000004                  CMP     @4(SP),R4               ; *,MAX1.LBNS
000632  101012                          BHI     11$
000634  001021                          BNE     12$                     ;                       2776
000636  013700  000000G                 MOV     CUOFF,R0                ;                       2777
000642  006300                          ASL     R0
000644  060500                          ADD     R5,R0
000646  010301                          MOV     R3,R1                   ; MAX0.LBNS,*
000650  005301                          DEC     R1
000652  026001  000002                  CMP     2(R0),R1
000656  101410                          BLOS    12$
000660  005076  000004          11$:    CLR     @4(SP)                  ;                       2781
000664  013700  000000G                 MOV     CUOFF,R0                ;                       2782
000670  006300                          ASL     R0
000672  060500                          ADD     R5,R0
000674  005060  000002                  CLR     2(R0)
000700  013700  000000G         12$:    MOV     CUOFF,R0                ;                       2786
000704  006300                          ASL     R0
000706  060500                          ADD     R5,R0
000710  026004  000010                  CMP     10(R0),R4               ; *,MAX1.LBNS
000714  101010                          BHI     13$
000716  001017                          BNE     14$                     ;                       2788
000720  013701  000000G                 MOV     CUOFF,R1                ;                       2789
000724  006301                          ASL     R1
000726  060501                          ADD     R5,R1
```

```
000730  026103  000006                          CMP     6(R1),R3                        ; ^,MAX0.LBNS
000734  101410                                  BLOS    14$
000736  010460  000010                  13$:    MOV     R4,10(R0)                       ; MAX1.LBNS,^                   2792
000742  013700  000000G                         MOV     CUOFF,R0                        ;                               2793
000746  006300                                  ASL     R0
000750  060500                                  ADD     R5,R0
000752  010360  000006                          MOV     R3,6(R0)                        ; MAX0.LBNS,^
000756  123727  000000G 000002          14$:    CMPB    ENTRY.REASON,#2                 ;                               2798
000764  001404                                  BEQ     15$
000766  123727  000000G 000001                  CMPB    ENTRY.REASON,#1                 ;                               2799
000774  001031                                  BNE     16$
000776  023727  000000G 000010          15$:    CMP     CRN.LOW,#10                     ;                               2801
001004  003025                                  BGT     16$
001006  005737  000000G                         TST     CRN.HIGH                        ;                               2802
001012  001022                                  BNE     16$
001014  013704  000000G                         MOV     L$LUN,R4                        ;                               2806
001020  010400                                  MOV     R4,R0
001022  006300                                  ASL     R0
001024  006300                                  ASL     R0
001026  013701  000000G                         MOV     CUOFF,R1
001032  006301                                  ASL     R1
001034  060501                                  ADD     R5,R1
001036  016160  000002  000000G                 MOV     2(R1),BST(R0)
001044  017660  000004  000002G                 MOV     @4(SP),BST+2(R0)                ;                               2807
001052  112764  000001  000000G                 MOVB    #1,TRK.SGN(R4)                  ;                               2808
001060  005001                          16$:    CLR     R1                              ;                               2811
001062  156201  000032                          BISB    32(R2),R1
001066  020127  000006                          CMP     R1,#6                           ;                               2815
001072  001007                                  BNE     17$
001074  013700  000000G                         MOV     CUOFF,R0
001100  006300                                  ASL     R0
001102  060500                                  ADD     R5,R0
001104  152710  000020                          BISB    #20,(R0)
001110  000421                                  BR      20$                             ;                               2811
001112  020127  000007                  17$:    CMP     R1,#7                           ;                               2816
001116  001403                                  BEQ     18$
001120  020127  000010                          CMP     R1,#10                          ;                               2817
001124  001007                                  BNE     19$
001126  013700  000000G                 18$:    MOV     CUOFF,R0
001132  006300                                  ASL     R0
001134  060500                                  ADD     R5,R0
001136  142710  000020                          BICB    #20,(R0)
001142  000404                                  BR      20$                             ;                               2811
001144  104455                          19$:    TRAP    55                              ;                               2820
001146  000031                                  .WORD   31
001150  0000C0G                                 .WORD   EGD.24
001152  000000G                                 .WORD   EMS.30
001154  032762  020000  000022          20$:    BIT     #20000,22(R2)                   ;                               2824
001162  001430                                  BEQ     21$
001164  013700  000000G                         MOV     CUOFF,R0                        ;                               2825
001170  006300                                  ASL     R0
001172  060500                                  ADD     R5,R0
001174  005710                                  TST     (R0)
```

```
001176  100022                          BPL     21$
001200  013700  000000G                 MOV     T.ADDR,R0               ;                       2828
001204  105260  000050                  INCB    50(R0)
001210  052776  010000  000006          BIS     #10000,@6(SP)           ;                       2829
001216  104455                          TRAP    55                      ;                       2830
001220  000020                          .WORD   20
001222  000000G                         .WORD   EGD.16
001224  000000G                         .WORD   EMS.30
001226  013700  000000G                 MOV     L$LUN,R0                ;                       2831
001232  112760  000011  000000G         MOVB    #11,DUR(R0)
001240  104451                          TRAP    51                      ;                       2832
001242  000414                          BR      23$                     ;                       2824
001244  052776  020000  000006  21$:    BIS     #20000,@6(SP)           ;                       2836
001252  013716  000000G                 MOV     CCTLR,(SP)              ;                       2837
001256  012746  000126                  MOV     #126,-(SP)
001262  004737  000000G                 JSR     PC,BL$MUL
001266  105260  000005G                 INCB    CST+5(R0)
001272  005726                  22$:    TST     (SP)+                   ;                       2835
001274  013716  000000G         23$:    MOV     RP.INDX,(SP)            ;                       2842
001300  004737  000000G                 JSR     PC,PUT.RETPKT
001304  062706  000010          24$:    ADD     #10,SP                  ;                       2673
001310  000207                          RTS     PC

; Routine Size:  357 words,     Routine Base:  $CODE$ + 5024
; Maximum stack depth per invocation:  13 words
```

```
;    2846  1     GLOBAL routine DR_ERR : novalue =
;    2847  1
;    2848  1     !+
;    2849  1     !       THIS ROUTINE IS DESIGNED TO PROCESS RETURN PACKETS THAT ORIGINATE AT
;    2850  1     !       THE "DRIVER" RATHER THAN THE DEVICE. DRIVER-ORIGINATED PACKETS INDICATE
;    2851  1     !       EITHER A FATAL DEVICE ERROR OR A COMMAND TIMEOUT. SINCE THIS ROUTINE IS
;    2852  1     !       ONLY CALLED DURING THE INITIALIZATION TEST, IT TREATS A COMMAND TIMEOUT
;    2853  1     !       AS AN INITIALIZATION ERROR.
;    2854  1     !
;    2855  1     !       IMPLICIT INPUTS:
;    2856  1     !               RP_ADDR - ADDRESS OF A RETPKT THAT ORIGINATED AT THE "DRIVER"
;    2857  1     !                       (I.E., CONNECTION ID = CID_DRIVER)
;    2858  1     !-
;    2859  1
;    2860  2         begin
;    2861  2
;    2862  2         local
;    2863  2             REASON : word initial (DU_TIME);        ! ASSUME COMMAND TIMEOUT
;    2864  2
;    2865  2         if .RP_ADDR [MESTYP] eql MT_FATAL           ! IF FATAL DEVICE ERROR
;    2866  2         then
;    2867  2
;    2868  2         DROP_CTLR (.CCTLR, .REASON);                ! DROP ALL UNITS ON CONTROLLER
;    2869  1         end;


                                        .SBTTL  DR.ERR INITIALIZATION TEST ROUTINES
000000  010146                  DR.ERR::MOV     R1,-(SP)                ;                       2846
000002  012701  000012                  MOV     #12,R1                  ; *,REASON              2860
000006  013700  000000G                 MOV     RP.ADDR,R0              ;                       2865
000012  116000  000002                  MOVB    2(R0),R0
000016  042700  177417                  BIC     #177417,R0
000022  020027  000060                  CMP     R0,#60
000026  001006                          BNE     1$
000030  013746  000000G                 MOV     CCTLR,-(SP)             ;                       2868
000034  010146                          MOV     R1,-(SP)                ; REASON,*
000036  004737  000000G                 JSR     PC,DROP.CTLR
000042  022626                          CMP     (SP)+,(SP)+
000044  012601                  1$:     MOV     (SP)+,R1                ;                       2846
000046  000207                          RTS     PC

; Routine Size:  20 words,      Routine Base:  $CODE$ + 6336
; Maximum stack depth per invocation:  4 words
```

```
;    2870  1     routine ACCESS : novalue =
;    2871  1
;    2872  1     !+
;    2873  1     !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT DISK
;    2874  1     !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;    2875  1     !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE DISK, AND CHECKING
;    2876  1     !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;    2877  1     !
;    2878  1     !       IMPLICIT INPUTS:
;    2879  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    2880  1     !               CDISK - CURRENT DISK ADDRESS (RD/RX DISK NUMBER)
;    2881  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    2882  1     !-
;    2883  1
;    2884  2         begin
;    2885  2
;    2886  2     local
;    2887  2             RESULT : word initial (FAILURE),        ! GUILTY UNTIL PROVEN INNOCENT
;    2888  2             LBN : word,
;    2889  2             PASS : word initial (1);                ! LOOP PASS COUNT
;    2890  2
;    2891  2         ST_CODE = SB_CODE = 0;                      ! STATUS CODE AND SUB-CODE
;    2892  2         LBN = (((.MAX_LBN [.L$LUN] + 1) ^ -1) and %o'77777') - 1;
;    2893  2                                                     ! START WITH LAST LBN ON TOP SURFACE: [(X+1)/2] -1
;    2894  2
;    2895  2         do
;    2896  3             begin                                   ! LOOP STARTS HERE
;    2897  3             P_INDEX = GET_PKT (.CCTLR);             ! GET AN MSCP PACKET
;    2898  3             MSCP_PKT [.P_INDEX, DK_NUM] = .CDISK;   ! SET DISK ADDR (RD/RX DISK NUMBER)
;    2899  3             MSCP_PKT [.P_INDEX, OPCODE] = OP_ACC;   ! ACCESS OPCODE
;    2900  3             MSCP_PKT [.P_INDEX, BC_LO] = 512;       ! BYTE COUNT (1 BLOCK)
;    2901  3             MSCP_PKT [.P_INDEX, LBN_L] = .LBN;      ! LOGICAL BLOCK NUMBER
;    2902  3             MSCP_PKT [.P_INDEX, CMD_TYPE] = NON_SEQ_CMD;    ! NON-SEQUENTIAL COMMAND
;    2903  3
;    2904  3             if SEND (.P_INDEX) eql FAILURE          ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;    2905  3             then
;    2906  4                 begin
;    2907  4                 PUT_PKT (.P_INDEX);                 ! RETURN PACKET TO POOL
;    2908  4                 PASS = 2;                           ! NO MORE TRIES
;    2909  4                 end
;    2910  3             else
;    2911  4                 begin                               ! IF SEND WAS SUCCESSFUL
;    2912  4
;    2913  4                 do
;    2914  5                     begin
;    2915  5                     WAIT ();                        ! WAIT FOR RESPONSE
;    2916  5                     RP_INDX = OUT_IODQ ();          ! GET RETPKT (RESPONSE) INDEX
;    2917  5                     RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
;    2918  5
;    2919  5                     if .RP_ADDR [MESTYP] neq MT_SEQ ! RETURN ALL RETPKTS NOT SENT BY CONTROLLER
;    2920  5                     then
;    2921  5                         PUT_RETPKT (.RP_INDX);
;    2922  5
```

```
;    2923  5                    end
;    2924  4                until (.RP_ADDR [CONID] eql CID_DRIVER) or
;    2925  5                      ((.RP_ADDR [MESTYP] eql MT_SEQ) and
;    2926  4                       ((.RP_ADDR [ENDCOD] and OP_END) eql OP_END));
;    2927  4
;    2928  4                if .RP_ADDR [CONID] eql CID_DRIVER  ! IF RETPKT CAME FROM "DRIVER"
;    2929  4                then
;    2930  4                    PASS = 2                        ! NO MORE TRIES
;    2931  4                else
;    2932  4
;    2933  5                    if .RP_ADDR [ENDCOD] neq (OP_ACC or OP_END)
;    2934  4                    then
;    2935  5                        begin
;    2936  5                        PRINTF (DBM29);             ! "RETPKT HAS BAD ENDCODE"
!ZZZ                            EMSCMD ();
;    2938  5                        end
;    2939  4                    else
;    2940  5                        begin                       ! RETPKT HAS CORRECT ENDCODE
;    2941  5                        ST_CODE = .RP_ADDR [STSCOD];! GET STATUS CODE FROM PACKET
;    2942  5                        SB_CODE = .RP_ADDR [SUBCOD];! GET SUB-CODE FROM PACKET
;    2943  5
;    2944  5                        if .ST_CODE eql ST_SUC      ! IF STATUS CODE INDICATES SUCCESS
;    2945  5                        then
;    2946  6                            begin
;    2947  6                            RESULT = SUCCESS;
;    2948  6                            PASS = 2;               ! NO NEED TO TRY AGAIN
;    2949  5                            end;
;    2950  5
;    2951  4                        end;                        ! IF RETPKT HAS CORRECT ENDCODE
;    2952  4
;    2953  4                        PUT_RETPKT (.RP_INDX);
;    2954  3                        end;                        ! IF SEND WAS SUCCESSFUL
;    2955  3
;    2956  3                LBN = .LBN + 1;                     ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
;    2957  3                PASS = .PASS + 1;                   ! SECOND PASS
;    2958  3                end                                 ! END OF PASS LOOP
;    2959  2            until .PASS gequ 3;
;    2960  2
;    2961  2            if .RESULT eql FAILURE
;    2962  2            then
;    2963  3                begin
;    2964  3                T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;    2965  3                CST_ADDR [.CUOFF, D_FATAL] = TRUE;  ! FATAL ERROR
;    2966  3                ERRDF (17, EGD_17, EMS_30);                 ! ACCESS FAILED
;    2967  3                DUR [.L$LUN] = DU_ACCESS;           ! SET REASON TO DROP UNIT
;    2968  3                DODU (.L$LUN);                      ! DROP UNIT
;    2969  2                end;                                ! IF ACCESS FAILED
;    2970  2
;    2971  1            end;                                    ! ROUTINE ACCESS


                                .SBTTL  ACCESS INITIALIZATION TEST ROUTINES
000000  004137  000000G    ACCESS: JSR     R1,$SAVE4                       ;                                    2870
```

```
000004  005003                          CLR     R3                      ; RESULT                2884
000006  012702  000001                  MOV     #1,R2                   ; *,PASS
000012  005037  000000G                 CLR     SB.CODE                 ;                       2891
000016  005037  000000G                 CLR     ST.CODE
000022  013700  000000G                 MOV     L$LUN,R0                ;                       2892
000026  006300                          ASL     R0
000030  016000  000054'                  MOV     MAX.LBN(R0),R0
000034  060200                          ADD     R2,R0
000036  006200                          ASR     R0
000040  010004                          MOV     R0,R4                   ; *,LBN
000042  042704  100000                  BIC     #100000,R4              ; *,LBN
000046  005304                          DEC     R4                      ; LBN
000050  013746  000000G         1$:     MOV     CCTLR,-(SP)             ;                       2897
000054  004737  000000G                 JSR     PC,GET.PKT
000060  010037  000000G                 MOV     R0,P.INDEX
000064  010016                          MOV     R0,(SP)                 ; P.INDEX,*             2898
000066  012746  000106                  MOV     #106,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  013760  000000G  000016G        MOV     CDISK,MSCP.PKT+16(R0)
000104  112760  000020  000022G        MOVB    #20,MSCP.PKT+22(R0)     ;                       2899
000112  012760  001000  000026G        MOV     #1000,MSCP.PKT+26(R0)   ;                       2900
000120  010460  000046G                 MOV     R4,MSCP.PKT+46(R0)      ; LBN,*                 2901
000124  112760  000002  000004G        MOVB    #2,MSCP.PKT+4(R0)       ;                       2902
000132  013716  000000G                 MOV     P.INDEX,(SP)            ;                       2904
000136  004737  000000G                 JSR     PC,SEND
000142  005700                          TST     R0
000144  001007                          BNE     2$
000146  013716  000000G                 MOV     P.INDEX,(SP)            ;                       2907
000152  004737  000000G                 JSR     PC,PUT.PKT
000156  012702  000002                  MOV     #2,R2                   ; *,PASS                2908
000162  000522                          BR      9$                      ;                       2904
000164  004737  000000G         2$:     JSR     PC,WAIT                 ;                       2915
000170  004737  000000G                 JSR     PC,OUT.IODQ             ;                       2916
000174  010037  000000G                 MOV     R0,RP.INDX
000200  010016                          MOV     R0,(SP)                 ; RP.INDX,*             2917
000202  012746  000054                  MOV     #54,(SP)
000206  004737  000000G                 JSR     PC,BL$MUL
000212  062700  000000G                 ADD     #RETPKT,R0
000216  010037  000000G                 MOV     R0,RP.ADDR
000222  132760  000360  000002         BITB    #360,2(R0)              ;                       2919
000230  001404                          BEQ     3$
000232  013716  000000G                 MOV     RP.INDX,(SP)            ;                       2921
000236  004737  000000G                 JSR     PC,PUT.RETPKT
000242  005726                  3$:     TST     (SP)+                   ;                       2914
000244  013701  000000G                 MOV     RP.ADDR,R1              ;                       2924
000250  005000                          CLR     R0
000252  126127  000003  000003         CMPB    3(R1),#3
000260  001002                          BNE     4$
000262  005200                          INC     R0
000264  000407                          BR      5$
000266  132761  000360  000002         BITB    #360,2(R1)              ;                       2925
000274  001333                          BNE     2$
000276  105761  000014                  TSTB    14(R1)                  ;                       2926
```

```
000302  100330                          BPL     2$
000304  006000                  5$:     ROR     R0                              ;                       2928
000306  103442                          BLO     7$                              ;                       2930
000310  126127  000014  000220          CMPB    14(R1),#220                     ;                       2933
000316  001410                          BEQ     6$
000320  012716  000000G                 MOV     #DBM29,(SP)                     ;                       2936
000324  012746  000001                  MOV     #1,-(SP)
000330  010600                          MOV     SP,R0                           ; SP,*
000332  104417                          TRAP    17
000334  005726                          TST     (SP)+                           ;                       2935
000336  000430                          BR      8$                              ;                       2933
000340  116137  000016  000000G  6$:     MOVB    16(R1),ST.CODE                  ;                       2941
000346  042737  177740  000000G          BIC     #177740,ST.CODE
000354  016100  000016                  MOV     16(R1),R0                       ;                       2942
000360  006200                          ASR     R0
000362  006200                          ASR     R0
000364  006200                          ASR     R0
000366  006200                          ASR     R0
000370  006200                          ASR     R0
000372  042700  174000                  BIC     #174000,R0
000376  010037  000000G                 MOV     R0,SB.CODE
000402  005737  000000G                 TST     ST.CODE                         ;                       2944
000406  001004                          BNE     8$
000410  012703  000001                  MOV     #1,R3                           ; *,RESULT              2947
000414  012702  000002          7$:     MOV     #2,R2                           ; *,PASS                2948
000420  013716  000000G          8$:     MOV     RP.INDX,(SP)                    ;                       2953
000424  004737  000000G                  JSR     PC,PUT.RETPKT
000430  005204                  9$:     INC     R4                              ; LBN                   2956
000432  005202                          INC     R2                              ; PASS                  2957
000434  022626                          CMP     (SP)+,(SP)+                     ;                       2896
000436  020227  000003                  CMP     R2,#3                           ; PASS,*                2959
000442  103602                          BLO     1$
000444  005703                          TST     R3                              ; RESULT                2961
000446  001025                          BNE     10$
000450  013700  000000G                 MOV     T.ADDR,R0                       ;                       2964
000454  105260  000050                  INCB    50(R0)
000460  013700  000000G                 MOV     CUOFF,R0                        ;                       2965
000464  006300                          ASL     R0
000466  063700  000000G                 ADD     CST.ADDR,R0
000472  052710  010000                  BIS     #10000,(R0)
000476  104455                          TRAP    55                              ;                       2966
000500  000021                          .WORD   21
000502  000000G                         .WORD   EGD,17
000504  000000G                         .WORD   EMS,30
000506  013700  000000G                 MOV     T$UCN,R0                        ;                       2967
000512  112760  000010  000000G         MOVB    #10,DUR(R0)
000520  104451                          TRAP    51                              ;                       2968
000522  000207                  10$:    RTS     PC                              ;                       2870

; Routine Size:  170 words,     Routine Base:  $CODE$ + 6406
; Maximum stack depth per invocation:  10 words
```

```
;    2972  1     %sbttl 'MULTI-DRIVE TEST ROUTINES'
;    2973  1
;    2974  1     GLOBAL routine MULTI_DRIVE : novalue =
;    2975  1
;    2976  1     !+
;    2977  1     !       THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;    2978  1     !       MULTI-DRIVE TEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;    2979  1     !       USER AN INDICATION OF HOW ONE OR SEVERAL RDRX DRIVES WOULD PERFORM IN
;    2980  1     !       AN OPERATING SYSTEM ENVIRONMENT.
;    2981  1     !
;    2982  1     !       THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
;    2983  1     !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE TEST DATA, THIS ROUTINE
;    2984  1     !       ENTERS A LOOP WHICH ISSUES QIOs TO ALL ACTIVE CONTROLLERS AND PROCESSES
;    2985  1     !       ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;    2986  1     !       DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;    2987  1     !       LOOP OCCURS WHEN QIOs ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;    2988  1     !       QIOS HAVE COMPLETED.
;    2989  1     !-
;    2990  1
;    2991  2         begin
;    2992  2
;    2993  2         local
;    2994  2             CUR_PRIORITY : word;
;    2995  2
;    2996  2         label
;    2997  2             SEND_COMMANDS;
;    2998  2
;    2999  2         MD_INIT ();                                     ! INIT MULTI-DRIVE TEST DATA
;    3000  2         INIT_OCCURED = TRUE;                            !
;    3001  2
;    3002  3         do  begin                                       ! START OF EXECUTIVE LOOP
;    3003  3
;    3004  3             incr CTLR from 0 to (MAX_CTLR - 1) do       ! FOR EACH CONTROLLER
;    3005  4                 begin
;    3006  4                 SET_CPAR (.CTLR);                       ! SET UP CURRENT CONTROLLER PARAMETERS
;    3007  4                 GETPRI (CUR_PRIORITY);                  !
;    3008  4                 SETPRI (PRI04);                         ! NO INTERRUPTS WHEN EXAMINING SA
;    3009  4                 ICTLR = .CCTLR;                         ! FAKE INTERRUPTING CONTROLLER'S NUMBER
;    3010  4                 ICST_ADDR = .CST_ADDR;                  ! FAKE INTERRUPTING CONTROLLER'S CST ADDR
;    3011  4                 IDCT_ADDR = .DCT_ADDR;                  ! FAKE INTERRUPTING CONTROLLER'S DCT ADDR
;    3012  4                 IRDRX_ADDR = .ICST_ADDR [IP_ADDR];      ! FAKE INTERRUPTING CONTROLLER'S ADDRESS
;    3013  4                 IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];   ! CONTENTS OF THE SA REGISTER
;    3014  4
;    3015  5                 if BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)    ! IF SA SHOWS AN ERROR
;    3016  4                 then
;    3017  5                     begin
;    3018  5                     FATAL_ERROR ();                     ! DECLARE FATAL ERROR
;    3019  5                     SETPRI (.CUR_PRIORITY);             ! LOWER PRIORITY
;    3020  5                     exitloop;                           ! QUIT
;    3021  5                     end
;    3022  4                 else
;    3023  4                     SETPRI (.CUR_PRIORITY);             ! IF NO ERROR, CONTINUE
;    3024  4
```

```
;  3025  4                 if QIO_OK ()                                        ! IF O.K. TO ISSUE QIO(S) TO CONTROLLER
;  3026  4                 then
;  3027  4                     SEND_COMMANDS:
;  3028  5                         begin
;  3029  5                         QIO_GEN ();                                 ! GENERATE 1 OR 2 QIOs
;  3030  5
;  3031  5                         if (.MX1 geq 0) and                         ! IF SUCCESS ON FIRST QIO
;  3032  6                            (not .EOP_FLAG)
;  3033  5                         then
;  3034  5
;  3035  5                             if SEND (.MX1) eql SUCCESS              ! ATTEMPT TO SEND IT. IF SUCCESS
;  3036  5                             then
;  3037  5                                 QIO [.CTLR] = .QIO [.CTLR] + 1      ! INCR OUTSTANDING QIO COUNT
;  3038  5                             else
;  3039  6                                 begin
;  3040  6                                 PUT_PKT (.MX1);                     ! RETURN PACKET TO POOL
;  3041  6                                 leave SEND_COMMANDS;
;  3042  5                                 end;
;  3043  5
;  3044  5                         if (.MX2 geq 0) and                         ! IF SUCCESS ON SECOND QIO
;  3045  6                            (not .EOP_FLAG)
;  3046  5                         then
;  3047  6                             begin
;  3048  6
;  3049  6                             do
;  3050  6                                 BREAK                               ! WAIT TILL 1 MORE SLOT AVAILABLE IN CRING
;  3051  6                             until (.DCT_ADDR [CRING_CNT] lssu CRING_LEN);   !
;  3052  6
;  3053  6                             if SEND (.MX2) eql SUCCESS              ! ATTEMPT TO SEND IT.
;  3054  6                             then
;  3055  6                                 QIO [.CTLR] = .QIO [.CTLR] + 1      ! IF SUCCESS, INCR OUTSTANDING QIO COUNT
;  3056  6                             else
;  3057  7                                 begin
;  3058  7                                 PRINTF (DBM121, .CRN HIGH, .CRN LOW);
;  3059  7                                 COMPARE_DATA = FALSE;               ! NO SENSE IN COMPARING WRITE DATA
;  3060  7                                 PUT_PKT (.MX2);                     ! RETURN PACKET TO POOL
;  3061  6                                 end;
;  3062  5                             end;
;  3063  5
;  3064  4                         end;                                        ! O.K. TO ISSUE QIO(S)
;  3065  3                     end;                                            ! CONTROLLER LOOP
;  3066  3
;  3067  3                 BREAK;                                              ! LET SUPERVISOR CATCH USER REQUESTS
;  3068  3                 PROC_RETPKT ();                                     ! PROCESS ANY RETURN PACKETS
;  3069  3
;  3070  3                 end                                                 ! EXECUTIVE PROCESSING LOOP
;  3071  3             until ((not QIO_OUT ()) or
;  3072  4                   ((.DCT_ADDR [CRING_CNT] eql 0) and
;  3073  2                    (.EOP_FLAG)));
;  3074  2
;  3075  2             DCT_ADDR [IG_INT] = TRUE;                               ! NO FURTHER INTERRUPTS ON THIS CONTROLLER
;  3076  2
;  3077  1             end;                                                    ! EXERCISER
```

```
                                        .SBTTL  MULTI.DRIVE MULTI-DRIVE TEST ROUTINES
000000  004137  000000G             MULTI.DRIVE::
                                        JSR     R1,$SAVE3                       ;                               2974
000004  005746                          TST     -(SP)
000006  004737  000000V                 JSR     PC,MD.INIT                      ;                               2999
000012  112737  000001  000000G         MOVB    #1,INIT.OCCURED                 ;                               3000
000020  005001                  1$:     CLR     R1                              ; CTLR                          3004
000022  010146                  2$:     MOV     R1,-(SP)                        ; CTLR,*                        3006
000024  004737  000000G                 JSR     PC,SET.CPAR
000030  104440                          TRAP    40                              ;                               3007
000032  010003                          MOV     R0,R3                           ; *,CUR.PRIORITY
000034  012700  000200                  MOV     #200,R0                         ;                               3008
000040  104441                          TRAP    41
000042  013737  000000G 000104'         MOV     CCTLR,ICTLR                     ;                               3009
000050  013737  000000G 000076'         MOV     CST.ADDR,ICST.ADDR              ;                               3010
000056  013737  000000G 000100'         MOV     DCT.ADDR,IDCT.ADDR              ;                               3011
000064  017737  000076' 000000G         MOV     @ICST.ADDR,IRDRX.ADDR           ;                               3012
000072  013700  000100'                 MOV     IDCT.ADDR,R0                    ;                               3013
000076  013702  000000G                 MOV     IRDRX.ADDR,R2
000102  016266  000002  000002         MOV     2(R2),2(SP)                     ; *,RC.REG
000110  016660  000002  000002         MOV     2(SP),2(R0)                     ; RC.REG,*
000116  016600  000002                  MOV     2(SP),R0                        ;                               3015
000122  042700  077777                  BIC     #77777,R0
000126  020027  100000                  CMP     R0,#-100000
000132  001006                          BNE     3$
000134  004737  000000V                 JSR     PC,FATAL..ERROR                 ;                               3018
000140  010300                          MOV     R3,R0                           ; CUR.PRIORITY,*                3019
000142  104441                          TRAP    41
000144  005726                          TST     (SP)+                           ;                               3017
000146  000511                          BR      9$
000150  010300                  3$:     MOV     R3,R0                           ; CUR.PRIORITY,*                3023
000152  104441                          TRAP    41
000154  004737  000000V                 JSR     PC,QIO.OK                       ;                               3025
000160  006000                          ROR     R0
000162  103077                          BCC     8$
000164  004737  000000V                 JSR     PC,QIO.GEN                      ;                               3029
000170  013700  000106'                 MOV     MX1,R0                          ;                               3031
000174  002422                          BLT     5$
000176  132737  000001  000000G         BITB    #1,EOP.FLAG                     ;                               3032
000204  001016                          BNE     5$
000206  010016                          MOV     R0,(SP)                         ;                               3035
000210  004737  0000000G                JSR     PC,SEND
000214  020027  000001                  CMP     R0,#1
000220  001003                          BNE     4$
000222  105261  0000000G                INCB    QIO(R1)                         ; *(CTLR)                       3037
000226  000405                          BR      5$                              ;                               3035
000230  013716  000106'          4$:     MOV     MX1,(SP)                        ;                               3040
000234  004737  0000000G                JSR     PC,PUT.PKT
000240  000450                          BR      8$                              ;                               3038
000242  005737  000110'          5$:     TST     MX2                             ;                               3044
000246  002445                          BLT     8$
```

```
000350  132737  000001  000000G          BITB    #1,EOP.FLAG             ;                       3045
000356  001041                           BNE     8$
000360  104422                   6$:     TRAP    22                      ;                       3049
000362  127727  000000G 000004           CMPB    @DCT.ADDR,#4            ;                       3051
000370  103373                           BHIS    6$
000372  013716  000110'                   MOV     MX2,(SP)                ;                       3053
000376  004737  000000G                   JSR     PC,SEND
000402  020027  000001                   CMP     R0,#1
000406  001003                           BNE     7$
000410  105261  000000G                   INCB    QIO(R1)                 ; *(CTLR)               3055
000414  000422                           BR      8$                      ;                       3053
000416  013716  000000G           7$:     MOV     CRN.LOW,(SP)            ;                       3058
000422  013746  000000G                   MOV     CRN.HIGH,-(SP)
000426  012746  000000G                   MOV     #DBM121,-(SP)
000432  012746  000003                   MOV     #3,-(SP)
000436  010600                           MOV     SP,R0                   ; SP,*
000440  104417                           TRAP    17
000442  105037  001170'                   CLRB    COMPARE.DATA            ;                       3059
000446  013716  000110'                   MOV     MX2,(SP)                ;                       3060
000452  004737  000000G                   JSR     PC,PUT.PKT
000456  062706  000006                   ADD     #6,SP                   ;                       3057
000462  005726                   8$:     TST     (SP)+                   ;                       3005
000464  005201                           INC     R1                      ; CTLR                  3004
000466  000243                           .WORD CLV!CLC

000370  003614                           BLE     2$
000372  104422                   9$:     TRAP    22                      ;                       3065
000374  004737  000000V                   JSR     PC,PROC.RETPKT          ;                       3068
000400  004737  000000V                   JSR     PC,QIO.OUT              ;                       3071
000404  006000                           ROR     R0
000406  103011                           BCC     10$
000410  105777  000000G                   TSTB    @DCT.ADDR               ;                       3072
000414  001201                           BNE     1$
000416  132737  000001  000000G          BITB    #1,EOP.FLAG             ;                       3073
000424  001002                           BNE     10$
000426  000137  007152'                   JMP     1$
000432  052777  040000  000000G   10$:    BIS     #40000,@DCT.ADDR        ;                       3075
000440  005726                           TST     (SP)+                   ;                       2974
000442  000207                           RTS     PC

; Routine Size:  146 words,    Routine Base:  $CODE$ + 7132
; Maximum stack depth per invocation:  11 words
```

```
;    3078  1     GLOBAL routine MD_INIT : novalue =
;    3079  1
;    3080  1     !+
;    3081  1     !       THIS ROUTINE IS CALLED BY ROUTINE MULTI_DRIVE TO INITIALIZE DATA ITEMS
;    3082  1     !       USED BY THE MULTI-DRIVE TEST.
;    3083  1     !-
;    3084  1
;    3085  2         begin
;    3086  2
;    3087  2     !!ZZZ     local
;    3088  2     !!ZZZ     AVG_XFER_SIZE : word,                                   ! SIZE (BYTES) OF AN AVERAGE I/O xFER
;    3089  2     !!ZZZ     QUICK_PASS_CNT : word;                                  ! AVG NO. OF I/O OPERATIONS IN A QUICK PASS
;    3090  2
;    3091  2         if not .INIT_OCCURED                                        ! IF THIS IS A START
;    3092  2         then
;    3093  2             INIT_IO_BUFF ();                                        ! PARTITION FREE MEMORY INTO I/O BUFFERS
;    3094  2
;    3095  2         if (.ENTRY_REASON neq CONT) and                             ! IF START, RESTART, OR PWR FAIL
;    3096  3            (.ENTRY_REASON neq NEW_PASS)
;    3097  2         then
;    3098  2
;    3099  2             incr CTLR from 0 to (MAX_CTLR - 1) do
;    3100  3                 begin
;    3101  3                 SET_CPAR (.CTLR);
;    3102  3
;    3103  4                 INCR DISK FROM (0 + OF_UN) TO (3 * UNIT_SIZE                !ZZZ
;    3104  3                     + OF_UN) BY UNIT_SIZE DO                            !ZZZ
;    3105  4                     BEGIN                                               !ZZZ
;    3106  4                     SET_UPAR (.DISK);                                   !ZZZ
;    3107  4                     DPST [.L$LUN) = DP_CNT;         !INIT DATA PTRN SEQ TABLE!ZZZ
;    3108  3                     END;                                                !ZZZ
;    3109  3
;    3110  2                 END;                                                    !ZZZ
;    3111  2             INCR COUNT FROM 0 TO (QIO_PER_CTLR * MAX_CTLR - 1) DO   !INIT   !ZZZ
;    3112  2                 BUFF_OWN [.COUNT] = -1;                 !I/O BUFF ALLOC TABLE   !ZZZ
;    3113  1             END;                                        !END MD_INIT            !ZZZ


                                    .SBTTL  MD.INIT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          MD.INIT::
                                    JSR     R1,$SAVE2                       ;                       3078
000004  132737  000001  000000G        BITB    #1,INIT.OCCURED                 ;                       3091
000012  001002                      BNE     1$
000014  004737  000000V            JSR     PC,INIT.IO.BUFF                 ;                       3093
000020  123727  000000G 000003  1$: CMPB    ENTRY.REASON,#3                 ;                       3095
000026  001433                      BEQ     4$
000030  123727  000000G 000005      CMPB    ENTRY.REASON,#5                 ;                       3096
000036  001427                      BEQ     4$
000040  005002                      CLR     R2                              ; CTLR                  3099
000042  010246              2$:     MOV     R2,-(SP)                        ; CTLR,*                3101
000044  004737  000000G            JSR     PC,SET.CPAR
000050  012701  000003            MOV     #3,R1                           ; *,DISK                3103
000054  010116              3$:     MOV     R1,(SP)                         ; DISK,*                3106
```

```
000056  004737  0000006                         JSR     PC,SET.UPAR
000062  013700  000000G                         MOV     L$LUN,R0                ;                                          3107
000066  112760  000025  000050'                 MOVB    #25,DPST(R0)
000074  062701  000012                          ADD     #12,R1                  ; *,DISK                                   3103
000100  020127  000041                          CMP     R1,#41                  ; DISK,*
000104  003763                                  BLE     3$
000106  005726                                  TST     (SP)+                   ;                                          3100
000110  005202                                  INC     R2                      ; CTLR                                     3099
000112  000243                                  .WORD CLV!CLC

000114  003752                                  BLE     2$
000116  005000                          4$:     CLR     R0                      ; COUNT                                    3111
000120  112760  000377  000000G         5$:     MOVB    #377,BUFF.OWN(R0)       ; *,*(COUNT)                               3112
000126  005200                                  INC     R0                      ; COUNT                                    3111
000130  020027  000007                          CMP     R0,#7                   ; COUNT,*
000134  003771                                  BLE     5$
000136  000207                                  RTS     PC                      ;                                          3078

; Routine Size:  48 words,      Routine Base:  $CODE$ + 7576
; Maximum stack depth per invocation:  5 words


;   3114  1
```

```
;    3115  1     GLOBAL routine INIT_IO_BUFF : novalue =
;    3116  1
;    3117  1     !+
;    3118  1     !       THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE TEST IS FIRST
;    3119  1     !       STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;    3120  1     !       COLLECTION OF I/O BUFFERS. THE SIZE OF EACH I/O BUFFER IS DETERMINED
;    3121  1     !       BY A NUMBER OF FACTORS, INCLUDING THE NUMBER OF UNITS, THE NUMBER OF
;    3122  1     !       CONTROLLERS, AND THE SIZE OF FREE MEMORY.
;    3123  1     !
;    3124  1     !       ONCE THE BUFFER SIZE IS DETERMINED, THE NUMBER OF I/O BUFFERS IS
;    3125  1     !       CALCULATED. FINALLY, THE BUFFER ADDRESS (BUFF_ADDR) TABLE IS LOADED
;    3126  1     !       WITH FIXED BUFFER DESCRIPTORS THAT ARE USED IN THE ALLOCATION AND
;    3127  1     !       DEALLOCATION PROCESS.
;    3128  1     !
;    3129  1     !       IMPLICIT INPUTS:
;    3130  1     !               CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
;    3131  1     !               L$UNIT - THE NUMBER OF UNITS AVAILABLE FOR TESTING
;    3132  1     !               FREE_MEM_ADDR - START OF FREE MEMORY
;    3133  1     !-
;    3134  1
;    3135  2         begin
;    3136  2         BUFF_ADDR [0] = (.FREE_MEM_ADDR + 2 + 1) and %o'177776';                ! START OF READ/WRITE BUFFERS
;    3137  2
;    3138  2         while (.BUFF_ADDR [0] and %o'37') neq 0 do                              ! FORCE FIRST I/O BUFFER TO START
;    3139  2             BUFF_ADDR [0] = .BUFF_ADDR [0] + 2;                                 ! ON EVEN BOUNDARY
;    3140  2
;    3141  2         BYTS_PER_QIO = ((.DRS_START - .BUFF_ADDR [0]) / (QIO_PER_CTLR * MAX_CTLR)) and %o'177740';
;    3142  2                                                                                 ! MAX TRANSFER SIZE
;    3143  2
;    3144  2         if .BYTS_PER_QIO gtru MAX_XFER_SIZE
;    3145  2         then
;    3146  2             BYTS_PER_QIO = MAX_XFER_SIZE;                                       ! ADJUST TRANSFER SIZE LOWER
;    3147  2
;    3148  2         if .BYTS_PER_QIO lssu 32
;    3149  2         then
;    3150  3             begin
;    3151  3             ERRSF (2, EGS_02, 0);                                           ! ERROR IF NOT ENOUGH MEMORY
;    3152  3             DOCLN;
;    3153  2             end;
;    3154  2
;    3155  2         if (QIO_PER_CTLR * MAX_CTLR) gtru 1
;    3156  2         then
;    3157  2
;    3158  2             incr index from 1 to (QIO_PER_CTLR * MAX_CTLR - 1) do               ! INIT REMAINING TABLE ENTRIES
;    3159  2                 BUFF_ADDR [.index] = .BUFF_ADDR [.index - 1] + .BYTS_PER_QIO;   ! FIXED BUFFER ADDRESS
;    3160  2
;    3161  1         end;                                                                    ! ROUTINE INIT_IO_BUFF


                                .SBTTL  INIT.IO.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G    INIT.IO.BUFF::
                            JSR     R1,$SAVE3                       ;                   3115
000004  013700  000000G            MOV     FREE.MEM.ADDR,R0                ;                   3136
```

```
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$:     BIT     #37,BUFF.ADDR                   ;                       3138
000034  001404                            BEQ     2$
000036  062737  000002  000000G           ADD     #2,BUFF.ADDR                    ;                       3139
000044  000770                            BR      1$                              ;                       3138
000046  013746  001160'            2$:     MOV     DRS.START,-(SP)                 ;                       3141
000052  163716  000000G                   SUB     BUFF.ADDR,(SP)
000056  012746  000010                    MOV     #10,-(SP)
000062  004737  000000G                   JSR     PC,BL$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              ;                       3144
000106  101403                            BLOS    3$
000110  012737  001400  000000G           MOV     #1400,BYTS.PER.QIO              ;                       3146
000116  023727  000000G 000040    3$:     CMP     BYTS.PER.QIO,#40                ;                       3148
000124  103005                            BHIS    4$
000126  104454                            TRAP    54                              ;                       3151
000130  000002                            .WORD   2
000132  000000G                           .WORD   EGS.02
000134  000000                            .WORD   0
000136  104444                            TRAP    44
000140  012702  000001            4$:     MOV     #1,R2                           ; *,INDEX               3155
000144  010200                    5$:     MOV     R2,R0                           ; INDEX,*               3159
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                 3155
000172  020227  000007                    CMP     R2,#7                           ; INDEX,*
000176  003762                            BLE     5$
000200  022626                            CMP     (SP)+,(SP)+                     ;                       3135
000202  000207                            RTS     PC                              ;                       3115

; Routine Size:  66 words,      Routine Base:  $CODE$ + 7736
; Maximum stack depth per invocation:  8 words
```

```
;    3162  1     GLOBAL routine QIO_OK =
;    3163  1
;    3164  1
;    3165  1     !+
;    3166  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
;    3167  1     !       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
;    3168  1     !       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
;    3169  1     !       THE CONTROLLER MEETS 3 REQUIREMENTS:
;    3170  1     !
;    3171  1     !               A. THE CONTROLLER IS ONLINE;
;    3172  1     !               B. THE NUMBER OF OUTSTANDING QIOs IS AT LEAST 2 LESS THAN THE
;    3173  1     !                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
;    3174  1     !               C. THERE IS AT LEAST ONE DISK ONLINE TO THE CONTROLLER.
;    3175  1     !
;    3176  1     !       IF ANY OF THESE TEST FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
;    3177  1     !
;    3178  1     !       IMPLICIT INPUTS:
;    3179  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3180  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3181  1     !-
;    3182  1
;    3183  1
;    3184  1         if (.CST_ADDR [STATE] eql ONLINE) and               ! IF CONTROLLER IS ONLINE
;    3185  1            (not .EOP_FLAG) and
;    3186  1            ((.QIO [.CCTLR] + 2) lequ QIO_PER_CTLR) and      ! IF OUTSTANDING QIO COUNT IS O.K.
;    3187  2            (.CST_ADDR [U_CNT] neq 0)                        ! IF THERE IS VALID UNIT
;    3188  2
;    3189  1         then
;    3190  1             return TRUE                                     ! "TRUE" EXIT POINT
;    3191  1
;    3192  1         else
;    3193  1             return FALSE;                                   ! "FALSE" EXIT POINT

                                        .SBTTL  QIO.OK MULTI-DRIVE TEST ROUTINES
000000  013700  000000G                QIO.OK::MOV     CST.ADDR,R0                     ;                       3184
000004  005760  000002                        TST     2(R0)
000010  100027                                BPL     1$
000012  132737  000001  000000G               BITB    #1,EOP.FLAG                     ;                       3185
000020  001023                                BNE     1$
000022  013700  000000G                       MOV     CCTLR,R0                        ;                       3186
000026  116000  000000G                       MOVB    QIO(R0),R0
000032  042700  177400                        BIC     #177400,R0
000036  062700  000002                        ADD     #2,R0
000042  020027  000010                        CMP     R0,#10
000046  101010                                BHI     1$
000050  013700  000000G                       MOV     CST.ADDR,R0                     ;                       3187
000054  105760  000005                        TSTB    5(R0)
000060  001403                                BEQ     1$
000062  012700  000001                        MOV     #1,R0                           ;                       3193
000066  000207                                RTS     PC
000070  005000                        1$:     CLR     R0
000072  000207                                RTS     PC                              ;                       3162
```

```
; Routine Size:  30 words,      Routine Base:  $CODE$ + 10142
; Maximum stack depth per invocation:  0 words

;   3194  1
```

```
;    3195  1     GLOBAL routine QIO_OUT =
;    3196  1
;    3197  1     !+
;    3198  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
;    3199  1     !       END OF THE MULTI-DRIVE TEST. ITS PURPOSE IS TO EXAMINE THE QIO VECTOR
;    3200  1     !       FOR ANY OUTSTANDING QIOs ON ANY CONTROLLER. A VALUE OF "TRUE" IS
;    3201  1     !       RETURNED IF THERE IS AT LEAST ONE QIO OUTSTANDING ON ANY CONTROLLER.
;    3202  1     !       OTHERWISE, "FALSE" IS RETURNED INDICATING NO OUTSTANDING QIOs.
;    3203  1     !
;    3204  1
;    3205  2         begin
;    3206  2
;    3207  2         incr CTLR from 0 to (MAX_CTLR - 1) do
;    3208  3             begin
;    3209  3             SET_CPAR (.CTLR);                       ! SET UP CURRENT CONTROLLER PARAMETERS
;    3210  3
;    3211  3             if .CST_ADDR [STATE] eql ONLINE         ! IF CONTROLLER IS ONLINE
;    3212  3             then
;    3213  3                 return TRUE;
;    3214  3
;    3215  2             end;
;    3216  2
;    3217  2         return FALSE;                               ! EXIT - NO CONTROLLERS ONLINE
;    3218  1         end;


                                        .SBTTL  QIO.OUT MULTI-DRIVE TEST ROUTINES
000000  010146                    QIO.OUT::
                                    MOV     R1,-(SP)            ;                       3195
000002  005001                      CLR     R1                  ; CTLR                  3207
000004  010146                1$:   MOV     R1,-(SP)            ; CTLR,*                3209
000006  004737  000000C             JSR     PC,SET.CPAR
000012  013700  000000G             MOV     CST.ADDR,R0         ;                       3211
000016  005760  000002              TST     2(R0)
000022  100004                      BPL     2$
000024  005726                      TST     (SP)+               ;                       3213
000026  012700  000001              MOV     #1,R0
000032  000405                      BR      3$
000034  005726                2$:   TST     (SP)+               ;                       3208
000036  005201                      INC     R1                  ; CTLR                  3207
000040  000243                      .WORD CLV!CLC

000042  003760                      BLE     1$
000044  005000                      CLR     R0                  ;                       3205
000046  012601                3$:   MOV     (SP)+,R1            ;                       3195
000050  000207                      RTS     PC

; Routine Size:  21 words,    Routine Base:  $CODE$ + 10236
; Maximum stack depth per invocation:  3 words
```

```
;    3219  1     GLOBAL routine QIO_GEN : novalue =
;    3220  1
;    3221  1     !+
;    3222  1     !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR AN ONLINE
;    3223  1     !       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
;    3224  1     !       RESPONSIBLE FOR SECURING ONE OR TWO MSCP PACKETS AND LOADING THEM
;    3225  1     !       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
;    3226  1     !       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
;    3227  1     !       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
;    3228  1     !
;    3229  1     !       EACH FIELD OF THE PACKET(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
;    3230  1     !       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
;    3231  1     !       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
;    3232  1     !
;    3233  1     !       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
;    3234  1     !       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
;    3235  1     !       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
;    3236  1     !       GENERATED WITH THE SAME LBN AND BYTE COUNT.
;    3237  1     !
;    3238  1     !       AFTER THE PACKET(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
;    3239  1     !       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
;    3240  1     !       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
;    3241  1     !       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
;    3242  1     !       CONTAIN VALID MSCP PACKET INDECES, OR -1.
;    3243  1     !
;    3244  1     !       IMPLICIT INPUTS:
;    3245  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    3246  1     !-
;    3247  1
;    3248  2         begin
;    3249  2         MX2 = -1;                                   ! ASSUME FAILURE IN SECURING 2ND PACKET
;    3250  2
;    3251  2         if (MX1 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 1ST PACKET. IF FAILURE
;    3252  2         then
;    3253  2             return;                                 ! NO POINT IN CONTINUING
;    3254  2
;    3255  2         if (MX2 = GET_PKT (.CCTLR)) lss 0           ! TRY TO GET 2ND PACKET. IF FAILURE
;    3256  2         then
;    3257  3             begin
;    3258  3             PUT_PKT (.MX1);                         ! RETURN 1ST PACKET TO POOL
;    3259  3             MX1 = -1;                               ! INDICATE FAILURE
;    3260  3             return;                                 ! DONE
;    3261  2             end;
;    3262  2
;    3263  2         MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);     ! CALCULATE STARTING ADDRESSES
;    3264  2         MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);     !     OF BOTH PACKETS
;    3265  2         GET_RANDOM ();                              ! GENERATE A SET OF RANDOM NUMBERS
;    3266  2         QIO_UNIT ();                                ! LOAD RANDOM UNIT NUMBER INTO PACKETS
;    3267  2
;    3268  2         if .EOP_FLAG                                ! RETURN IF NO UNIT ONLINE
;    3269  2         then
;    3270  2             return;
;    3271  2
```

```
;    3272  2          QIO_FUNC ();                          ! LOAD RANDOM FUNCTION CODE (OPCODE)
;    3273  2          QIO_LBN ();                           ! LOAD LBN (RANDOM OR SEQUENTIAL)
;    3274  2          QIO_SIZE ();                          ! LOAD RANDOM BYTE COUNT
;    3275  2          GET_IO_BUFF (MAD1 [BUF_0]);           ! TRY TO GET AN I/O BUFFER
;    3276  2
;    3277  2          if .MX2 geq 0                         ! IF TWO QIOs ARE TO BE ISSUED
;    3278  2          then
;    3279  3              begin
;    3280  3              GET_IO_BUFF (MAD2 [BUF_0]);       ! TRY TO GET 2ND I/O BUFFER
;    3281  3
;    3282  3              if .MAD2 [BUF_0] eqla 0           ! IF 2ND BUFFER ALLOCATION FAILED
;    3283  3              then
;    3284  4                  begin
;    3285  4
;    3286  4                  if .MAD1 [BUF_0] neqa 0       ! IF 1ST I/O BUFFER WAS ALLOCATED
;    3287  4                  then
;    3288  5                      begin
;    3289  5                      PUT_IO_BUFF (MAD1 [BUF_0]);   ! RETURN 1ST I/O BUFFER TO POOL
;    3290  5                      MAD1 [BUF_0] = 0;             ! MARK IT AS FAILED
;    3291  4                      end;
;    3292  4
;    3293  4                  PUT_PKT (.MX2);               ! RETURN 2ND PACKET TO POOL
;    3294  4                  MX2 = -1;                     ! INDICATE FAILURE
;    3295  3                  end;                          ! IF 2ND I/O BUFFER ALLOCATION FAILED
;    3296  3
;    3297  2              end;                              ! IF TWO QIOs ARE TO BE ISSUED
;    3298  2
;    3299  2          if .MAD1 [BUF_0] eqla 0               ! IF 1ST I/O BUFFER ALLOCATION FAILED
;    3300  2          then
;    3301  3              begin
;    3302  3              PUT_PKT (.MX1);                   ! RETURN 1ST PACKET TO POOL
;    3303  3              MX1 = -1;                         ! INDICATE FAILURE
;    3304  3              end
;    3305  2          else
;    3306  2
;    3307  2              if .MAD1 [OPCODE] eql OP_WRT      ! OTHERWISE, IF 1ST OPCODE IS A WRITE  (ALL IS O.K.)
;    3308  2              then
;    3309  2                  FILL_BUFF ();                 ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;    3310  2
;    3311  1          end;                                  ! ROUTINE QIO_GEN

                                        .SBTTL  QIO.GEN MULTI-DRIVE TEST ROUTINES
000000  012737  177777  000110'  QIO.GEN::
                                        MOV     #-1,MX2                 ;                       3249
000006  013746  000000G                 MOV     CCTLR,-(SP)             ;                       3251
000012  004737  000000G                 JSR     PC,GET.PKT
000016  010037  000106'                 MOV     R0,MX1
000022  005726                          TST     (SP)+
000024  005700                          TST     R0                      ; MX1
000026  002563                          BLT     6$                      ;                       3253
000030  013746  000000G                 MOV     CCTLR,-(SP)             ;                       3255
000034  004737  000000G                 JSR     PC,GET.PKT
```

```
000040  010037  000110'                        MOV     R0,MX2
000044  005726                                 TST     (SP)+
000046  005700                                 TST     R0                      ; MX2
000050  002011                                 BGE     1$
000052  013746  000106'                        MOV     MX1,-(SP)               ;                       3258
000056  004737  000000G                        JSR     PC,PUT.PKT
000062  012737  177777  000106'                MOV     #-1,MX1                 ;                       3259
000070  005726                                 TST     (SP)+                   ;                       3260
000072  000207                                 RTS     PC                      ;                       3257
000074  013746  000106'             1$:        MOV     MX1,-(SP)               ;                       3263
000100  012746  000106                         MOV     #106,-(SP)
000104  004737  000000G                        JSR     PC,BL$MUL
000110  062700  000000G                        ADD     #MSCP.PKT,R0
000114  010037  000112'                        MOV     R0,MAD1
000120  013716  000110'                        MOV     MX2,(SP)                ;                       3264
000124  012746  000106                         MOV     #106,-(SP)
000130  004737  000000G                        JSR     PC,BL$MUL
000134  062700  000000G                        ADD     #MSCP.PKT,R0
000140  010037  000114'                        MOV     R0,MAD2
000144  004737  000000V                        JSR     PC,GET.RANDOM           ;                       3265
000150  004737  000000V                        JSR     PC,QIO.UNIT             ;                       3266
000154  132737  000001  000000G               BITB    #1,EOP.FLAG             ;                       3268
000162  001103                                 BNE     5$                      ;                       3219
000164  004737  000000V                        JSR     PC,QIO.FUNC             ;                       3272
000170  004737  000000V                        JSR     PC,QIO.LBN              ;                       3273
000174  004737  000000V                        JSR     PC,QIO.SIZE             ;                       3274
000200  013716  000112'                        MOV     MAD1,(SP)               ;                       3275
000204  062716  000032                         ADD     #32,(SP)
000210  004737  000000G                        JSR     PC,GET.IO.BUFF
000214  005737  000110'                        TST     MX2                     ;                       3277
000220  002437                                 BLT     3$
000222  013716  000114'                        MOV     MAD2,(SP)               ;                       3280
000226  062716  000032                         ADD     #32,(SP)
000232  004737  000000G                        JSR     PC,GET.IO.BUFF
000236  013700  000114'                        MOV     MAD2,R0                 ;                       3282
000242  005760  000032                         TST     32(R0)
000246  001024                                 BNE     3$
000250  013700  000112'                        MOV     MAD1,R0                 ;                       3286
000254  062700  000032                         ADD     #32,R0
000260  005710                                 TST     (R0)
000262  001407                                 BEQ     2$
000264  010016                                 MOV     R0,(SP)                 ;                       3289
000266  004737  000000G                        JSR     PC,PUT.IO.BUFF
000272  013700  000112'                        MOV     MAD1,R0                 ;                       3290
000276  005060  000032                         CLR     32(R0)
000302  013716  000110'             2$:        MOV     MX2,(SP)                ;                       3293
000306  004737  000000G                        JSR     PC,PUT.PKT
000312  012737  177777  000110'                MOV     #-1,MX2                 ;                       3294
000320  013700  000112'             3$:        MOV     MAD1,R0                 ;                       3299
000324  005760  000032                         TST     32(R0)
000330  001010                                 BNE     4$
000332  013716  000106'                        MOV     MX1,(SP)                ;                       3302
000336  004737  000000G                        JSR     PC,PUT.PKT
```

```
000342  012737  177777  000106'                    MOV     #-1,MX1                         ;                3303
000350  000410                                     BR      5$                              ;                3299
000352  013700  000112'                  4$:       MOV     MAD1,R0                         ;                3307
000356  126027  000022  000042                     CMPB    22(R0),#42
000364  001002                                     BNE     5$
000366  004737  000000V                            JSR     PC,FILL.BUFF                    ;                3309
000372  062706  000006                  5$:       ADD     #6,SP                           ;                3243
000376  000207                        6$:         RTS     PC                              ;                3219

; Routine Size:  128 words,     Routine Base:  $CODE$ + 10310
; Maximum stack depth per invocation:  4 words
```

```
;    3312  1     GLOBAL routine GET_RANDOM : novalue =
;    3313  1
;    3314  1     !+
;    3315  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;    3316  1     !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;    3317  1     !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;    3318  1     !       OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;    3319  1     !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;    3320  1     !-
;    3321  1
;    3322  2         begin
;    3323  2
;    3324  2         own
;    3325  2             SEED : word initial (173),
;    3326  2             NEXT_RANDNUM : word initial (245);
;    3327  2
;    3328  2         incr COUNT from 0 to (RDM_LEN - 1) do
;    3329  3             begin
;    3330  3             SEED = (.SEED + .NEXT_RANDNUM + 1) * 4;
;    3331  3             NEXT_RANDNUM = (.NEXT_RANDNUM / 4) + .SEED;
;    3332  3             RANDOM [.COUNT] = .NEXT_RANDNUM;
;    3333  2             end;
;    3334  2
;    3335  1         end;

001200                          .PSECT  $GGG$,  RO
001200  000255         SEED:    .WORD   255
001202  000365         NEXT.RANDNUM:
                                .WORD   365


                                .SBTTL  GET.RANDOM MULTI DRIVE TEST ROUTINES
010710                          .PSECT  $CODE$,  RO

000000  004137  0000006  GET.RANDOM::
                                JSR     R1,$SAVE3               ;                       3312
000004  013705  001200'         MOV     SEED,R3                 ;                       3330
000010  013702  001202'         MOV     NEXT.RANDNUM,R2
000014  005001                  CLR     R1                      ; COUNT                 3328
000016  010200         1$:      MOV     R2,R0                   ;                       3330
000020  060300                  ADD     R3,R0
000022  006300                  ASL     R0
000024  006300                  ASL     R0
000026  010037  001200'         MOV     R0,SEED
000032  062737  000004  001200' ADD     #4,SEED
000040  010246                  MOV     R2,-(SP)                ;                       3331
000042  012746  000004          MOV     #4,-(SP)
000046  004737  0000006         JSR     PC,@#$DIV
000052  013703  001200'         MOV     SEED,R3
000056  060300                  ADD     R3,R0
000060  010037  001202'         MOV     R0,NEXT.RANDNUM
```

```
000064  010002                          MOV     R0,R2                   ; NEXT,RANDNUM,*                3332
000066  010261  0000006                 MOV     R2,RANDOM(R1)           ; *,*(COUNT)
000072  022626                          CMP     (SP)+,(SP)+             ;                               3329
000074  062701  000002                  ADD     #2,R1                   ; *,COUNT                       3328
000100  020127  000036                  CMP     R1,#36                  ; COUNT,*
000104  003744                          BLE     1$
000106  000207                          RTS     PC                      ;                               3312

; Routine Size:  36 words,      Routine Base:  $CODE$ + 10710
; Maximum stack depth per invocation:  7 words
```

```
;    3336  1      GLOBAL routine QIO_UNIT : novalue =
;    3337  1
;    3338  1      !+
;    3339  1      !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;    3340  1      !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;    3341  1      !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;    3342  1      !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;    3343  1      !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;    3344  1      !
;    3345  1      !       IMPLICIT INPUTS:
;    3346  1      !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3347  1      !
;    3348  1      !       IMPLICIT OUTPUTS:
;    3349  1      !               THE RD/RX DISK NUMBER (DISK ADDRESS) IS LOADED INTO THE
;    3350  1      !                       APPROPRIATE FIELD OF BOTH MSCP PACKETS.
;    3351  1      !-
;    3352  1
;    3353  2          begin
;    3354  2
;    3355  2          local
;    3356  2              MOD_COUNT : byte,
;    3357  2              TBL_COUNT : byte,
;    3358  2              SELECT_RD : byte initial (byte (TRUE)),
;    3359  2              RD_COUNT : word initial (0),
;    3360  2              RX_COUNT : word initial (0);
;    3361  2
;    3362  2      !
;    3363  2      ! THE UNITS WILL BE SELECTED ON AN ADJUSTABLE RATIO, RD51/52 TO RX50,
;    3364  2      ! SELECTED VIA THE SOFTWARE PARAMETERS
;    3365  2      !
;    3366  2      ! THIS MODE IS FOR SELECTING DEVICES ON THE FOLLOWING SCHEME:
;    3367  2      ! CHOOSE A DEVICE AND KEEP IT SELECTED FOR A CONSTANT TIME, THEN
;    3368  2      ! MOVE TO THE NEXT. THIS IS NON-RANDOM, FIXED SEQUENTIAL OPERATIONAL
;    3369  2      ! MODE
;    3370  2      !
;    3371  2
;    3372  2          if (not BIT_TST (SWP_FLAGS, SWF_RDM)) and           ! NOT RANDOM MODE
;    3373  3             (not BIT_TST (SWP_FLAGS, SWF_SEQ))               ! NOT RANDOM SEQUENTIAL MODE
;    3374  2          then
;    3375  2
;    3376  2              if (.BST_CNT neq 0) and
;    3377  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3378  2                 (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3379  3                 (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3380  2              then
;    3381  3                  begin                                       ! ALREADY WITHIN DEVICE
;    3382  3                  BST_CNT = .BST_CNT - 1;
;    3383  3                  SET_UPAR (.BST_DEV);
;    3384  3                  MAD1 [DK_NUM] = .CDISK;
;    3385  3                  MAD2 [DK_NUM] = .CDISK;
;    3386  3                  return;
;    3387  3                  end
;    3388  2              else
```

```
;    3389  3                begin                                   ! GET NEW DEVICE
;    3390  3
;    3391  3                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3392  3
;    3393  3                    if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3394  3                       (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3395  4                       (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3396  3                    then
;    3397  3
;    3398  4                        if (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED)
;    3399  3                        then
;    3400  3                            RD_COUNT = .RD_COUNT + 1        ! NUMBER OF RD51/52s UNDER TEST
;    3401  3                        else
;    3402  3                            RX_COUNT = .RX_COUNT + 1;       ! NUMBER OF RX50s UNDER TEST
;    3403  3
;    3404  3                incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3405  4                    begin
;    3406  4
;    3407  4                    if (.BST_DEV eql 0) or
;    3408  5                       (.BST_DEV eql ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN))
;    3409  4                    then
;    3410  4                        BST_DEV = OF_UN
;    3411  4                    else
;    3412  4                        BST_DEV = .BST_DEV + UNIT_SIZE;
;    3413  4
;    3414  4                    if (.CST_ADDR [.BST_DEV + OF_DATA, D_PRES] eql PRESENT) and
;    3415  4                       (.CST_ADDR [.BST_DEV + OF_DATA, D_STAT] eql ONLINE) and
;    3416  5                       (not .CST_ADDR [.BST_DEV + OF_DATA, D_FATAL])
;    3417  4                    then
;    3418  5                        begin
;    3419  5
;    3420  5                        if .CST_ADDR [.BST_DEV + OF_DATA, D_TYPE] eql REMOVABLE
;    3421  5                        then
;    3422  5                            BST_CNT = .RX_MAX_SEQ_CNT / .RX_COUNT
;    3423  5                        else
;    3424  5                            BST_CNT = .RD_MAX_SEQ_CNT / .RD_COUNT;
;    3425  5
;    3426  5                        if .BST_CNT eql 0
;    3427  5                        then
;    3428  5                            BST_CNT = 1;
;    3429  5
;    3430  5                        SET_UPAR (.BST_DEV);
;    3431  5                        MAD1 [DK_NUM] = .CDISK;
;    3432  5                        MAD2 [DK_NUM] = .CDISK;
;    3433  5                        return;
;    3434  4                        end;
;    3435  4
;    3436  3                    end;
;    3437  3
;    3438  2                end;
;    3439  2
;    3440  2      !
;    3441  2      ! RANDOM SELECTION OF DRIVES
```

```
;    3442  2      !
;    3443  2      !
;    3444  2      ! DETERMINE IF RD51/52s ARE TO BE SELECTED
;    3445  2      !
;    3446  2
;    3447  2          if ((.RANDOM [RDM_LEN - 1] and %o'077777') mod 100) gequ .SWP_RAT
;    3448  2          then
;    3449  2              SELECT_RD = FALSE;
;    3450  2
;    3451  2      !
;    3452  2      ! IF RD51/52s SELECTED
;    3453  2      !
;    3454  2      ! COUNT NUMBER OF RD51/52s AVAILABLE
;    3455  2      !
;    3456  2
;    3457  2          if .SELECT_RD
;    3458  2          then
;    3459  3              begin
;    3460  3              MOD_COUNT = 0;                                  ! COUNT THE NUMBER OF RDs UNDER TEST
;    3461  3
;    3462  3              incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3463  3
;    3464  3                  if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3465  3                     (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3466  3                     (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql FIXED) and
;    3467  4                     (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3468  3                  then
;    3469  4                      begin
;    3470  4                      STORAGE [.MOD_COUNT] = .OFFSET;
;    3471  4                      MOD_COUNT = .MOD_COUNT + 1;
;    3472  3                      end;
;    3473  3
;    3474  3      !
;    3475  3      ! SELECT ON OF THE RD51/52s
;    3476  3      !
;    3477  3
;    3478  3              if .MOD_COUNT neq 0                             ! IF AT LEAST ONE RD51/52 PRESENT
;    3479  3              then
;    3480  4                  begin
;    3481  4                  TBL_COUNT = 0;
;    3482  4
;    3483  4                  do
;    3484  5                      begin
;    3485  5                      SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3486  5                      TBL_COUNT = .TBL_COUNT + 1;
;    3487  5                      end
;    3488  5                  until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3489  5                         (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3490  4                         (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3491  4                        (.TBL_COUNT eql RDM_LEN);
;    3492  4
;    3493  4                  MAD1 [DK_NUM] = .CDISK;
;    3494  4                  MAD2 [DK_NUM] = .CDISK;
```

```
;    3495  4                  return;
;    3496  3                  end;
;    3497  3
;    3498  2              end;
;    3499  2
;    3500  2      !
;    3501  2      ! IF NO RD51/52 SELECTED, SELECT AN RX50
;    3502  2      !
;    3503  2      ! COUNT THE NUMBER OF RX50s
;    3504  2      !
;    3505  2
;    3506  2          MOD_COUNT = 0;
;    3507  2
;    3508  2          incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3509  2
;    3510  2              if ( CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3511  2                 (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3512  2                 (.CST_ADDR [.OFFSET + OF_DATA, D_TYPE] eql REMOVABLE) and
;    3513  3                 (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3514  2              then
;    3515  3                  begin
;    3516  3                  STORAGE [.MOD_COUNT] = .OFFSET;
;    3517  3                  MOD_COUNT = .MOD_COUNT + 1;
;    3518  2                  end;
;    3519  2
;    3520  2      !
;    3521  2      ! AND CHOOSE ONE!
;    3522  2      !
;    3523  2
;    3524  2          if .MOD_COUNT neq 0
;    3525  2          then
;    3526  3              begin
;    3527  3              TBL_COUNT = 0;
;    3528  3
;    3529  3              do
;    3530  4                  begin
;    3531  4                  SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3532  4                  TBL_COUNT = .TBL_COUNT + 1;
;    3533  4                  end
;    3534  4              until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3535  4                     (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3536  3                     (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3537  3                    (.TBL_COUNT eql RDM_LEN);
;    3538  3
;    3539  3              MAD1 [DK_NUM] = .CDISK;
;    3540  3              MAD2 [DK_NUM] = .CDISK;
;    3541  3              return;
;    3542  2              end;
;    3543  2
;    3544  2      !
;    3545  2      ! IF NO UNIT SELECTED SO FAR BY ABOVE METHOD, SELECT ANY ONE AT RANDOM
;    3546  2      !
;    3547  2      ! COUNT ALL UNITS AVAILABLE
```

```
;    3548  2      !
;    3549  2
;    3550  2          MOD_COUNT = 0;
;    3551  2
;    3552  2          incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    3553  2
;    3554  2              if (.CST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and
;    3555  2                 (.CST_ADDR [.OFFSET + OF_DATA, D_STAT] eql ONLINE) and
;    3556  3                 (not .CST_ADDR [.OFFSET + OF_DATA, D_FATAL])
;    3557  2              then
;    3558  3                  begin
;    3559  3                  STORAGE [.MOD_COUNT] = .OFFSET;
;    3560  3                  MOD_COUNT = .MOD_COUNT + 1;
;    3561  2                  end;
;    3562  2
;    3563  2      !
;    3564  2      ! SELECT ANY ONE ONE UNIT AT RANDOM
;    3565  2      !
;    3566  2          if .MOD_COUNT neq 0
;    3567  2          then
;    3568  3              begin
;    3569  3              TBL_COUNT = 0;
;    3570  3
;    3571  3              do
;    3572  4                  begin
;    3573  4                  SET_UPAR (.STORAGE [(.RANDOM [.TBL_COUNT] and %o'077777') mod .MOD_COUNT]);
;    3574  4                  TBL_COUNT = .TBL_COUNT + 1;
;    3575  4                  end
;    3576  4              until ((.CST_ADDR [.CUOFF + OF_DATA, D_PRES] eql PRESENT) and
;    3577  4                     (.CST_ADDR [.CUOFF + OF_DATA, D_STAT] eql ONLINE) and
;    3578  3                     (not .CST_ADDR [.CUOFF + OF_DATA, D_FATAL])) or
;    3579  3                    (.TBL_COUNT eql RDM_LEN);
;    3580  3
;    3581  3              MAD1 [DK_NUM] = .CDISK;
;    3582  3              MAD2 [DK_NUM] = .CDISK;
;    3583  3              return
;    3584  3              end
;    3585  3
;    3586  3      !
;    3587  3      ! DECLARE END-OF-PASS IF NO UNIT ONLINE
;    3588  3      !
;    3589  3
;    3590  2          else
;    3591  2              EOP_FLAG = TRUE;
;    3592  2
;    3593  1          end;                                        ! ROUTINE QIO_UNIT


                                .SBTTL  QIO.UNIT MULTI-DRIVE TEST ROUTINES
000000  004137  000006  QIO.UNIT::
                        JSR     R1,$SAVE4                       ;                                   3356
000004  112704  000001          MOVB    #1,R4                   ; #,SELECT.RD                       3353
000010  005002                  CLR     R2                      ; RD.COUNT
```

```
000012  005003                          CLR     R3                              ; RX.COUNT
000014  032737  000002  000000G         BIT     #2,SWP.FLAGS                    ;                       3372
000022  001003                          BNE     1$
000024  032737  001000  000000G         BIT     #1000,SWP.FLAGS                 ;                       3373
000032  001402                  1$:     BEQ     2$
000034  000137  011504'                 JMP     15$
000040  005737  001146'          2$:     TST     BST.CNT                         ;                       3376
000044  001452                          BEQ     3$
000046  013700  001150'                 MOV     BST.DEV,R0                      ;                       3377
000052  006300                          ASL     R0
000054  063700  000000G                 ADD     CST.ADDR,R0
000060  032710  040000                  BIT     #40000,(R0)
000064  001442                          BEQ     3$
000066  013700  001150'                 MOV     BST.DEV,R0                      ;                       3378
000072  006300                          ASL     R0
000074  063700  000000G                 ADD     CST.ADDR,R0
000100  032710  020000                  BIT     #20000,(R0)
000104  001432                          BEQ     3$
000106  013700  001150'                 MOV     BST.DEV,R0                      ;                       3379
000112  006300                          ASL     R0
000114  063700  000000G                 ADD     CST.ADDR,R0
000120  032710  010000                  BIT     #10000,(R0)
000124  001022                          BNE     3$
000126  005337  001146'                 DEC     BST.CNT                         ;                       3382
000132  013746  001150'                 MOV     BST.DEV,-(SP)                   ;                       3383
000136  004737  000000G                 JSR     PC,SET.UPAR
000142  013700  000112'                 MOV     MAD1,R0                         ;                       3384
000146  013760  000000G 000016          MOV     CDISK,16(R0)
000154  013700  000114'                 MOV     MAD2,R0                         ;                       3385
000160  013760  000000G 000016          MOV     CDISK,16(R0)
000166  005726                          TST     (SP)+                           ;                       3386
000170  000207                          RTS     PC                              ;                       3381
000172  012701  000006          3$:     MOV     #6,R1                           ; *,OFFSET              3391
000176  010100                  4$:     MOV     R1,R0                           ; OFFSET,*              3393
000200  063700  000000G                 ADD     CST.ADDR,R0
000204  032710  040000                  BIT     #40000,(R0)
000210  001414                          BEQ     6$
000212  032710  020000                  BIT     #20000,(R0)                     ;                       3394
000216  001411                          BEQ     6$
000220  032710  010000                  BIT     #10000,(R0)                     ;                       3395
000224  001006                          BNE     6$
000226  132710  000020                  BITB    #20,(R0)                        ;                       3398
000232  001402                          BEQ     5$
000234  005202                          INC     R2                              ; RD.COUNT              3400
000236  000401                          BR      6$                              ;                       3398
000240  005203                  5$:     INC     R3                              ; RX.COUNT              3402
000242  062701  000024          6$:     ADD     #24,R1                          ; *,OFFSET              3391
000246  020127  000102                  CMP     R1,#102                         ; OFFSET,*
000252  003751                          BLE     4$
000254  012701  000003                  MOV     #3,R1                           ; *,OFFSET              3404
000260  013700  001150'          7$:     MOV     BST.DEV,R0                      ;                       3407
000264  001403                          BEQ     8$
000266  020027  000041                  CMP     R0,#41                          ;                       3408
```

```
000272  001004                          BNE     9$
000274  012737  000003  001150'    8$:   MOV     #3,BST.DEV                      ;                       3410
000302  000403                          BR      10$                             ;                       3407
000304  062737  000012  001150'    9$:   ADD     #12,BST.DEV                     ;                       3412
000312  013700  001150'            10$:  MOV     BST.DEV,R0                      ;                       3414
000316  006300                          ASL     R0
000320  063700  000000G                 ADD     CST.ADDR,R0
000324  032710  040000                  BIT     #40000,(R0)
000330  001450                          BEQ     14$
000332  032710  020000                  BIT     #20000,(R0)                     ;                       3415
000336  001445                          BEQ     14$
000340  032710  010000                  BIT     #10000,(R0)                     ;                       3416
000344  001042                          BNE     14$
000346  132710  000020                  BITB    #20,(R0)                        ;                       3420
000352  001004                          BNE     11$
000354  013746  001176'                 MOV     RX.MAX.SEQ.CNT,-(SP)            ;                       3422
000360  010346                          MOV     R3,-(SP)                        ; RX.COUNT,*
000362  000403                          BR      12$
000364  013746  001174'            11$:  MOV     RD.MAX.SEQ.CNT,-(SP)            ;                       3424
000370  010246                          MOV     R2,-(SP)                        ; RD.COUNT,*
000372  004737  000000G            12$:  JSR     PC,BL$DIV
000376  010037  001146'                 MOV     R0,BST.CNT
000402  001003                          BNE     13$                             ;                       3426
000404  012737  000001  001146'         MOV     #1,BST.CNT                      ;                       3428
000412  013716  001150'            13$:  MOV     BST.DEV,(SP)                    ;                       3430
000416  004737  000000G                 JSR     PC,SET.UPAR
000422  013700  000112'                 MOV     MAD1,R0                         ;                       3431
000426  013760  000000G  000016         MOV     CDISK,16(R0)
000434  013700  000114'                 MOV     MAD2,R0                         ;                       3432
000440  013760  000000G  000016         MOV     CDISK,16(R0)
000446  022626                          CMP     (SP)+,(SP)+                     ;                       3433
000450  000207                          RTS     PC                              ;                       3418
000452  062701  000012            14$:  ADD     #12,R1                          ; *,OFFSET              3404
000456  020127  000041                  CMP     R1,#41                          ; OFFSET,*
000462  003676                          BLE     7$
000464  013746  000036G            15$:  MOV     RANDOM+36,-(SP)                 ;                       3447
000470  042716  100000                  BIC     #100000,(SP)
000474  012746  000144                  MOV     #144,-(SP)
000500  004737  000000G                 JSR     PC,BL$MOD
000504  022626                          CMP     (SP)+,(SP)+
000506  020037  000000G                 CMP     R0,SWP.RAT
000512  103401                          BLO     16$
000514  105004                          CLRB    R4                              ; SELECT.RD             3449
000516  006004            16$:  ROR     R4                              ; SELECT.RD             3457
000520  103105                          BCC     21$
000522  105003                          CLRB    R3                              ; MOD.COUNT             3460
000524  012701  000003                  MOV     #3,R1                           ; *,OFFSET              3462
000530  010100            17$:  MOV     R1,R0                           ; OFFSET,*              3464
000532  006300                          ASL     R0
000534  063700  000000G                 ADD     CST.ADDR,R0
000540  032710  040000                  BIT     #40000,(R0)
000544  001417                          BEQ     18$
000546  032710  020000                  BIT     #20000,(R0)                     ;                       3465
```

```
000552  001414                          BEQ     18$
000554  132710  000020                  BITB    #20,(R0)                ;                               3466
000560  001411                          BEQ     18$
000562  032710  010000                  BIT     #10000,(R0)             ;                               3467
000566  001006                          BNE     18$
000570  005000                          CLR     R0                      ;                               3470
000572  150300                          BISB    R3,R0                   ; MOD.COUNT,*
000574  006300                          ASL     R0
000576  010160  000064'                  MOV     R1,STORAGE(R0)          ; OFFSET,*
000602  105203                          INCB    R3                      ; MOD.COUNT                     3471
000604  062701  000012          18$:    ADD     #12,R1                  ; *,OFFSET                      3462
000610  020127  000041                  CMP     R1,#41                  ; OFFSET,*
000614  003745                          BLE     17$
000616  105703                          TSTB    R3                      ; MOD.COUNT                     3478
000620  001445                          BEQ     21$
000622  105002                          CLRB    R2                      ; TBL.COUNT                     3481
000624  005000                  19$:    CLR     R0                      ;                               3485
000626  150200                          BISB    R2,R0                   ; TBL.COUNT,*
000630  006300                          ASL     R0
000632  016046  000000G                 MOV     RANDOM(R0),-(SP)
000636  042716  100000                  BIC     #100000,(SP)
000642  005046                          CLR     -(SP)
000644  110316                          MOVB    R3,(SP)                 ; MOD.COUNT,*
000646  004737  000000G                 JSR     PC,BL$MOD
000652  006300                          ASL     R0
000654  016016  000064'                  MOV     STORAGE(R0),(SP)
000660  004737  000000G                 JSR     PC,SET.UPAR
000664  105202                          INCB    R2                      ; TBL.COUNT                     3486
000666  022626                          CMP     (SP)+,(SP)+             ;                               3484
000670  013700  000000G                 MOV     CUOFF,R0                ;                               3488
000674  006300                          ASL     R0
000676  063700  000000G                 ADD     CST.ADDR,R0
000702  032710  040000                  BIT     #40000,(R0)
000706  001406                          BEQ     20$
000710  032710  020000                  BIT     #20000,(R0)             ;                               3489
000714  001403                          BEQ     20$
000716  032710  010000                  BIT     #10000,(R0)             ;                               3490
000722  001510                          BEQ     26$
000724  120227  000020          20$:    CMPB    R2,#20                  ; TBL.COUNT,*                   3491
000730  001335                          BNE     19$
000732  000504                          BR      26$                     ;                               3493
000734  105003                  21$:    CLRB    R3                      ; MOD.COUNT                     3506
000736  012701  000003                  MOV     #3,R1                   ; *,OFFSET                      3508
000742  010100                  22$:    MOV     R1,R0                   ; OFFSET,*                      3510
000744  006300                          ASL     R0
000746  063700  000000G                 ADD     CST.ADDR,R0
000752  032710  040000                  BIT     #40000,(R0)
000756  001417                          BEQ     23$
000760  032710  020000                  BIT     #20000,(R0)             ;                               3511
000764  001414                          BEQ     23$
000766  132710  000020                  BITB    #20,(R0)                ;                               3512
000772  001011                          BNE     23$
000774  032710  010000                  BIT     #10000,(R0)             ;                               3513
```

```
001000  001006                          BNE     23$
001002  005000                          CLR     R0                              ;                               3516
001004  150300                          BISB    R3,R0                           ; MOD.COUNT,*
001006  006300                          ASL     R0
001010  010160  000064'                 MOV     R1,STORAGE(R0)                  ; OFFSET,*
001014  105203                          INCB    R3                              ; MOD.COUNT                     3517
001016  062701  000012          23$:    ADD     #12,R1                          ; *,OFFSET                      3508
001022  020127  000041                  CMP     R1,#41                          ; OFFSET,'
001026  003745                          BLE     22$
001030  105703                          TSTB    R3                              ; MOD.COUNT                     3524
001032  001445                          BEQ     27$
001034  105002                          CLRB    R2                              ; TBL.COUNT                     3527
001036  005000          24$:    CLR     R0                                      ;                               3531
001040  150200                          BISB    R2,R0                           ; TBL.COUNT,*
001042  006300                          ASL     R0
001044  016046  000000G                 MOV     RANDOM(R0),-(SP)
001050  042716  100000                  BIC     #100000,(SP)
001054  005046                          CLR     -(SP)
001056  110316                          MOVB    R3,(SP)                         ; MOD.COUNT,*
001060  004737  000000G                 JSR     PC,BL$MOD
001064  006300                          ASL     R0
001066  016016  000064'                 MOV     STORAGE(R0),(SP)
001072  004737  000000G                 JSR     PC,SET.UPAR
001076  105202                          INCB    R2                              ; TBL.COUNT                     3532
001100  022626                          CMP     (SP)+,(SP)+                     ;                               3530
001102  013700  000000G                 MOV     CUOFF,R0                        ;                               3534
001106  006300                          ASL     R0
001110  063700  000000G                 ADD     CST.ADDR,R0
001114  032710  040000                  BIT     #40000,(R0)
001120  001406                          BEQ     25$
001122  032710  020000                  BIT     #20000,(R0)                     ;                               3535
001126  001403                          BEQ     25$
001130  032710  010000                  BIT     #10000,(R0)                     ;                               3536
001134  001505                          BEQ     32$
001136  120227  000020          25$:    CMPB    R2,#20                          ; TBL.COUNT,*                   3537
001142  001335                          BNE     24$
001144  000501          26$:    BR      32$                                     ;                               3539
001146  105003          27$:    CLRB    R3                              ; MOD.COUNT                             3550
001150  012701  000003                  MOV     #3,R1                           ; *,OFFSET                      3552
001154  010100          28$:    MOV     R1,R0                           ; OFFSET,*                              3554
001156  006300                          ASL     R0
001160  063700  000000G                 ADD     CST.ADDR,R0
001164  032710  040000                  BIT     #40000,(R0)
001170  001414                          BEQ     29$
001172  032710  020000                  BIT     #20000,(R0)                     ;                               3555
001176  001411                          BEQ     29$
001200  032710  010000                  BIT     #10000,(R0)                     ;                               3556
001204  001006                          BNE     29$
001206  005000                          CLR     R0                              ;                               3559
001210  150300                          BISB    R3,R0                           ; MOD.COUNT,*
001212  006300                          ASL     R0
001214  010160  000064'                 MOV     R1,STORAGE(R0)                  ; OFFSET,*
001220  105203                          INCB    R3                              ; MOD.COUNT                     3560
```

```
001222  062701  000012                29$:    ADD     #12,R1                  ; *,OFFSET                      3552
001226  020127  000041                        CMP     R1,#41                  ; OFFSET,*
001232  003750                              BLE     28$
001234  105703                              TSTB    R3                      ; MOD.COUNT                     3566
001236  001457                              BEQ     33$
001240  105002                              CLRB    R2                      ; TBL.COUNT                     3569
001242  005000                      30$:    CLR     R0                      ;                               3573
001244  150200                              BISB    R2,R0                   ; TBL.COUNT,*
001246  006300                              ASL     R0
001250  016046  000000G                     MOV     RANDOM(R0),-(SP)
001254  042716  100000                      BIC     #100000,(SP)
001260  005046                              CLR     (SP)
001262  110316                              MOVB    R3,(SP)                 ; MOD.COUNT,*
001264  004737  000000G                     JSR     PC,BL$MOD
001270  006300                              ASL     R0
001272  016016  000064'                     MOV     STORAGE(R0),(SP)
001276  004737  000000G                     JSR     PC,SET.UPAR
001302  105202                              INCB    R2                      ; TBL.COUNT                     3574
001304  022626                              CMP     (SP)+,(SP)+             ;                               3572
001306  013700  000000G                     MOV     CUOFF,R0                ;                               3576
001312  006300                              ASL     R0
001314  063700  000000G                     ADD     CST.ADDR,R0
001320  032710  040000                      BIT     #40000,(R0)
001324  001406                              BEQ     31$
001326  032710  020000                      BIT     #20000,(R0)             ;                               3577
001332  001403                              BEQ     31$
001334  032710  010000                      BIT     #10000,(R0)             ;                               3578
001340  001403                              BEQ     32$
001342  120227  000020                31$:    CMPB    R2,#20                  ; TBL.COUNT,*                   3579
001346  001335                              BNE     30$
001350  013700  000112'                32$:    MOV     MAD1,R0                 ;                               3581
001354  013760  000000G 000016              MOV     CDISK,16(R0)
001362  013700  000114'                      MOV     MAD2,R0                 ;                               3582
001366  013760  000000G 000016              MOV     CDISK,16(R0)
001374  000207                              RTS     PC                      ;                               3568
001376  112737  000001  000000G     33$:    MOVB    #1,EOP.FLAG             ;                               3591
001404  000207                              RTS     PC                      ;                               3336

; Routine Size:  387 words,    Routine Base:  $CODE$ + 11020
; Maximum stack depth per invocation:  8 words
```

```
;    3594  1     GLOBAL routine QIO_FUNC : novalue =
;    3595  1
;    3596  1     !+
;    3597  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;    3598  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;    3599  1     !       BY THE FOLLWING ALGORITHM:
;    3600  1     !
;    3601  1     !               IF THE CHOSEN UNIT IS PROTECTED
;    3602  1     !               THEN
;    3603  1     !                       FUNCTION = READ
;    3604  1     !               ELSE (UNPROTECTED)
;    3605  1     !                       FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;    3606  1     !                       NUMBER
;    3607  1     !
;    3608  1     !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
;    3609  1     !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;    3610  1     !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;    3611  1     !       PACKET. OTHERWISE, THE SECOND MSCP PACKET IS RETURNED TO THE POOL.
;    3612  1     !
;    3613  1     !
;    3614  1     !       PERIODIACLLY, THIS ROUTINE WILL CALL THE DUP ROUTINE BEFORE IT   ZZZ
;    3615  1     !       BEGINS ITS OWN TASK. IF THE OPERATOR HAS SELECTED, "ALSO RUN     ZZZ
;    3616  1     !       DUP EXERCISER," THEN DUP TESTING OF DBNS WILL BE INTERLEAVED     ZZZ
;    3617  1     !       WITH THE REGULAR MSCP TESTING OF THE LBNS.                       ZZZ
;    3618  1     !
;    3619  1     !       TO AVOID LONG, CUMULATIVE INIT TIMES, THE DUP CODE IS ONLY       ZZZ
;    3620  1     !       EXECUTED AFTER (25 TIMES 'DUPROUND') MSCP I/O's HAVE BEEN DONE.  ZZZ
;    3621  1     !       THE NUMBER OF DUP I/O's IS 'DUPROUND'. THIS GIVES US A 25 TO 1   ZZZ
;    3622  1     !       INTERLEAVE.                                                      ZZZ
;    3623  1     !
;    3624  1     !       THE DUP TESTING IS DONE BY EXECUTING CONTROLLER LOCAL PROGRAMS   ZZZ
;    3625  1     !       TO READ OR WRITE/READ DBNS. AFTER THE DUP TESTING, THE CON       ZZZ
;    3626  1     !       TROLLER IS REINITIALIZED, AND QIO_FUNC ROUTINE CONTINUES FROM    ZZZ
;    3627  1     !       WHERE IT LEFT OFF.                                               ZZZ
;    3628  1     !
;    3629  1     !
;    3630  1     !       IMPLICIT INPUTS:
;    3631  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    3632  1     !               CUOFF - CURRENT UNIT CST OFFSET
;    3633  1     !
;    3634  1     !       IMPLICIT OUTPUTS:
;    3635  1     !               THE OPCODE FIELD OF ONE OR BOTH MSCP PACKETS IS LOADED.
;    3636  1     !-
;    3637  1
;    3638  2         begin
;    3639  2
;    3640  2         local
;    3641  2             FUNC : word;                                          ! OPCODE (READ OR WRITE)
;    3642  2
;    3643  2     DUOFF = .CUOFF;                                  !SAVE IN CASE OTHER CMDS ZZZ
;    3644  2                                                      !LEFT IN QUEUE           ZZZ
;    3645  3     IF ((.CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] LEQ 0) AND !MSCP CNT<0       ZZZ
;    3646  3         (.CST_ADDR [.DUOFF, D_TYPE] NEQ RX_50) AND          !FIXED DISK      ZZZ
```

```
;    3647  3          (.CST_ADDR [.DUOFF + OF_DUPFLAGS, NODUPMEDIA] NEQ 1)) !MEDIA IN      ZZZ
;    3648  3                                                                              !ZZZ
;    3649  2      THEN                                                                    !ZZZ
;    3650  3          BEGIN                                                               !ZZZ
;    3651  3          PUT_PKT (.MX2);                          !RETURN 2ND ENVLOPE        !ZZZ
;    3652  3          MX2 = -1;                                !INDICATE FAILURE          ZZZ
;    3653  3          DUP ();                                  !DO DUP TEST               ZZZ
;    3654  3          CST_ADDR [.DUOFF + OF_COUNT, D_COUNT] =  !REINIT MSCP FUN-          ZZZ
;    3655  3                  (25 * .DUPROUND);                !CTION COUNTER             ZZZ
;    3656  3                                                                              !ZZZ
;    3657  3      !       THE FOLLOWING REINITS 2 ENVELOPES, SO THAT THE MSCP EXERCISER   ZZZ
;    3658  3      !       CAN PROCEED AS BEFORE THE DUP EXERCISER WAS CALLED.             ZZZ
;    3659  3      !                                                                       ZZZ
;    3660  3          DUP_FLAGS = .DUP_FLAGS OR SWP_DINT;      !SET DUP INIT FLAG         ZZZ
;    3661  3          INIT_TEST ();                            !REINIT CONTROLLER         ZZZ
;    3662  3          DUP_FLAGS = .DUP_FLAGS AND (NOT SWP_DINT);  !CLR DUP INIT DLAG      ZZZ
;    3663  3                                                                              !ZZZ
;    3664  3          MX2 = -1;                                !ASSUME NO 2ND ENVELOPE    ZZZ
;    3665  3          IF (MX1 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 1ST ENVELOPE      ZZZ
;    3666  4              OR (.EOP_FLAG)                       !IF FAILURE                ZZZ
;    3667  3          THEN    RETURN;                          !NO POINT TO GO ON         ZZZ
;    3668  3          IF (MX2 = GET_PKT (.CCTLR)) LSS 0        !TRY FOR 2ND ENVELOPE      ZZZ
;    3669  4              OR (.EOP_FLAG)                       !IF FAILURE                ZZZ
;    3670  4          THEN    BEGIN                                                       !ZZZ
;    3671  4              PUT_PKT (.MX1);                      !PUT 1ST BACK IN POOL      ZZZ
;    3672  4              MX1 = -1;                            !INDICATE FAILURE          ZZZ
;    3673  4              RETURN;                              !DONE                      ZZZ
;    3674  3              END;                                                            !ZZZ
;    3675  3                                                                              !ZZZ
;    3676  3              MAD1 = MSCP_PKT + (.MX1 * PKT_LEN * 2);  !CALC START ADDR      ZZZ
;    3677  3              MAD2 = MSCP_PKT + (.MX2 * PKT_LEN * 2);  !OF BOTH ENVELOPES    ZZZ
;    3678  3              GET_RANDOM ();                       !GET SET OF RANDOM NOS     ZZZ
;    3679  3              QIO_UNIT ();                         !PUT RAND UNIT NO IN       ZZZ
;    3680  2              END;                                 !ENVELOPES                 ZZZ
;    3681  2                                                                             !ZZZ
;    3682  2      !       MSCP CODE STARTS HERE                                          ZZZ
;    3683  2      !                                                                      ZZZ
;    3684  2
;    3685  2      CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] =      !                         ZZZ
;    3686  2          .CST_ADDR [.CUOFF + OF_COUNT, D_COUNT] -1;  !DECR MSCP FUNCTION CNTR ZZZ
;    3687  2
;    3688  2          MAD2 [OPCODE] = 0;                                                 ! ASSUME 2ND PACKET NOT NEEDED
;    3689  2
;    3690  2          if (.CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql UNPROTECTED) and      ! IF "FORCED_ERROR" SET IN LAST READ,
;    3691  2             (.CST_ADDR [.CUOFF + OF_DATA, D_TYPE] eql FIXED) and            !       REWRITE SAME BLOCK
;    3692  3             (.FORCED_ERROR)                                                !
;    3693  2          then
;    3694  2              FUNC = OP_WRT
;    3695  2          else
;    3696  2
;    3697  2              if .CST_ADDR [.CUOFF + OF_DATA, D_PROT] eql PROTECTED          ! IF UNIT IS PROTECTED
;    3698  2              then
;    3699  2                  FUNC = OP_RD                                               ! SET FUNCTION TO READ
```

```
;    3700  2          else
;    3701  2
;    3702  3              if (.RANDOM [1] and 1)                          ! USE 2ND RANDOM NUMBER TO SELECT
;    3703  3              then
;    3704  2                  FUNC = OP_RD                                !     READ
;    3705  2              else
;    3706  2                  FUNC = OP_WRT;                              !     WRITE
;    3707  2
;    3708  2          if (MAD1 [OPCODE] = .FUNC) eql OP_WRT               ! LOAD CHOSEN OPCODE, IF WRITE
;    3709  2          then
;    3710  3              begin
;    3711  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                  ! NON-SEQUENTIAL COMMAND
;    3712  3
;    3713  4              if BIT_TST (SWP_FLAGS, SWF_CWC)                 ! IF CONTROLLER DOES WRITE-COMPARES
;    3714  3              then
;    3715  3                  MAD1 [MODIFY] = MD_CMP                      ! ADD COMPARE MODIFIER
;    3716  3              else
;    3717  3
;    3718  4                  if BIT_TST (SWP_FLAGS, SWF_HWC)             ! IF HOST DOES WRITE-COMPARES
;    3719  3                  then
;    3720  4                      begin
;    3721  4                      MAD1 [MODIFY] = MD_EXP;                 ! SET WRITE AS AN EXPRESS REQUEST
;    3722  4                      MAD2 [OPCODE] = OP_RD;                  ! SET READ OPCODE INTO 2ND MSCP PACKET
;    3723  4                      MAD2 [MODIFY] = MD_EXP;                 ! SET READ AS AN EXPRESS REQUEST TOO
;    3724  4                      MAD2 [CMD_TYPE] = NON_SEQ_CMD;          ! NON-SEQUENTIAL COMMAND
;    3725  3                      end;
;    3726  3              end
;    3727  2          else
;    3728  3              begin
;    3729  3              MAD1 [CMD_TYPE] = NON_SEQ_CMD;                  ! NON-SEQUENTIAL COMMAND
;    3730  3
;    3731  4              if BIT_TST (SWP_FLAGS, SWF_CRC)                 ! IF READ COMPARES   FUNCTION IS READ
;    3732  3              then
;    3733  3                  MAD1 [MODIFY] = MD_CMP;                     ! ADD COMPARE MODIFIER
;    3734  3
;    3735  2              end;
;    3736  2
;    3737  2          if .MAD2 [OPCODE] eql 0                             ! IF NO OPCODE IN 2ND PACKET
;    3738  2          then
;    3739  3              begin
;    3740  3              PUT_PKT (.MX2);                                 ! RETURN 2ND PACKET TO POOL
;    3741  3              MX2 = -1;                                       ! MARK IT UNUSED
;    3742  2              end;
;    3743  2
;    3744  1          end;                                                ! ROUTINE QIO_FUNC


                                .SBTTL  QIO.FUNC MULTI-DRIVE TEST ROUTINES
000000  004137  000000G      QIO.FUNC::
                                JSR     R1,$SAVE4                       ;                       3594
000004  013737  000000G 001156'     MOV     CUOFF,DUOFF                     ;                       3644
000012  013702  000000G             MOV     CST,ADDR,R2                     ;                       3645
000016  013701  001156'             MOV     UUOFF,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           ;                       3646
000040  006300                          ASL     R0
000042  060200                          ADD     R2,R0
000044  132710  000020                  BITB    #20,(R0)
000050  001540                          BEQ     4$
000052  010100                          MOV     R1,R0           ;                       3647
000054  006300                          ASL     R0
000056  060200                          ADD     R2,R0
000060  005760  000020                  TST     20(R0)
000064  100532                          BMI     4$
000066  013746  000110'                  MOV     MX2,-(SP)       ;                       3651
000072  004737  0000006                 JSR     PC,PUT.PKT
000076  012737  177777  000110'         MOV     #-1,MX2         ;                       3652
000104  004737  000000V                 JSR     PC,DUP          ;                       3653
000110  013701  001156'                  MOV     DUOFF,R1        ;                       3654
000114  006301                          ASL     R1
000116  063701  0000006                 ADD     CST.ADDR,R1
000122  013716  0000006                 MOV     DUPROUND,(SP)   ;                       3655
000126  012746  000031                  MOV     #31,-(SP)
000132  004737  0000006                 JSR     PC,BL$MUL
000136  010061  000022                  MOV     R0,22(R1)
000142  052737  000002  0000006         BIS     #2,DUP.FLAGS    ;                       3660
000150  004737  001140'                  JSR     PC,INIT.TEST    ;                       3661
000154  042737  000002  0000006         BIC     #2,DUP.FLAGS    ;                       3662
000162  012737  177777  000110'         MOV     #-1,MX2         ;                       3664
000170  013716  0000006                 MOV     CCTLR,(SP)      ;                       3665
000174  004737  0000006                 JSR     PC,GET.PKT
000200  010037  000106'                  MOV     R0,MX1
000204  002426                          BLT     2$
000206  132737  000001  0000006         BITB    #1,EOP.FLAG     ;                       3666
000214  001022                          BNE     2$              ;                       3594
000216  013716  0000006                 MOV     CCTLR,(SP)      ;                       3668
000222  004737  0000006                 JSR     PC,GET.PKT
000226  010037  000110'                  MOV     R0,MX2
000232  002404                          BLT     1$
000234  132737  000001  0000006         BITB    #1,EOP.FLAG     ;                       3669
000242  001411                          BEQ     5$
000244  013716  000106'          1$:     MOV     MX1,(SP)        ;                       3671
000250  004737  0000006                 JSR     PC,PUT.PKT
000254  012737  177777  000106'         MOV     #-1,MX1         ;                       3672
000262  022626                  2$:     CMP     (SP)+,(SP)+     ;                       3673
000264  000207                          RTS     PC              ;                       3670
000266  013716  000106'          3$:     MOV     MX1,(SP)        ;                       3676
000272  012746  000106                  MOV     #106, (SP)
000276  004737  0000006                 JSR     PC,BL$MUL
000302  062700  0000006                 ADD     #MSCP.PKT,R0
000306  010037  000112'                  MOV     R0,MAD1
000312  013716  000110'                  MOV     MX2,(SP)        ;                       3677
```

```
000316  012746  000106                          MOV     #106,-(SP)
000322  004737  000000G                         JSR     PC,BL$MUL
000326  062700  000000G                         ADD     #MSCP.PKT,R0
000332  010037  000114'                         MOV     R0,MAD2
000336  004737  010710'                         JSR     PC,GET.RANDOM                   ;                       3673
000342  004737  011020'                         JSR     PC,QIO.UNIT                     ;                       3679
000346  062706  000010                          ADD     #10,SP                          ;                       3650
000352  013700  000000G             4$:         MOV     CUOFF,R0                        ;                       3685
000356  006300                                  ASL     R0
000360  063700  000000G                         ADD     CST.ADDR,R0
000364  005360  000022                          DEC     22(R0)                          ;                       3686
000370  013701  000114'                         MOV     MAD2,R1                         ;                       3688
000374  012704  000022                          MOV     #22,R4
000400  060104                                  ADD     R1,R4
000402  105014                                  CLRB    (R4)
000404  013700  000000G                         MOV     CUOFF,R0                        ;                       3690
000410  006300                                  ASL     R0
000412  063700  000000G                         ADD     CST.ADDR,R0
000416  005710                                  TST     (R0)
000420  100007                                  BPL     5$
000422  132710  000020                          BITB    #20,(R0)                        ;                       3691
000426  001404                                  BEQ     5$
000430  132737  000001  000000G                 BITB    #1,FORCED.ERROR                 ;                       3692
000436  001012                                  BNE     7$                              ;                       3694
000440  032710  100000             5$:         BIT     #100000,(R0)                    ;                       3697
000444  001404                                  BEQ     6$                              ;                       3699
000446  032737  000001  000002G                 BIT     #1,RANDOM+2                     ;                       3702
000454  001403                                  BEQ     7$
000456  012702  000041             6$:         MOV     #41,R2                          ; *,FUNC                3704
000462  000402                                  BR      8$                              ;                       3702
000464  012702  000042             7$:         MOV     #42,R2                          ; *,FUNC                3706
000470  013700  000112'             8$:         MOV     MAD1,R0                         ;                       3708
000474  013703  000000G                         MOV     SWP.FLAGS,R3                    ;                       3713
000500  110260  000022                          MOVB    R2,22(R0)                       ; FUNC,*                3708
000504  020227  000042                          CMP     R2,#42                          ; FUNC,*
000510  001025                                  BNE     9$
000512  112760  000002  000004                 MOVB    #2,4(R0)                        ;                       3711
000520  032703  000020                          BIT     #20,R3                          ;                       3713
000524  001025                                  BNE     10$                             ;                       3715
000526  032703  000040                          BIT     #40,R3                          ;                       3718
000532  001425                                  BEQ     11$
000534  012760  100000  000024                 MOV     #-100000,24(R0)                 ;                       3721
000542  112714  000041                          MOVB    #41,(R4)                        ;                       3723
000546  012761  100000  000024                 MOV     #-100000,24(R1)                 ;                       3725
000554  112761  000002  000004                 MOVB    #2,4(R1)                        ;                       3724
000562  000411                                  BR      11$                             ;                       3708
000564  112760  000002  000004     9$:         MOVB    #2,4(R0)                        ;                       3729
000572  032703  000004                          BIT     #4,R3                           ;                       3731
000576  001403                                  BEQ     11$
000600  012760  040000  000024     10$:        MOV     #40000,24(R0)                   ;                       3733
000606  105714                     11$:        TSTB    (R4)                            ;                       3737
000610  001010                                  BNE     12$
000612  013746  000110'                         MOV     MX2,-(SP)                       ;                       3740
```

F10


```
000616  004737  000000G                        JSR     PC,PUT.PKT
000622  012737  177777  000110'                MOV     #-1,MX2                 ;                       3741
000630  005726                                 TST     (SP)+                   ;                       3739
000632  000207                          12$:   RTS     PC                      ;                       3594

; Routine Size:  206 words,     Routine Base:  $CODE$ + 12426
; Maximum stack depth per invocation:  10 words
```

```
;  3745  1
;  3746  1      GLOBAL ROUTINE DUP : NOVALUE =                                          !ZZZ
;  3747  1      !+                                                                     !ZZZ
;  3748  1      !       THIS ROUTINE IS CALLED BY QIO_FUNC AFTER 25 * 'DUPROUND' RD/WTS. ZZZ
;  3749  1      !       THIS EXERCISER WAS PLACED IN THE MIDDLE OF THE MSCP EXERCISER,   ZZZ
;  3750  1      !       SO COMMON INIT AND OTHER ROUTINES COULD BE USED.                 ZZZ
;  3751  1      !                                                                        ZZZ
;  3752  1      !       THE DUP EXERCISER WILL PERFORM EITHER READ-ONLY, OR WRITE-READ-  ZZZ
;  3753  1      !       COMPARE OPERATIONS ON THE DIAGNOSTIC BLOCKS (DBNS). IT WILL      ZZZ
;  3754  1      !       RECORD THE STATISTICS IN THE TALLY TABLES.                       ZZZ
;  3755  1      !                                                                        ZZZ
;  3756  1      !       THE PROGRAM USES CONTROLLER LOCAL PROGRAMS TO WRITE AND READ     ZZZ
;  3757  1      !       DBNS. WHEN WRITING TO THE DBNS, A ONE WORD PATTERN WILL BE       ZZZ
;  3758  1      !       SELECTED, AND REPLICATED THROUGH A 256 WORD BLOCK FOR DATA.      ZZZ
;  3759  1      !       THE ROUTINE WILL WRITE 'DUPROUND' NUMBER OF SEQUENTIAL DBN       ZZZ
;  3760  1      !       BLOCKS. IF THE CONTROLLER LOCAL PROGRAMS EXIST, AND THE OPERATOR ZZZ
;  3761  1      !       SELECTS 'WRITE TO DIAGNOSTIC AREA', WRITE-READ-COMPARES WILL BE  ZZZ
;  3762  1      !       PERFORMED ON THE DBNS. OTHERWISE, READS WITH NO COMPARES WILL BE ZZZ
;  3763  1      !       DONE. BAD BLOCKS FOUND IN THE COMPARISON TESTS WILL NOT BE LIST- ZZZ
;  3764  1      !       ED IN THE RCT TABLES.                                            ZZZ
;  3765  1      !                                                                        ZZZ
;  3766  1      !       AFTER 'DUPROUND' NUMBER OF DBNS HAVE BEEN TESTED, THE ENVELOPES  ZZZ
;  3767  1      !       WILL BE REINITIATED, SO THAT THE MSCP EXERCISER CAN CONTINUE     ZZZ
;  3768  1      !       AS BEFORE.                                                       ZZZ
;  3769  1      !                                                                        ZZZ
;  3770  1      !       IMPLICIT INPUTS:                                                 ZZZ
;  3771  1      !                       CCTLR - CURRENT CONTROLLER NUMBER                ZZZ
;  3772  1      !                       CST_ADDRS - CONTAINS THE CURRENT CONTROLLER      ZZZ
;  3773  1      !                                   STATUS TABLE                         ZZZ
;  3774  1      !                       CUOFF - CURRENT OFFSET IN CST TABLE FOR          ZZZ
;  3775  1      !                               PARTICULAR DRIVE                         ZZZ
;  3776  1      !                                                                        ZZZ
;  3777  1      !       IMPLICIT OUTPUTS:                                                ZZZ
;  3778  1      !                       S_PATTERN - PATTERN BEING WRITTEN TO DBNS        ZZZ
;  3779  1      !                                                                        ZZZ
;  3780  1      !                                                                       !ZZZ
;  3781  1      !-                                                                      !ZZZ
;  3782  1
```

```
;    3783  1
;    3784  1
;    3785  2     BEGIN
;    3786  2     OWN
;    3787  2             TEMP : WORD;
;    3788  2
;    3789  2     !PRINTX (DBM110);
;    3790  2     !PRINTX (DER10);
;    3791  2
;    3792  2     until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    3793  2           (.EOP_FLAG eql true) do                  ! Make sure all MSCP commands are completed
;    3794  3         begin
;    3795  3         BREAK;                                     ! BREAK FOR ACT
;    3796  3         PROC_RETPKT();                             ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    3797  3         RP_INDX = .RP_INDX + 1;                    ! INCREMENT RP_INDX
;    3798  3         if .RP_INDX geq RP_CNT then (RP_INDX = 0;); ! MAKE SURE THE COUNTER DOES NOT GET TO BIG
;    3799  3         RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;    3800  2         end;
;    3801  2
;    3802  2
;    3803  2     S_PATTERN = .RANDOM [1];                                  !OTHER UNIT VARIABLES
;    3804  2
;    3805  2     IF (.CST_ADDR [.DUOFF + OF_DBN, D_DBN] + .dupround) GEQ 144     ! TEST TO SEE IF NEXT DBN'S TO LARGE
;    3806  2     THEN (CST_ADDR [.DUOFF + OF_DBN, D_DBN] = 0);             ! CIRCLE ARROUND IF DBN TO LARGE
;    3807  2
;    3808  2     DUPIDLE ();                                               ! DO A GET DUST STATUS TO FIND IF LOCAL DUP MEDIA
;    3809  2     IF .CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] EQL 1 THEN RETURN; ! IF DUP LOCAL MEDIA NOT THERE THEN RETURN
;    3810  2
;    3811  2     TEMP = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];
;    3812  2     INCR DBNCNT FROM (.TEMP + 1) TO (.TEMP + .dupround) DO                ! INCREMENT FROM RELATIVE DBN TO DBN + dupro
und
;    3813  3             BEGIN
;    3814  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPWRITE]                          ! IF WRITE FLAG SET IN CST TABLE THE
N WRITE DBN'S
;    3815  3             THEN
;    3816  4                     BEGIN
;    3817  4                     DUPIDLE ();                                       ! MAKE SURE THE CONTROLLER IS IN AN IDLE STA
TE
;    3818  4                     DUPWRTDBN ();                                     ! CALL ROUTINE TO HANDLE WRITTING ROUTINES
;    3819  3                     END;
;    3820  3
;    3821  3             DUPIDLE ();                                               ! MAKE SURE CONTROLLER IN IDLE STATE
;    3822  3             DUPREDDBN ();                                             ! CALL ROUTINE TO HANDLE READING DBN'S
;    3823  3
;    3824  3             CST_ADDR [.DUOFF + OF_DBN, D_DBN] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN] + 1; ! INCREMENT RELATIVE DBN COUNTER
;    3825  3
;    3826  3             IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1            ! ERROR IN DUP REINITIALIZE
;    3827  3             THEN RETURN;                                              ! AND RETURN
;    3828  2             END;
;    3829  1     END;


001204                          .PSECT  $GGG$,  RO
001204                  TEMP:   .BLKW   1
```

```
                                        .SBTTL  DUP MULTI-DRIVE TEST ROUTINES
013262                                  .PSECT  $CODE$,  RO

000000  004137  000000G           DUP::  JSR     R1,$SAVE3                       ;                       3746
000004  013700  000000G           1$:    MOV     RP.ADDR,R0                      ;                       3792
000010  023760  000000G 000004           CMP     CRN.LOW,4(R0)
000016  001433                           BEQ     3$
000020  123727  000000G 000001           CMPB    EOP.FLAG,#1                     ;                       3793
000026  001427                           BEQ     3$
000030  104422                           TRAP    22                              ;                       3794
000032  004737  000000V                  JSR     PC,PROC.RETPKT                  ;                       3796
000036  005237  000000G                  INC     RP.INDX                         ;                       3797
000042  023727  000000G 000010           CMP     RP.INDX,#10                     ;                       3798
000050  002402                           BLT     2$
000052  005037  000000G                  CLR     RP.INDX
000056  013746  000000G           2$:    MOV     RP.INDX,-(SP)                   ;                       3799
000062  012746  000054                   MOV     #54,-(SP)
000066  004737  000000G                  JSR     PC,BL$MUL
000072  062700  000000G                  ADD     #RETPKT,R0
000076  010037  000000G                  MOV     R0,RP.ADDR
000102  022626                           CMP     (SP)+,(SP)+                     ;                       3794
000104  000737                           BR      1$                              ;                       3792
000106  013737  000002G 000000G  3$:     MOV     RANDOM+2,S.PATTERN              ;                       3803
000114  013700  001156'                  MOV     DUOFF,R0                        ;                       3805
000120  006300                           ASL     R0
000122  063700  000000G                  ADD     CST.ADDR,R0
000126  005001                           CLR     R1
000130  156001  000020                   BISB    20(R0),R1
000134  063701  000000G                  ADD     DUPROUND,R1
000140  020127  000220                   CMP     R1,#220
000144  002402                           BLT     4$
000146  105060  000020                   CLRB    20(R0)                          ;                       3806
000152  004737  000000V           4$:    JSR     PC,DUPIDLE                      ;                       3808
000156  013700  001156'                  MOV     DUOFF,R0                        ;                       3809
000162  006300                           ASL     R0
000164  063700  000000G                  ADD     CST.ADDR,R0
000170  005760  000020                   TST     20(R0)
000174  100456                           BMI     8$
000176  116037  000020  001204'         MOVB    20(R0),TEMP                     ;                       3811
000204  105037  001205'                  CLRB    TEMP+1
000210  013703  001204'                  MOV     TEMP,R3                         ;                       3812
000214  063703  000000G                  ADD     DUPROUND,R3
000220  013700  001156'                  MOV     DUOFF,R0                        ;                       3814
000224  006300                           ASL     R0
000226  063700  000000G                  ADD     CST.ADDR,R0
000232  010001                           MOV     R0,R1
000234  062701  000020                   ADD     #20,R1
000240  013702  001204'                  MOV     TEMP,R2                         ; *,DBNCNT              3813
000244  000427                           BR      7$
000246  032711  010000           5$:     BIT     #10000,(R1)                     ;                       3814
000252  001404                           BEQ     6$
000254  004737  000000V                  JSR     PC,DUPIDLE                      ;                       3817
```

```
000260  004737  000000V                        JSR     PC,DUPWRTDBN            ;                       3818
000264  004737  000000V                6$:     JSR     PC,DUPIDLE              ;                       3821
000270  004737  000000V                        JSR     PC,DUPREDDBN            ;                       3822
000274  013700  001156'                        MOV     DUOFF,R0                ;                       3824
000300  006300                                ASL     R0
000302  063700  000000G                        ADD     CST.ADDR,R0
000306  010001                                MOV     R0,R1
000310  062701  000020                        ADD     #20,R1
000314  105211                                INCB    (R1)
000316  032711  040000                        BIT     #40000,(R1)             ;                       3826
000322  001003                                BNE     8$                      ;                       3827
000324  005202                7$:             INC     R2                      ; DBNCNT                3812
000326  020203                                CMP     R2,R3                   ; DBNCNT,*
000330  003746                                BLE     5$
000332  000207                8$:             RTS     PC                      ;                       3746

; Routine Size:  110 words,    Routine Base:  $CODE$ + 13262
; Maximum stack depth per invocation:  7 words


;   3830  1
```

```
;    3831  1      GLOBAL ROUTINE DUPWRTDBN : NOVALUE =
;    3832  1
;    3833  1      !+
;    3834  1      ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;    3835  1      ! "WRTDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;    3836  1      ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM WRITES TO A DIAGONOSTIC BLOCK (DBN)
;    3837  1      ! THE WORD IN "S_PATTERN" IS WRITTEN TO THE 256 WORDS IN THE DBN.  IF AN ERROR OCCURS
;    3838  1      ! WHILE RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;    3839  1      ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;    3840  1      !
;    3841  1      !       IMPLICIT INPUTS:
;    3842  1      !               CST_ADDRS  - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;    3843  1      !               DUOFF      - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;    3844  1      !               S_PATTERN  - CONTAINS PATTERN WORD!-
;    3845  2      BEGIN
;    3846  2      !PRINTX (DER11);
;    3847  2      T_ADDR [T_DBN_WT] = .T_ADDR [T_DBN_WT] + 1;     ! INCREMENT # OF WRITES GIVEN
;    3848  2
;    3849  2      MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;               ! PACKET SIZE                                   EXECUTE LOCAL PROGRAM WRT DB
N
;    3850  2      MSCP_PKT [.MX1, OPCODE] = OP_ELP;               ! OPCODE = EXECUTE LOCAL PROGRAM
;    3851  2      MSCP_PKT [.MX1, L1] = %ascii'W';                ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;    3852  2      MSCP_PKT [.MX1, L2] = %ascii'R';
;    3853  2      MSCP_PKT [.MX1, L3] = %ascii'T';
;    3854  2      MSCP_PKT [.MX1, L4] = %ascii'D';
;    3855  2      MSCP_PKT [.MX1, L5] = %ascii'B';
;    3856  2      MSCP_PKT [.MX1, L6] = %ascii'N';
;    3857  2      MSCP_PKT [.MX1, MODIFY] = 1;                    ! STANDALONE MODIFIER
;    3858  2      MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;              ! CALL IT IMMEDIATE
;    3859  2      DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    3860  2
;    3861  2      IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  !status error
;    3862  2      THEN RETURN;                                    ! AND RETURN
;    3863  2
;    3864  3      DO (MX1 = GET_PKT (.CCTLR))
;    3865  2      UNTIL (.MX1 GEQ 0);                     ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    3866  2
;    3867  2              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;               ! PACKET SIZE                                   RECIEVE DATA
;    3868  2              MSCP_PKT [.MX1, OPCODE] = OP_RCD;               ! OPCODE = RECEIVE DATA
;    3869  2              MSCP_PKT [.MX1, BC_LO] = 80;                    ! BYTE COUNT TO BE TRANSFERED EQUALS 2 ***see pg 26 of DUP s
pec
;    3870  2              MSCP_PKT [.MX1, BUF_0] = DUPPKT;                ! LOAD DESCRIBTOR BUFFER
;    3871  2              MSCP_PKT [.MX1, MODIFY] = 0;                    !
;    3872  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;              ! CALL IT sequential
;    3873  2              DUPCOMMAND ();                                  ! SENDS AND RECEIVES THE COMMAND
;    3874  2
;    3875  2              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR     !status error
;    3876  2                 (.DUPPKT [DUPTYPE] NEQU 1) OR                        !dup type error
;    3877  3                 (.DUPPKT [DUPMESG] NEQU 6)
;    3878  2              THEN
;    3879  3                      (HARD_ERROR ();
;    3880  3                       CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    3881  2                       RETURN;);                                      ! NO POINT IN CONTINUING
;    3882  2
;    3883  3              DO (MX1 = GET_PKT (.CCTLR))
```

```
;    3884  2              UNTIL (.MX1 GEQ 0);                         ! TRY TO GET AN ENVELOPE, IF FAILURE LOOP PRG ERROR
;    3885  2
;    3886  2              MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;           ! PACKET SIZE                              SEND DATA
;    3887  2              MSCP_PKT [.MX1, OPCODE] = OP_SDD;           ! OPCODE = SEND DATA
;    3888  2              MSCP_PKT [.MX1, BC_LO] = 6;                 ! BYTE COUNT TO BE TRANSFERED EQUALS 6
;    3889  2              MSCP_PKT [.MX1, BUF_0] = DUPPKT;            ! LOAD DESCRIBTOR BUFFER
;    3890  2              DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; !LOAD UNIT NUMBER (RDRX)
;    3891  2              DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;    3892  2              DUPPKT [DUPBF2] = .S_PATTERN;               ! LOAD PATTERN
;    3893  2              MSCP_PKT [.MX1, MODIFY] = 0;                !
;    3894  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;          ! CALL IT sequential
;    3895  2              DUPCOMMAND ();                              ! SENDS AND RECEIVES THE COMMAND
;    3896  2
;    3897  2              IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  ! status error
;    3898  2              THEN RETURN;
;    3899  2
;    3900  3              DO (MX1 = GET_PKT (.CCTLR))
;    3901  2              UNTIL (.MX1 GEQ 0);                 ! TRY TO GET AN ENVELOPE, IF FAILURE LOOP PRG ERROR
;    3902  2
;    3903  2              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;           ! PACKET SIZE                              RECEIVE DATA
;    3904  2              MSCP_PKT [.MX1, OPCODE] = OP_RCD;           ! OPCODE = RECEIVE DATA
;    3905  2              MSCP_PKT [.MX1, BC_LO] = 4;                 ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;    3906  2              MSCP_PKT [.MX1, BUF_0] = DUPPKT;            ! LOAD DESCRIBTOR BUFFER
;    3907  2              MSCP_PKT [.MX1, MODIFY] = 0;                !
;    3908  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;          ! CALL IT sequential
;    3909  2              DUPCOMMAND ();                              ! SENDS AND RECEIVES THE COMMAND
;    3910  2
;    3911  2              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) or     !status error
;    3912  2                 (.DUPPKT [DUPTYPE] NEQU 3) or                        !dup type error
;    3913  2                 (.DUPPKT [DUPMESG] NEQU 3) or
;    3914  3                 (.DUPPKT [DUPBF1] NEQU 0)                            !non successful write code
;    3915  2              THEN
;    3916  3                      (HARD_ERROR ();
;    3917  3                       CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    3918  2                       RETURN;);                                      ! NO POINT IN CONTINUING
;    3919  2
;    3920  3      DO (MX1 = GET_PKT (.CCTLR))
;    3921  2      UNTIL (.MX1 GEQ 0);                             ! TRY TO GET AN ENVELOPE,
;    3922  2
;    3923  2      T_ADDR [T_BLK_WT] = .T_ADDR [T_BLK_WT] + 1;     !INCREMENT COUNTER IF A SUCCESS
;    3924  2
;    3925  1      END;


                                        .SBTTL  DUPWRTDBN MULTI-DRIVE TEST ROUTINES
000000  010146                  DUPWRTDBN::
                                        MOV     R1, (SP)                    ;                                  3831
000002  013700  000000G                 MOV     T.ADDR,R0                   ;                                  3847
000006  005260  000060                  INC     60(R0)
000012  013746  000106'                 MOV     MX1, -(SP)                  ;                                  3849
000016  012746  000106                  MOV     #106,-(SP)
000022  004737  000000G                 JSR     PC,BL$MUL
000026  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
```

```
000034  112760  000003  000022G                 MOVB    #3,MSCP.PKT+22(R0)          ;                3850
000042  112760  000127  000026G                 MOVB    #127,MSCP.PKT+26(R0)        ;                3851
000050  112760  000122  000027G                 MOVB    #122,MSCP.PKT+27(R0)        ;                3852
000056  112760  000124  000030G                 MOVB    #124,MSCP.PKT+30(R0)        ;                3853
000064  112760  000104  000031G                 MOVB    #104,MSCP.PKT+31(R0)        ;                3854
000072  112760  000102  000032G                 MOVB    #102,MSCP.PKT+32(R0)        ;                3855
000100  112760  000116  000033G                 MOVB    #116,MSCP.PKT+33(R0)        ;                3856
000106  012760  000001  000024G                 MOV     #1,MSCP.PKT+24(R0)          ;                3857
000114  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)        ;                3858
000122  004737  000000V                         JSR     PC,DUPCOMMAND               ;                3859
000126  013700  001156'                         MOV     DUOFF,R0                    ;                3861
000132  006300                                  ASL     R0
000134  063700  000000G                         ADD     CST.ADDR,R0
000140  032760  040000  000020                 BIT     #40000,20(R0)
000146  001402                                  BEQ     1$
000150  022626                                  CMP     (SP)+,(SP)+                 ;                3831
000152  000505                                  BR      3$                          ;                3862
000154  013716  000000G                 1$:     MOV     CCTLR,(SP)                  ;                3864
000160  004737  000000G                         JSR     PC,GET.PKT
000164  010037  000106'                         MOV     R0,MX1
000170  002771                                  BLT     1$                          ;                3865
000172  010016                                  MOV     R0,(SP)                     ;                3867
000174  012746  000106                         MOV     #106,-(SP)
000200  004737  000000G                         JSR     PC,BL$MUL
000204  012760  000034  000006G                 MOV     #34,MSCP.PKT+6(R0)
000212  112760  000005  000022G                 MOVB    #5,MSCP.PKT+22(R0)          ;                3868
000220  012760  000120  000026G                 MOV     #120,MSCP.PKT+26(R0)        ;                3869
000226  012760  000000G  000032G                 MOV     #DUPPKT,MSCP.PKT+32(R0)     ;                3870
000234  005060  000024G                         CLR     MSCP.PKT+24(R0)             ;                3871
000240  142760  000360  000010G                 BICB    #360,MSCP.PKT+10(R0)        ;                3872
000246  152760  000020  000010G                 BISB    #20,MSCP.PKT+10(R0)
000254  004737  000000V                         JSR     PC,DUPCOMMAND               ;                3873
000260  013700  001156'                         MOV     DUOFF,R0                    ;                3875
000264  006300                                  ASL     R0
000266  063700  000000G                         ADD     CST.ADDR,R0
000272  032760  040000  000020                 BIT     #40000,20(R0)
000300  001016                                  BNE     2$
000302  013700  000000G                         MOV     DUPPKT,R0                   ;                3876
000306  042700  007777                         BIC     #7777,R0
000312  020027  010000                         CMP     R0,#10000
000316  001007                                  BNE     2$
000320  013700  000000G                         MOV     DUPPKT,R0                   ;                3877
000324  042700  170000                         BIC     #170000,R0
000330  020027  000006                         CMP     R0,#6
000334  001415                                  BEQ     4$
000336  004737  000000V                 2$:     JSR     PC,HARD.ERROR               ;                3879
000342  013700  001156'                         MOV     DUOFF,R0                    ;                3880
000346  006300                                  ASL     R0
000350  063700  000000G                         ADD     CST.ADDR,R0
000354  052760  040000  000020                 BIS     #40000,20(R0)
000362  062706  000006                         ADD     #6,SP                       ;                3881
000366  000504                          3$:     BR      5$                          ;                3879
000370  013716  000000G                 4$:     MOV     CCTLR,(SP)                  ;                3883
```

```
000374  004737  000000G                        JSR     PC,GET.PKT
000400  010037  000106'                        MOV     R0,MX1
000404  002771                                 BLT     4$                              ;       3884
000406  010016                                 MOV     R0,(SP)                         ;       3886
000410  012746  000106                         MOV     #106,-(SP)
000414  004737  000000G                        JSR     PC,BL$MUL
000420  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000426  112760  000004  000022G                MOVB    #4,MSCP.PKT+22(R0)              ;       3887
000434  012760  000006  000026G                MOV     #6,MSCP.PKT+26(R0)              ;       3888
000442  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)         ;       3889
000450  013701  001156'                        MOV     DUOFF,R1                        ;       3890
000454  006301                                 ASL     R1
000456  063701  000000G                        ADD     CST.ADDR,R1
000462  111137  000000G                        MOVB    (R1),DUPPKT
000466  042737  177760  000000G                BIC     #177760,DUPPKT
000474  013701  001156'                        MOV     DUOFF,R1                        ;       3891
000500  006301                                 ASL     R1
000502  063701  000000G                        ADD     CST.ADDR,R1
000506  116137  000020  000002G                MOVB    20(R1),DUPPKT+2
000514  105037  000003G                        CLRB    DUPPKT+3
000520  013737  000000G 000004G                MOV     S.PATTERN,DUPPKT+4              ;       3892
000526  005060  000024G                        CLR     MSCP.PKT+24(R0)                 ;       3893
000532  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)            ;       3894
000540  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000546  004737  000000V                        JSR     PC,DUPCOMMAND                   ;       3895
000552  013700  001156'                        MOV     DUOFF,R0                        ;       3897
000556  006300                                 ASL     R0
000560  063700  000000G                        ADD     CST.ADDR,R0
000564  032760  040000  000020                BIT     #40000,20(R0)
000572  001403                                 BEQ     6$
000574  062706  000010                         ADD     #10,SP                          ;       3831
000600  000524                         5$:     BR      10$                             ;       3898
000602  013716  000000G                6$:     MOV     CCTLR,(SP)                      ;       3900
000606  004737  000000G                        JSR     PC,GET.PKT
000612  010037  000106'                        MOV     R0,MX1
000616  002771                                 BLT     6$                              ;       3901
000620  010016                                 MOV     R0,(SP)                         ;       3903
000622  012746  000106                         MOV     #106,-(SP)
000626  004737  000000G                        JSR     PC,BL$MUL
000632  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000640  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)              ;       3904
000646  012760  000004  000026G                MOV     #4,MSCP.PKT+26(R0)              ;       3905
000654  012760  000000G 000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)         ;       3906
000662  005060  000024G                        CLR     MSCP.PKT+24(R0)                 ;       3907
000666  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)            ;       3908
000674  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000702  004737  000000V                        JSR     PC,DUPCOMMAND                   ;       3909
000706  013700  001156'                        MOV     DUOFF,R0                        ;       3911
000712  006300                                 ASL     R0
000714  063700  000000G                        ADD     CST.ADDR,R0
000720  032760  040000  000020                BIT     #40000,20(R0)
000726  001021                                 BNE     7$
000730  013700  000000G                        MOV     DUPPKT,R0                       ;       3912
```

```
000734  042700  007777                          BIC     #7777,R0
000740  020027  030000                          CMP     R0,#30000
000744  001012                                  BNE     7$
000746  013700  000000G                         MOV     DUPPKT,R0               ;                       3913
000752  042700  170000                          BIC     #170000,R0
000756  020027  000003                          CMP     R0,#3
000762  001003                                  BNE     7$
000764  005737  000002G                         TST     DUPPKT+2                ;                       3914
000770  001413                                  BEQ     8$
000772  004737  000000V                 7$:     JSR     PC,HARD.ERROR           ;                       3916
000776  013700  001156'                         MOV     DUOFF,R0                ;                       3917
001002  006300                                  ASL     R0
001004  063700  000000G                         ADD     CST.ADDR,R0
001010  052760  040000  000020                  BIS     #40000,20(R0)
001016  000413                                  BR      9$                      ;                       3918
001020  013716  000000G                 8$:     MOV     CCTLR,(SP)              ;                       3920
001024  004737  000000G                         JSR     PC,GET.PKT
001030  010037  000106'                         MOV     R0,MX1
001034  002771                                  BLT     8$                      ;                       3921
001036  013700  000000G                         MOV     T.ADDR,R0               ;                       3923
001042  005260  000056                          INC     56(R0)
001046  062706  000012                 9$:     ADD     #12,SP                  ;                       3845
001052  012601                         10$:    MOV     (SP)+,R1                ;                       3831
001054  000207                                  RTS     PC

; Routine Size:  279 words,    Routine Base:  $CODE$ + 13616
; Maximum stack depth per invocation:  7 words
```

```
;    3926  1      GLOBAL ROUTINE DUPREDDBN : NOVALUE =
;    3927  1
;    3928  1      !+
;    3929  1      ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO USE THE CONTROLLER LOCAL PROGRAM
;    3930  1      ! "REDDBN".  TO USE THE PROGRAM THE OPTIONAL DUP SUB-PROTOCAL IS USED TO
;    3931  1      ! COMMUNICATE WITH THE CONTROLLER.  THE PROGRAM READS A DIAGONOSTIC BLOCK (DBN)
;    3932  1      ! AND PLACES IT IN THE DUP BUFFER CALLED "DUPPKT".  IF AN ERROR OCCURS WHILE
;    3933  1      ! RUNNING THE CONTROLLER LOCAL PROGRAM THE ERROR IS USSUALLY REPORTED IN THE
;    3934  1      ! DUP BUFFER. (EX. ILLEGAL UNIT NUMBER, ILLEGAL BLK #, DEVICE ERROR, ZERO LENGHT MSG)
;    3935  1      !
;    3936  1      !
;    3937  1      !       IMPLICIT INPUTS:
;    3938  1      !               CST_ADDR    - CONTAINS THE CURRENT CONTROLLER STATUS TABLE
;    3939  1      !               DUOFF       - CURRENT OFFSET IN CST TABLE FOR PARTICULAR DRIVE
;    3940  1      !
;    3941  2      BEGIN
;    3942  2      !PRINTX (DER12);
;    3943  2      T_ADDR [T_DBN_RD] = .T_ADDR [T_DBN_RD] + 1;      ! INCREMENT # OF READS GIVEN
;    3944  2
;    3945  2      MSCP_PKT [.MX1, MSGLEN] = SZ_ELP;                ! PACKET SIZE                          EXECUTE REDDBN PROGRAM
;    3946  2      MSCP_PKT [.MX1, OPCODE] = OP_ELP;                ! OPCODE = EXECUTE LOCAL PROGRAM
;    3947  2      MSCP_PKT [.MX1, L1] = %ascii'R';                 ! FILL IN PROGRAM NAME WITH ASCII LETTERS
;    3948  2      MSCP_PKT [.MX1, L2] = %ascii'E';
;    3949  2      MSCP_PKT [.MX1, L3] = %ascii'D';
;    3950  2      MSCP_PKT [.MX1, L4] = %ascii'D';
;    3951  2      MSCP_PKT [.MX1, L5] = %ascii'B';
;    3952  2      MSCP_PKT [.MX1, L6] = %ascii'N';
;    3953  2      MSCP_PKT [.MX1, MODIFY] = 1;                     ! STANDALONE MODIFIER
;    3954  2      MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;               ! CALL IT IMMEDIATE
;    3955  2      DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;    3956  2
;    3957  2      IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1   !status error
;    3958  2      THEN RETURN;
;    3959  2
;    3960  3      DO (MX1 = GET_PKT (.CCTLR))
;    3961  2      UNTIL (.MX1 GEQ 0);                    ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    3962  2
;    3963  2              MSCP_PKT [.MX1, MSGLEN] = SZ_REC;        ! PACKET SIZE                          RECIEVE DATA
;    3964  2              MSCP_PKT [.MX1, OPCODE] = OP_RCD;        ! OPCODE = RECEIVE DATA
;    3965  2              MSCP_PKT [.MX1, BC_LO] = 80;             ! BYTE COUNT TO BE TRANSFERED EQUALS 2 *****see pg 26 DUP sp
ec
;    3966  2              MSCP_PKT [.MX1, BUF_0] = DUPPKT;         ! LOAD DESCRIBTOR BUFFER
;    3967  2              MSCP_PKT [.MX1, MODIFY] = 0;             !
;    3968  2              MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;       ! CALL IT sequential
;    3969  2              DUPCOMMAND ();                           ! SENDS AND RECEIVES THE COMMAND
;    3970  2
;    3971  2              IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR      !status error
;    3972  2                 (.DUPPKT [DUPTYPE] NEQU 1) OR                         !dup type error
;    3973  3                 (.DUPPKT [DUPMESG] NEQU 5)
;    3974  2              THEN
;    3975  3                      (HARD_ERROR ();
;    3976  3                       CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    3977  2                       RETURN;);                                      ! NO POINT IN CONTINUING
;    3978  2
```

```
;    3979  3          DO (MX1 = GET_PKT (.CCTLR))
;    3980  2          UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    3981  2
;    3982  2          MSCP_PKT [.MX1, MSGLEN] = SZ_SEN;            ! PACKET SIZE                              SEND DATA
;    3983  2          MSCP_PKT [.MX1, OPCODE] = OP_SDD;            ! OPCODE = SEND DATA
;    3984  2          MSCP_PKT [.MX1, BC_LO] = 4;                  ! BYTE COUNT TO BE TRANSFERED EQUALS 4
;    3985  2          MSCP_PKT [.MX1, BUF_0] = DUPPKT;             ! LOAD DESCRIBTOR BUFFER
;    3986  2          DUPPKT [DUPBF0] = .CST_ADDR [.DUOFF, D_DISK_NUM]; ! LOAD UNIT NUMBER (RDRX)
;    3987  2          DUPPKT [DUPBF1] = .CST_ADDR [.DUOFF + OF_DBN, D_DBN];  ! LOAD DBN NUMBER
;    3988  2          MSCP_PKT [.MX1, MODIFY] = 0;                 !
;    3989  2          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;           ! CALL IT sequential
;    3990  2          DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
;    3991  2
;    3992  2          IF .CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1  !status error
;    3993  2          THEN RETURN;
;    3994  2
;    3995  3          DO (MX1 = GET_PKT (.CCTLR))
;    3996  2          UNTIL (.MX1 GEQ 0);                        ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    3997  2
;    3998  2          MSCP_PKT [.MX1, MSGLEN] = SZ_REC;            ! PACKET SIZE                              RECEIVE DATA
;    3999  2          MSCP_PKT [.MX1, OPCODE] = OP_RCD;            ! OPCODE = GET DUST STATUS
;    4000  2          MSCP_PKT [.MX1, BC_LO] = 514;                ! BYTE COUNT TO BE TRANSFERED EQUALS 512
;    4001  2          MSCP_PKT [.MX1, BUF_0] = DUPPKT;             ! LOAD DESCRIBTOR BUFFER
;    4002  2          MSCP_PKT [.MX1, MODIFY] = 0;                 !
;    4003  2          MSCP_PKT [.MX1, MSGTYP] = SEQ_CMD;           ! CALL IT sequential
;    4004  2          DUPCOMMAND ();                               ! SENDS AND RECEIVES THE COMMAND
;    4005  2
;    4006  2          IF (.CST_ADDR [.DUOFF + OF_DBN, DUPERROR] EQL 1) OR      !status error
;    4007  2             (.DUPPKT [DUPTYPE] NEQU 6) OR                 !dup type error
;    4008  3             (.DUPPKT [DUPMESG] NEQU 2)
;    4009  2          THEN
;    4010  3                  (HARD_ERROR ();
;    4011  3                   CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;      ! SET FLAG
;    4012  2                   RETURN;);                                 ! NO POINT IN CONTINUING
;    4013  2
;    4014  3     DO (MX1 = GET_PKT (.CCTLR))
;    4015  2     UNTIL (.MX1 GEQ 0);                          ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRG ERROR
;    4016  2
;    4017  2     T_ADDR [T_BLK_RD] = .T_ADDR [T_BLK_RD] + 1;              !IF DUP NO ERROR THEN INCREMENT COUNTER
;    4018  2
;    4019  1     END;


                                        .SBTTL  DUPREDDBN MULTI-DRIVE TEST ROUTINES
000000  010146                    DUPREDDBN::
                                        MOV     R1, (SP)                ;                                  3926
000002  013700  000000G                 MOV     T.ADDR,R0               ;                                  3943
000006  005260  000064                  INC     64(R0)
000012  013746  000106'                 MOV     MX1, -(SP)              ;                                  3945
000016  012746  000106                  MOV     #106, -(SP)
000022  004737  000000G                 JSR     PC,BL$MUL
000026  012760  000022  000006G         MOV     #22,MSCP.PKT+6(R0)
000034  112760  000003  000022G         MOVB    #3,MSCP.PKT+22(R0)      ;                                  3946
```

```
000042  112760  000122  000026G                MOVB    #122,MSCP.PKT+26(R0)            ;                       3947
000050  112760  000105  000027G                MOVB    #105,MSCP.PKT+27(R0)            ;                       3948
000056  112760  000104  000030G                MOVB    #104,MSCP.PKT+30(R0)            ;                       3949
000064  112760  000104  000031G                MOVB    #104,MSCP.PKT+31(R0)            ;                       3950
000072  112760  000102  000032G                MOVB    #102,MSCP.PKT+32(R0)            ;                       3951
000100  112760  000116  000033G                MOVB    #116,MSCP.PKT+33(R0)            ;                       3952
000106  012760  000001  000024G                MOV     #1,MSCP.PKT+24(R0)              ;                       3953
000114  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)            ;                       3954
000122  004737  000000V                        JSR     PC,DUPCOMMAND                   ;                       3955
000126  013700  001156'                        MOV     DUOFF,R0                        ;                       3957
000132  006300                                 ASL     R0
000134  063700  000000G                        ADD     CST.ADDR,R0
000140  032760  040000  000020                BIT     #40000,20(R0)
000146  001402                                 BEQ     1$
000150  022626                                 CMP     (SP)+,(SP)+                     ;                       3926
000152  000505                                 BR      3$                              ;                       3958
000154  013716  000000G                1$:     MOV     CCTLR,(SP)                      ;                       3960
000160  004737  000000G                        JSR     PC,GET.PKT
000164  010037  000106'                        MOV     R0,MX1
000170  002771                                 BLT     1$                              ;                       3961
000172  010016                                 MOV     R0,(SP)                         ;                       3963
000174  012746  000106                        MOV     #106,-(SP)
000200  004737  000000G                        JSR     PC,BL$MUL
000204  012760  000034  000006G                MOV     #34,MSCP.PKT+6(R0)
000212  112760  000005  000022G                MOVB    #5,MSCP.PKT+22(R0)              ;                       3964
000220  012760  000120  000026G                MOV     #120,MSCP.PKT+26(R0)            ;                       3965
000226  012760  000000G  000032G                MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       3966
000234  005060  000024G                        CLR     MSCP.PKT+24(R0)                 ;                       3967
000240  142760  000360  000010G                BICB    #360,MSCP.PKT+10(R0)            ;                       3968
000246  152760  000020  000010G                BISB    #20,MSCP.PKT+10(R0)
000254  004737  000000V                        JSR     PC,DUPCOMMAND                   ;                       3969
000260  013700  001156'                        MOV     DUOFF,R0                        ;                       3971
000264  006300                                 ASL     R0
000266  063700  000000G                        ADD     CST.ADDR,R0
000272  032760  040000  000020                BIT     #40000,20(R0)
000300  001016                                 BNE     2$
000302  013700  000000G                        MOV     DUPPKT,R0                       ;                       3972
000306  042700  007777                        BIC     #7777,R0
000312  020027  010000                        CMP     R0,#10000
000316  001007                                 BNE     2$
000320  013700  000000G                        MOV     DUPPKT,R0                       ;                       3973
000324  042700  170000                        BIC     #170000,R0
000330  020027  000005                        CMP     R0,#5
000334  001415                                 BEQ     4$
000336  004737  000000V                2$:     JSR     PC,HARD.ERROR                   ;                       3975
000342  013700  001156'                        MOV     DUOFF,R0                        ;                       3976
000346  006300                                 ASL     R0
000350  063700  000000G                        ADD     CST.ADDR,R0
000354  052760  040000  000020                BIS     #40000,20(R0)
000362  062706  000006                        ADD     #6,SP                           ;                       39
000366  000501                         3$:     BR      5$                              ;                       39
000370  013716  000000G                4$:     MOV     CCTLR,(SP)                      ;                       39
000374  004737  000000G                        JSR     PC,GET.PKT
```

```
000400  010037  000106'                    MOV     R0,MX1
000404  002771                             BLT     4$                              ;                       3980
000406  010016                             MOV     R0,(SP)                         ;                       3982
000410  012746  000106                     MOV     #106,-(SP)
000414  004737  000000G                    JSR     PC,BL$MUL
000420  012760  000034  000006G           MOV     #34,MSCP.PKT+6(R0)
000426  112760  000004  000022G           MOVB    #4,MSCP.PKT+22(R0)              ;                       3983
000434  012760  000004  000026G           MOV     #4,MSCP.PKT+26(R0)              ;                       3984
000442  012760  000000G 000032G           MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       3985
000450  013701  001156'                    MOV     DUOFF,R1                        ;                       3986
000454  006301                             ASL     R1
000456  063701  000000G                    ADD     CST.ADDR,R1
000462  111137  000000G                    MOVB    (R1),DUPPKT
000466  042737  177760  000000G           BIC     #177760,DUPPKT
000474  013701  001156'                    MOV     DUOFF,R1                        ;                       3987
000500  006301                             ASL     R1
000502  063701  000000G                    ADD     CST.ADDR,R1
000506  116137  000020  000002G           MOVB    20(R1),DUPPKT+2
000514  105037  000003G                    CLRB    DUPPKT+3
000520  005060  000024G                    CLR     MSCP.PKT+24(R0)                 ;                       3988
000524  142760  000360  000010G           BICB    #360,MSCP.PKT+10(R0)            ;                       3989
000532  152760  000020  000010G           BISB    #20,MSCP.PKT+10(R0)
000540  004737  000000V                    JSR     PC,DUPCOMMAND                   ;                       3990
000544  013700  001156'                    MOV     DUOFF,R0                        ;                       3992
000550  006300                             ASL     R0
000552  063700  000000G                    ADD     CST.ADDR,R0
000556  032760  040000  000020           BIT     #40000,20(R0)
000564  001403                             BEQ     6$
000566  062706  000010                     ADD     #10,SP                          ;                       3926
000572  000521                     5$:     BR      10$                             ;                       3993
000574  013716  000000G             6$:     MOV     CCTLR,(SP)                      ;                       3995
000600  004737  000000G                    JSR     PC,GET.PKT
000604  010037  000106'                    MOV     R0,MX1
000610  002771                             BLT     6$                              ;                       3996
000612  010016                             MOV     R0,(SP)                         ;                       3998
000614  012746  000106                     MOV     #106,-(SP)
000620  004737  000000G                    JSR     PC,BL$MUL
000624  012760  000034  000006G           MOV     #34,MSCP.PKT+6(R0)
000632  112760  000005  000022G           MOVB    #5,MSCP.PKT+22(R0)              ;                       3999
000640  012760  001002  000026G           MOV     #1002,MSCP.PKT+26(R0)           ;                       4000
000646  012760  000000G 000032G           MOV     #DUPPKT,MSCP.PKT+32(R0)         ;                       4001
000654  005060  000024G                    CLR     MSCP.PKT+24(R0)                 ;                       4002
000660  142760  000360  000010G           BICB    #360,MSCP.PKT+10(R0)            ;                       4003
000666  152760  000020  000010G           BISB    #20,MSCP.PKT+10(R0)
000674  004737  000000V                    JSR     PC,DUPCOMMAND                   ;                       4004
000700  013700  001156'                    MOV     DUOFF,R0                        ;                       4006
000704  006300                             ASL     R0
000706  063700  000000G                    ADD     CST.ADDR,R0
000712  032760  040000  000020           BIT     #40000,20(R0)
000720  001016                             BNE     7$
000722  013700  000000G                    MOV     DUPPKT,R0                       ;                       4007
000726  042700  007777                     BIC     #7777,R0
000732  020027  060000                     CMP     R0,#60000
```

```
000736  001007                          BNE     7$
000740  013700  000000G                 MOV     DUPPKT,R0               ;                       4008
000744  042700  170000                  BIC     #170000,R0
000750  020027  000002                  CMP     R0,#2
000754  001413                          BEQ     8$
000756  004737  000000V          7$:     JSR     PC,HARD.ERROR           ;                       4010
000762  013700  001156'                 MOV     DUOFF,R0                ;                       4011
000766  006300                          ASL     R0
000770  063700  000000G                 ADD     CST.ADDR,R0
000774  052760  040000  000020          BIS     #40000,20(R0)
001002  000413                          BR      9$                      ;                       4012
001004  013716  000000G          8$:     MOV     CCTLR,(SP)              ;                       4014
001010  004737  000000G                 JSR     PC,GET.PKT
001014  010037  000106'                 MOV     R0,MX1
001020  002771                          BLT     8$                      ;                       4015
001022  013700  000000G                 MOV     T.ADDR,R0               ;                       4017
001026  005260  000062                  INC     62(R0)
001032  062706  000012          9$:     ADD     #12,SP                  ;                       3941
001036  012601                 10$:     MOV     (SP)+,R1                ;                       3926
001040  000207                          RTS     PC

; Routine Size:  273 words,     Routine Base:  $CODE$ + 14674
; Maximum stack depth per invocation:  7 words

;   4020  1
```

```
;    4021  1
;    4022  1     GLOBAL ROUTINE DUPCOMMAND : NOVALUE =
;    4023  1
;    4024  1     !+
;    4025  1     ! THIS ROUTINE IS CALLED BY DUP TO PROCESS COMMANDS.
;    4026  1     ! THE COMMAND ENVOLOPES ARE FILLED IN DUP ROUTINES IN THE "MX1" INDEX.
;    4027  1     ! WITH THE INDEX THIS ROUTINE SENDS THE COMMAND, WAITS FOR A
;    4028  1     ! RESPONSES AND THEN PROCESSES THE RETURN PACKET.
;    4029  1     !-
;    4030  2     BEGIN
;    4031  2     !PRINTX (DER13);
;    4032  2
;    4033  2     MSCP_PKT [.MX1, CREDITS] = 0;                   ! DUP DOES NOT USE THE CREDIT SYSTEM
;    4034  2     MSCP_PKT [.MX1, CONNID] = CID_DUP;              ! MAKE PACKAGE EQUAL A DUP COMMAND
;    4035  2     MSCP_PKT [.MX1, DK_NUM] = 0;                    ! DISK NUMBER (NOT APPLICABLE)
;    4036  2
;    4037  2     IF SEND (.MX1) EQLU FAILURE                     ! ATTEMPT SEND; IF CTLR IS OFFLINE
;    4038  2     THEN
;    4039  3             BEGIN
;    4040  3             PUT_PKT (.MX1);
;    4041  3             MX1 = -1;                               ! RETURN ENVELOPE TO POOL
;    4042  3             CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;
;    4043  3             PRINTF (DBM112);                        ! "DUP; PKT NOT AVAILABLE"
;    4044  3             END
;    4045  3
;    4046  2     ELSE
;    4047  2             do
;    4048  3             begin
;    4049  3             BREAK;                                  ! BREAK FOR ACT
;    4050  3             PROC_RETPKT ();                                 ! PROCESS RETURN PACKET TO SEE IF OK FOR DUP
;    4051  3             end
;    4052  2             until (.CRN_LOW eqlu .RP_ADDR [CRF_LO]) or      ! TO ENSURE THAT ALL RETURN MESSAGES HAVE BEEN PROCESSED
;    4053  2             (.EOP_FLAG eql true);                   ! or end of pass caused by error
;    4054  1     END;


                                .SBTTL  DUPCOMMAND MULTI-DRIVE TEST ROUTINES
000000  013746  000106'      DUPCOMMAND::
                                MOV     MX1,-(SP)               ;                       4033
000004  012746  000106          MOV     #106,-(SP)
000010  004737  000000G          JSR     PC,BL$MUL
000014  142760  000017  000010G  BICB    #17,MSCP.PKT+10(R0)
000022  112760  000002  000011G  MOVB    #2,MSCP.PKT+11(R0)      ;                       4034
000030  005060  000016G          CLR     MSCP.PKT+16(R0)         ;                       4035
000034  013716  000106'          MOV     MX1,(SP)                ;                       4037
000040  004737  000000G          JSR     PC,SEND
000044  005700                  TST     R0
000046  001027                  BNE     1$
000050  013716  000106'          MOV     MX1,(SP)                ;                       4040
000054  004737  000000G          JSR     PC,PUT.PKT
000060  012737  177777  000106'  MOV     #-1,MX1                 ;                       4041
000066  013700  001156'          MOV     DUOFF,R0                ;                       4042
000072  006300                  ASL     R0
```

```
000074  063700  000000G                        ADD     CST.ADDR,R0
000100  052760  040000  000020                 BIS     #40000,20(R0)
000106  012716  000000G                        MOV     #DBM112,(SP)            ;                       4043
000112  012746  000001                         MOV     #1,-(SP)
000116  010600                                 MOV     SP,R0                   ; SP,*
000120  104417                                 TRAP    17
000122  005726                                 TST     (SP)+                   ;                       4039
000124  000415                                 BR      2$                      ;                       4037
000126  104422                         1$:     TRAP    22                      ;                       4048
000130  004737  000000V                        JSR     PC,PROC.RETPKT          ;                       4050
000134  013700  000000G                        MOV     RP.ADDR,R0              ;                       4052
000140  023760  000000G 000004                 CMP     CRN.LOW,4(R0)
000146  001404                                 BEQ     2$
000150  123727  000000G 000001                 CMPB    EOP.FLAG,#1             ;                       4053
000156  001363                                 BNE     1$
000160  022626                         2$:     CMP     (SP)+,(SP)+             ;                       4030
000162  000207                                 RTS     PC                      ;                       4022

; Routine Size:  58 words,      Routine Base:  $CODE$ + 15736
; Maximum stack depth per invocation:  5 words
```

```
;    4055  1
;    4056  1     GLOBAL ROUTINE DUPIDLE : NOVALUE =
;    4057  1     !+
;    4058  1     ! THIS ROUTINE IS CALLED BY DUP ROUTINE TO INSURE THAT THE CONTROLLER
;    4059  1     ! IS NOT IN A ACTIVE STATE. IF CALLED AND THE CONTROLLER IS IN AN ACTIVE
;    4060  1     ! STATE THE CONTROLLER WILL GIVE AN ABORT COMMAND WHICH SHOULD KILL THE
;    4061  1     ! CURRENT JOB OR LOCAL PROGRAM.
;    4062  1     !-
;    4063  2     BEGIN
;    4064  2     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 0;                !CLEAR DUP ERROR FLAG;
;    4065  2
;    4066  2     MSCP_PKT [.MX1, MSGLEN] = SZ_GDS;                ! PACKET SIZE                                  GET DUST STATUS
;    4067  2     MSCP_PKT [.MX1, OPCODE] = OP_GDS;                ! OPCODE = GET DUST STATUS
;    4068  2     MSCP_PKT [.MX1, MODIFY] = 0;                     !
;    4069  2     MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;               ! CALL IT IMMEDIATE
;    4070  2     DUPCOMMAND ();                                   ! SENDS AND RECEIVES THE COMMAND
;    4071  2                                                      ! GDS ONLY RETURNS SUCCESS or it don't return
;    4072  2
;    4073  3     DO (MX1 = GET_PKT (.CCTLR))
;    4074  2     UNTIL (.MX1 GEQ 0);                              ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;    4075  2
;    4076  2     if .CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] neq IDLE        ! if not in idle state then abort the program
;    4077  2     then
;    4078  3         begin
;    4079  3         MSCP_PKT [.MX1, MSGLEN] = SZ_ABT;                    ! PACKET SIZE                                  ABORT CMD
;    4080  3         MSCP_PKT [.MX1, OPCODE] = OP_ABT;                    ! OPCODE = ABORT PROGRAM
;    4081  3         MSCP_PKT [.MX1, MODIFY] = 0;                         !
;    4082  3         MSCP_PKT [.MX1, MSGTYP] = IMM_CMD;                   ! CALL IT IMMEDIATE
;    4083  3         DUPCOMMAND ();                                       ! SENDS AND RECEIVES THE COMMAND
;    4084  3                                                              !ONLY ERROR IS already in idle state
;    4085  4         DO (MX1 = GET_PKT (.CCTLR))
;    4086  3         UNTIL (.MX1 GEQ 0);                                  ! TRY TO GET AN ENVELOPE. IF FAILURE LOOP PRGRAM ERROR
;    4087  2         end;
;    4088  1     end;


                                        .SBTTL  DUPIDLE MULTI-DRIVE TEST ROUTINES
000000  010146                 DUPIDLE::
                                        MOV     R1,-(SP)                ;                                       4056
000002  013700  001156'                 MOV     DUOFF,R0                ;                                       4064
000006  006300                          ASL     R0
000010  063700  000000G                 ADD     CST.ADDR,R0
000014  042760  040000  000020          BIC     #40000,20(R0)
000022  013746  000106'                 MOV     MX1,-(SP)               ;                                       4066
000026  012746  000106                  MOV     #106,-(SP)
000032  004737  000000G                 JSR     PC,BL$MUL
000036  012760  000014  000006G         MOV     #14,MSCP.PKT+6(R0)
000044  112760  000001  000022G         MOVB    #1,MSCP.PKT+22(R0)      ;                                       4067
000052  005060  000024G                 CLR     MSCP.PKT+24(R0)         ;                                       4068
000056  142760  000360  000010G         BICB    #360,MSCP.PKT+10(R0)    ;                                       4069
000064  004737  015736'                 JSR     PC,DUPCOMMAND           ;                                       4070
000070  013716  000000G         1$:     MOV     CCTLR,(SP)              ;                                       4073
000074  004737  000000G                 JSR     PC,GET.PKT
```

```
000100  010037  000106'                          MOV     R0,MX1
000104  010001                                   MOV     R0,R1                   ; MX1,*                 4074
000106  002770                                   BLT     1$
000110  013700  001156'                          MOV     DUOFF,R0                ;                       4076
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)                 ;                       4079
000134  012746  000106                          MOV     #106,-(SP)
000140  004737  000000G                          JSR     PC,BL$MUL
000144  012760  000014  000006G                  MOV     #14,MSCP.PKT+6(R0)
000152  112760  000006  000022G                  MOVB    #6,MSCP.PKT+22(R0)      ;                       4080
000160  005060  000024G                          CLR     MSCP.PKT+24(R0)         ;                       4081
000164  142760  000360  000010G                  BICB    #360,MSCP.PKT+10(R0)    ;                       4082
000172  004737  015736'                          JSR     PC,DUPCOMMAND           ;                       4083
000176  013716  000000G                2$:       MOV     CCTLR,(SP)              ;                       4085
000202  004737  000000G                          JSR     PC,GET.PKT
000206  010037  000106'                          MOV     R0,MX1
000212  002771                                   BLT     2$                      ;                       4086
000214  005726                                   TST     (SP)+                   ;                       4078
000216  022626                        3$:       CMP     (SP)+,(SP)+             ;                       4063
000220  012601                                   MOV     (SP)+,R1                ;                       4056
000222  000207                                   RTS     PC

; Routine Size:  74 words,      Routine Base:  $CODE$ + 16122
; Maximum stack depth per invocation:  5 words
```

```
;    4089  1     GLOBAL ROUTINE QIO_LBN : NOVALUE =
;    4090  1
;    4091  1     !+
;    4092  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
;    4093  1     !       BE USED FOR THE CURRENT QIO OR QIO PAIR.
;    4094  1     !
;    4095  1     !       IF THE OPERATOR CHOSE THE RANDOM BLOCK MODE OPTION, THEN THE LBN IS
;    4096  1     !       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR THE LBN.
;    4097  1     !       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
;    4098  1     !       TABLE (BST).
;    4099  1     !
;    4100  1     !       IMPLICIT INPUTS:
;    4101  1     !               L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
;    4102  1     !
;    4103  1     !       IMPLICIT OUTPUTS:
;    4104  1     !               THE LBN IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4105  1     !-
;    4106  1
;    4107  2     begin
;    4108  2
;    4109  2     local
;    4110  2         RD_DISK : byte;      ! FLAG TO INDICATE WINCHESTER DISK SELECTED
;    4111  2
;    4112  2
;    4113  2     if .CST_ADDR [.CUOFF, D_TYPE] eql RX_50
;    4114  2     then
;    4115  2         RD_DISK = FALSE
;    4116  2     else
;    4117  2         RD_DISK = TRUE;
;    4118  2
;    4119  2     MAD1 [LBN_L] = .BST [.L$LUN, LO_WRD];                   ! LOAD LBN INTO 1ST PACKET
;    4120  2     MAD1 [LBN_H] = .BST [.L$LUN, HI_WRD];                   ! LOAD LBN INTO 1ST PACKET
;    4121  2
;    4122  2     if .MX2 geq 0                               ! IF 2 QIOS
;    4123  2     then
;    4124  3         begin
;    4125  3         MAD2 [LBN_L] = .BST [.L$LUN, LO_WRD];                       ! LOAD LBN INTO 2ND PACKET
;    4126  3         MAD2 [LBN_H] = .BST [.L$LUN, HI_WRD];               ! LOAD LBN INTO 2ND PACKET
;    4127  2         end;
;    4128  2
;    4129  3     if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF RANDOM BLOCK MODE
;    4130  2     then
;    4131  3         begin
;    4132  4         if NOT ((.RD_DISK) and
;    4133  4           (((.RANDOM [0] and %o'077777') mod (99)) lequ 33))                    ! REDUCES SEEKS TO 66% OF TRANFERS o
n winchesters
;    4134  3         then
;    4135  4             begin                                                               ! NOTE IF NOT CALLED THE LBN REMAINS
 THE SAME
;    4136  4             if .CST_ADDR [.CUOFF + 2, D_BEG1] eqlu .CST_ADDR [.CUOFF + 4, D_END1]
;    4137  4             then                                                                ! if upper word of beg trk and endin
g trk same
;    4138  4                     BST [.L$LUN, LO_WRD] = .CST_ADDR [.CUOFF + 1, D_BEG0] +             ! select low lbn from random
 number WINDOW
;    4139  4                     MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], .CST_ADDR [.CUOFF + 3, D_END0])
;    4140  4
;    4141  4             else                                                                ! if upper word of beg trk and end t
rk different
```

```
;    4142  5                begin
;    4143  5                BST [.L$LUN, HI_WRD] = .CST_ADDR [.CUOFF + 2, D_BEG1] +              ! select upper lbn from window
;    4144  5                MODULAS (.CST_ADDR [.CUOFF + 2, D_BEG1], .CST_ADDR [.CUOFF + 4, D_END1]);
;    4145  5
;    4146  5                if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 4, D_END1]          ! IF UPPER WORD EQUALS HI LIMIT BE S
URE LOWER
;    4147  5                then .BST [.L$LUN, LO_WRD] = MODULAS (0, .CST_ADDR [.CUOFF + 3, D_END0]); ! WORD DOES NOT PASS HI LIMIT
;    4148  5
;    4149  5                if .BST [.L$LUN, HI_WRD] eql .CST_ADDR [.CUOFF + 2, D_BEG1]          ! if upper word equal lo limit make
sure lower
;    4150  5                then BST [.L$LUN, LO_WRD] = %O'177777' - MODULAS (.CST_ADDR [.CUOFF + 1, D_BEG0], %O'177777');
;    4151  5                                                                                     ! word is above lo limit
;    4152  5
;    4153  5                if .BST [.L$LUN, HI_WRD] gtr .CST_ADDR [.CUOFF + 2, D_BEG1] and
;    4154  5                   .BST [.L$LUN, HI_WRD] lss .CST_ADDR [.CUOFF + 4, D_END1]          ! if neither of the above then any n
umber is good
;    4155  5                then BST [.L$LUN, LO_WRD] = .RANDOM [5];
;    4156  4                end;
;    4157  3            end;
;    4158  3        end
;    4159  2    else
;    4160  3        begin                                                ! ELSE - SEQUENTIAL LBN MODE
;    4161  4        if (.TRK_SGN [.L$LUN] geq 1)
;    4162  3        then                                                 ! if positive track direction add one to multiword
;    4163  4            (if .BST [.L$LUN, LO_WRD] eqlu %o'177777'
;    4164  4            then
;    4165  5                    begin
;    4166  5                    BST [.L$LUN, LO_WRD] = 0;
;    4167  5                    BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] + 1;
;    4168  5                    end
;    4169  4                else
;    4170  4                    BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] + 1)
;    4171  3        else                                                 ! if negative track direction subtract one from multiword
;    4172  3            if .BST [.L$LUN, LO_WRD] eqlu %o'0'
;    4173  3            then
;    4174  4                    begin
;    4175  4                    BST [.L$LUN, LO_WRD] = %o'177777';
;    4176  4                    BST [.L$LUN, HI_WRD] = .BST [.L$LUN, HI_WRD] - 1;
;    4177  4                    end
;    4178  3            else
;    4179  3                    BST [.L$LUN, LO_WRD] = .BST [.L$LUN, LO_WRD] - 1;
;    4180  3
;    4181  3
;    4182  3        if  .BST [.L$LUN, LO_WRD] gequ (.CST_ADDR [.CUOFF + 3, D_END0]) and ! if hi limit then change direction
;    4183  4            .BST [.L$LUN, HI_WRD] gequ (.CST_ADDR [.CUOFF + 4, D_END1])
;    4184  3        then TRK_SGN [.L$LUN] =  -1;
;    4185  3
;    4186  3        if  .BST [.L$LUN, LO_WRD] lequ (.CST_ADDR [.CUOFF + 1, D_BEG0] + 1)  and    ! if low limit then change direction
;    4187  4            .BST [.L$LUN, HI_WRD] lequ (.CST_ADDR [.CUOFF + 2, D_BEG1])
;    4188  3        then TRK_SGN [.L$LUN] = 1;
;    4189  2        end;
;    4190  1    end;                                                     ! ROUTINE QIO_LBN


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

```
                                                JSR     R1,$SAVE5                       ;                               4089
000004  013705  000000G                        MOV     CST.ADDR,R5                     ;                               4113
000010  013700  000000G                        MOV     CUOFF,R0
000014  006300                                  ASL     R0
000016  060500                                  ADD     R5,R0
000020  132710  000020                         BITB    #20,(R0)
000024  001002                                  BNE     1$
000026  105004                                  CLRB    R4                              ; RD.DISK                       4115
000030  000402                                  BR      2$                              ;                               4117
000032  112704  000001              1$:         MOVB    #1,R4                           ; *,RD.DISK                     4117
000036  013700  000112'              2$:         MOV     MAD1,R0                         ;                               4119
000042  013701  000000G                        MOV     L$LUN,R1
000046  010103                                  MOV     R1,R3
000050  006303                                  ASL     R3
000052  006303                                  ASL     R3
000054  012702  000000G                        MOV     #BST,R2
000060  060302                                  ADD     R3,R2
000062  011260  000046                         MOV     (R2),46(R0)
000066  062703  000002G                        ADD     #BST+2,R3                       ;                               4120
000072  011360  000050                         MOV     (R3),50(R0)
000076  005737  000110'                        TST     MX2                             ;                               4122
000102  002406                                  BLT     3$
000104  013700  000114'                        MOV     MAD2,R0                         ;                               4125
000110  011260  000046                         MOV     (R2),46(R0)
000114  011360  000050                         MOV     (R3),50(R0)                     ;                               4126
000120  032737  040000  000000G      3$:       BIT     #40000,SWP.FLAGS                ;                               4129
000126  001002                                  BNE     4$
000130  000137  017122'                        JMP     10$
000134  006004                      4$:         ROR     R4                              ; RD.DISK                       4132
000136  103015                                  BCC     5$
000140  013746  000000G                        MOV     RANDOM,-(SP)                    ;                               4133
000144  042716  100000                         BIC     #100000,(SP)
000150  012746  000143                         MOV     #143,-(SP)
000154  004737  000000G                        JSR     PC,BL$MOD
000160  022626                                  CMP     (SP)+,(SP)+
000162  020027  000041                         CMP     R0,#41
000166  101001                                  BHI     5$
000170  000207                                  RTS     PC
000172  013700  000000G              5$:         MOV     CUOFF,R0                        ;                               4136
000176  006300                                  ASL     R0
000200  060500                                  ADD     R5,R0
000202  016004  000004                         MOV     4(R0),R4
000206  013700  000000G                        MOV     CUOFF,R0
000212  006300                                  ASL     R0
000214  060500                                  ADD     R5,R0
000216  020460  000010                         CMP     R4,10(R0)
000222  001022                                  BNE     6$
000224  013701  000000G                        MOV     CUOFF,R1                        ;                               4138
000230  006301                                  ASL     R1
000232  060501                                  ADD     R5,R1
000234  016146  000002                         MOV     2(R1),-(SP)                     ;                               4139
000240  013700  000000G                        MOV     CUOFF,R0
000244  006300                                  ASL     R0
```

```
000246  060500                          ADD     R5,R0
000250  016046  000006                  MOV     6(R0), (SP)
000254  004737  0000006                 JSR     PC,MODULAS
000260  066100  000002                  ADD     2(R1),R0                ;                       4138
000264  010012                          MOV     R0,(R2)
000266  000530                          BR      9$                      ;                       4139
000270  010446                  6$:     MOV     R4, (SP)                ;                       4144
000272  016046  000010                  MOV     10(R0), (SP)
000276  004737  0000006                 JSR     PC,MODULAS
000302  060400                          ADD     R4,R0                   ;                       4143
000304  010013                          MOV     R0,(R3)
000306  013701  0000006                 MOV     L$LUN,R1                ;                       4146
000312  006301                          ASL     R1
000314  006301                          ASL     R1
000316  013700  0000006                 MOV     CUOFF,R0
000322  006300                          ASL     R0
000324  063700  0000006                 ADD     CST.ADDR,R0
000330  026160  0000026 000010          CMP     BST+2(R1),10(R0)
000336  001015                          BNE     7$
000340  005016                          CLR     (SP)                    ;                       4147
000342  013700  0000006                 MOV     CUOFF,R0
000346  006300                          ASL     R0
000350  063700  0000006                 ADD     CST.ADDR,R0
000354  016046  000006                  MOV     6(R0), (SP)
000360  004737  0000006                 JSR     PC,MODULAS
000364  010071  0000006                 MOV     R0,@BST(R1)
000370  005726                          TST     (SP)+
000372  013701  0000006         7$:     MOV     L$LUN,R1                ;                       4149
000376  006301                          ASL     R1
000400  006301                          ASL     R1
000402  013700  0000006                 MOV     CUOFF,R0
000406  006300                          ASL     R0
000410  063700  0000006                 ADD     CST.ADDR,R0
000414  026160  0000026 000004          CMP     BST+2(R1),4(R0)
000422  001021                          BNE     8$
000424  013700  0000006                 MOV     CUOFF,R0                ;                       4150
000430  006300                          ASL     R0
000432  063700  0000006                 ADD     CST.ADDR,R0
000436  016016  000002                  MOV     2(R0),(SP)
000442  012746  177777                  MOV     #-1, (SP)
000446  004737  0000006                 JSR     PC,MODULAS
000452  012761  177777  0000006         MOV     #-1,BYT(R1)
000460  160061  0000006                 SUB     R0,BYT(R1)
000464  005726                          TST     (SP)+
000466  013700  0000006         8$:     MOV     L$LUN,R0                ;                       4155
000472  006300                          ASL     R0
000474  006300                          ASL     R0
000476  013701  0000006                 MOV     CUOFF,R1
000502  006301                          ASL     R1
000504  063701  0000006                 ADD     CST.ADDR,R1
000510  026061  0000026 000004          CMP     BST+2(R0),4(R1)
000516  003414                          BLE     9$
000520  013701  0000006                 MOV     CUOFF,R1                ;                       4154
```

```
000534  006301                          ASL     R1
000536  063701  000000G                 ADD     CST,ADDR,R1
000542  026061  000002G 000010          CMP     BST+2(R0),10(R1)
000550  002003                          BGE     9$
000552  013760  000012G 000000G         MOV     RANDOM+12,BST(R0)       ;                       4155
000560  020626                  9$:     CMP     (SP)+,(SP)+             ;                       4135
000562  000207                          RTS     PC                      ;                       4139
000564  062701  000000G         10$:    ADD     #TRK.SGN,R1             ;                       4161
000570  105711                          TSTB    (R1)
000572  003410                          BLE     12$
000574  021227  177777                  CMP     (R2),#-1                ;                       4163
000600  001003                          BNE     11$
000602  005012                          CLR     (R2)                    ;                       4166
000604  005213                          INC     (R3)                    ;                       4167
000606  000411                          BR      14$                     ;                       4163
000610  005212                  11$:    INC     (R2)                    ;                       4170
000612  000407                          BR      14$                     ;                       4161
000614  005712                  12$:    TST     (R2)                    ;                       4172
000616  001004                          BNE     13$
000620  012712  177777                  MOV     #-1,(R2)                ;                       4175
000624  005313                          DEC     (R3)                    ;                       4176
000626  000401                          BR      14$                     ;                       4172
000630  005312                  13$:    DEC     (R2)                    ;                       4179
000632  013700  000000G         14$:    MOV     C$OFF,R0                ;                       4182
000636  006300                          ASL     R0
000640  060500                          ADD     R5,R0
000642  021260  000006                  CMP     (R2),6(R0)
000646  103411                          BLO     15$
000650  013700  000000G                 MOV     C$OFF,R0                ;                       4183
000654  006300                          ASL     R0
000656  060500                          ADD     R5,R0
000660  021360  000010                  CMP     (R3),10(R0)
000664  103402                          BLO     15$
000666  112711  000377                  MOVB    #377,(R1)               ;                       4184
000672  013700  000000G         15$:    MOV     C$OFF,R0                ;                       4186
000676  006300                          ASL     R0
000700  060500                          ADD     R5,R0
000702  016000  000002                  MOV     2(R0),R0
000706  005200                          INC     R0
000710  021200                          CMP     (R2),R0
000712  101011                          BHI     16$
000714  013700  000000G                 MOV     C$OFF,R0                ;                       4187
000720  006300                          ASL     R0
000722  060500                          ADD     R5,R0
000724  021360  000004                  CMP     (R3),4(R0)
000730  101002                          BHI     16$
000732  112711  000001                  MOVB    #1,(R1)                 ;                       4188
000736  000207                  16$:    RTS     PC                      ;                       4089

; Routine Size:  256 words,    Routine Base:  $CODE$ + 16346
; Maximum stack depth per invocation:  10 words
```

```
;    4191  1     !!ZZZ routine QIO_SIZE : novalue =
;    4192  1     GLOBAL ROUTINE QIO_SIZE : NOVALUE =
;    4193  1
;    4194  1     !+
;    4195  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;    4196  1     !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR.  THE BYTE COUNT IS
;    4197  1     !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS FALL BETWEEN 1 AND THE
;    4198  1     !       I/O BUFFER SIZE (BYTS_PER_QIO).  It is assumed that BYTS_PER_QIO will
;    4199  1     !       never be larger than one binary word or  65000 bytes.
;    4200  1     !
;    4201  1     !       IMPLICIT OUTPUTS:
;    4202  1     !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP PACKETS.
;    4203  1     !-
;    4204  1
;    4205  2         begin
;    4206  2
;    4207  2         local
;    4208  2             SIZE : word,                                                    ! BYTE COUNT
;    4209  2             BLOCKS_LEFT : word;                                             ! REMAINING BLOCKS LEFT
;    4210  2
;    4211  2         SIZE = ((.RANDOM [4] and %o'077777') mod (.BYTS_PER_QIO + 1)) and %o'177760';!GET BYTE COUNT FROM RANDOM NUMBER
;    4212  2
;    4213  2         if .SIZE eql 0
;    4214  2         then
;    4215  2             SIZE = 16;
;    4216  2
;    4217  2         if .CST_ADDR [.CUOFF + 4, D_END1] gtru .MAD1 [LBN_H]
;    4218  2         then BLOCKS_LEFT = %c'177777'                                       ! find
;    4219  2         else BLOCKS_LEFT = .CST_ADDR [.CUOFF + 3, D_END0] - .MAD1 [LBN_L] + 1;          ! REMAINING BLOCK COUNT
;    4220  2
;    4221  2         if ((.SIZE + BYTES_PER_SECT - 1) / BYTES_PER_SECT) gtru .BLOCKS_LEFT    ! IF BLOCK COUNT NOT ENOUGH
;    4222  2         then
;    4223  2             SIZE = .BLOCKS_LEFT * BYTES_PER_SECT;                           ! ADJUST BYTE COUNT DOWN
;    4224  2
;    4225  2         MAD1 [BC_LO] = .SIZE;                                               ! LOAD SIZE INTO 1ST MSCP PACKET
;    4226  2
;    4227  2         if .MX2 geq 0                                                       ! IF 2 QIOS
;    4228  2         then
;    4229  2             MAD2 [BC_LO] = .SIZE;                                           ! LOAD SIZE INTO 2ND MSCP PACKET
;    4230  2
;    4231  1         end;                                                                ! ROUTINE QIO_SIZE

                                .SBTTL  QIO.SIZE MULTI DRIVE TEST ROUTINES
000000  004137  000000G    QIO.SIZE::
                            JSR     R1,$SAVE3                       ;                       4192
000004  013746  000010G             MOV     RANDOM+10,-(SP)         ;                       4211
000010  042716  100000              BIC     #100000,(SP)
000014  013746  000000G             MOV     BYTS.PER.QIO,-(SP)
000020  005216                      INC     (SP)
000022  004737  000000G             JSR     PC,BL$MOD
000026  010003                      MOV     R0,R3                   ; *,SIZE
000030  042703  000017              BIC     #17,R3                  ; *,SIZE
```

```
000034  001002                          BNE     1$                              ;                               4213
000036  012703  000020                  MOV     #20,R3                          ; *,SIZE                        4215
000042  013700  000000G         1$:     MOV     CUOFF,R0                        ;                               4217
000046  006300                          ASL     R0
000050  063700  000000G                 ADD     CST.ADDR,R0
000054  013701  000112'                  MOV     MAD1,R1
000060  026061  000010  000050          CMP     10(R0),50(R1)
000066  101403                          BLOS    2$
000070  012702  177777                  MOV     #-1,R2                          ; *,BLOCKS.LEFT                 4218
000074  000413                          BR      3$                              ;                               4217
000076  013700  000000G         2$:     MOV     CUOFF,R0                        ;                               4219
000102  006300                          ASL     R0
000104  063700  000000G                 ADD     CST.ADDR,R0
000110  016000  000006                  MOV     6(R0),R0
000114  166100  000046                  SUB     46(R1),R0
000120  010002                          MOV     R0,R2                           ; *,BLOCKS.LEFT
000122  005202                          INC     R2                              ; BLOCKS.LEFT
000124  010316                  3$:     MOV     R3,(SP)                         ; SIZE,*                        4221
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,*                 4223
000152  000300                          SWAB    R0
000154  105000                          CLRB    R0
000156  006300                          ASL     R0
000160  010003                          MOV     R0,R3                           ; *,SIZE
000162  010361  000026          4$:     MOV     R3,26(R1)                       ; SIZE,*                        4225
000166  005737  000110'                 TST     MX2                             ;                               4227
000172  002404                          BLT     5$
000174  013700  000114'                 MOV     MAD2,R0                         ;                               4229
000200  010360  000026                  MOV     R3,26(R0)                       ; SIZE,*
000204  022626                  5$:     CMP     (SP)+,(SP)+                     ;                               4205
000206  000207                          RTS     PC                              ;                               4192

; Routine Size:  68 words,      Routine Base:  $CODE$ + 17276
; Maximum stack depth per invocation:  8 words
```

```
;    4232  1     GLOBAL routine FILL_BUFF : novalue =
;    4233  1
;    4234  1     !+
;    4235  1     !       THIS ROUTINE IS CALLED BY QIO_GEN TO LOAD THE I/O BUFFER DESCRIBED IN
;    4236  1     !       THE FIRST MSCP PACKET WITH THE APPROPRIATE DATA PATTERN.
;    4237  1     !
;    4238  1     !       THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
;    4239  1     !
;    4240  1     !               IF THE OPERATOR DEFINED A DATA PATTERN
;    4241  1     !               THEN
;    4242  1     !                       SELECT IT
;    4243  1     !               ELSE
;    4244  1     !                       GET DATA PATTERN NUMBER FROM SW P-TABLE
;    4245  1     !                       IF DATA PATTERN NUMBER = 0
;    4246  1     !                       THEN
;    4247  1     !                               GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
;    4248  1     !                               IN THE DATA PATTERN SEQUENCE TABLE (DPST)
;    4249  1     !
;    4250  1     !       NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 7 -
;    4251  1     !       21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
;    4252  1     !
;    4253  1     !       IMPLICIT INPUTS:
;    4254  1     !               L$LUN - CURRENT (DRS) UNIT NUMBER
;    4255  1     !-
;    4256  1
;    4257  2         begin
;    4258  2
;    4259  2         local
;    4260  2             DP_NUM : word,                          ! DATA PATTERN NUMBER SELECTED
;    4261  2             DP_ADDR,                                ! ADDR OF DATA PATTERN (LENGTH)
;    4262  2             IOB_ADDR,                               ! I/O BUFFER ADDRESS (DESTINATION)
;    4263  2             SRC_ADDR,                               ! WORKING SOURCE ADDRESS
;    4264  2             COUNT : word;                           ! NO. OF WORDS IN DATA PATTERN
;    4265  2
;    4266  3         if BIT_TST (SWP_FLAGS, SWF_UDP)             ! IF USER DEFINED A DATA PATTERN
;    4267  2         then
;    4268  2             DP_ADDR = SWP_UCNT                      ! SELECT IT
;    4269  2         else
;    4270  3             begin
;    4271  3
;    4272  3             if .SWP_DPAT neq 0                      ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;    4273  3             then
;    4274  3                 DP_NUM = .SWP_DPAT                  ! SELECT IT
;    4275  3             else
;    4276  4                 begin
;    4277  4                 DP_NUM = .DPST [.L$LUN];            ! GET PATTERN NUMBER FROM SEQUENCE TABLE
;    4278  4                 DPST [.L$LUN] = .DPST [.L$LUN] + 1; ! ADVANCE TO NEXT PATTERN NUMBER
;    4279  4
;    4280  4                 if .DPST [.L$LUN] gtru DP_CNT       ! CHECK FOR HIGH LIMIT
;    4281  4                 then
;    4282  4                     DPST [.L$LUN] = 1;
;    4283  4
;    4284  3                 end;
```

```
;  4285  3
;  4286  3          DP_ADDR = .DPA_TBL [.DP_NUM - 1];                ! ADDRESS OF DATA PATTERN (COUNT)
;  4287  3
;  4288  3          if .DP_NUM gequ 17
;  4289  3          then
;  4290  3
;  4291  3              if .DP_NUM                                   ! CHECK MACRO (IF PATTERN 17, 19, OR 21)
;  4292  3              then
;  4293  3                  (.DP_ADDR + 2) = .MAD1 [LBN_L]           ! LOAD LBN INTO FIRST WORD OF PATTERN
;  4294  3              else
;  4295  3                  (.DP_ADDR + 4) = .MAD1 [LBN_L];          ! LOAD LBN INTO SECOND WORD OF PATTERN
;  4296  3
;  4297  2              end;
;  4298  2
;  4299  2          IOB_ADDR = .MAD1 [BUF_O];                        ! I/O BUFFER ADDRESS
;  4300  2          COUNT = ..DP_ADDR;                               ! NO. OF WORDS IN DATA PATTERN
;  4301  2          SRC_ADDR = .DP_ADDR + 2;                         ! START OF THE ACTUAL DATA PATTERN
;  4302  2
;  4303  2          incr N from 1 to ((.MAD1 [BC_LO] + 1) / 2) do    ! FOR EACH WORD IN THIS WRITE REQUEST
;  4304  3              begin
;  4305  3              .IOB_ADDR = ..SRC_ADDR;                      ! MOVE 1 WORD
;  4306  3              IOB_ADDR = .IOB_ADDR + 2;                    ! ADVANCE DESTINATION ADDRESS
;  4307  3              SRC_ADDR = .SRC_ADDR + 2;                    ! ADVANCE SOURCE ADDRESS
;  4308  3              COUNT = .COUNT - 1;                          ! DECREMENT COUNT
;  4309  3
;  4310  3              if .COUNT eql 0                              ! IF END OF DATA PATTERN
;  4311  3              then
;  4312  4                  begin
;  4313  4                  COUNT = ..DP_ADDR;                       ! REPEAT DATA PATTERN
;  4314  4                  SRC_ADDR = .DP_ADDR + 2;
;  4315  3                  end;
;  4316  3
;  4317  2              end;                                         ! WORD TRANSFER LOOP
;  4318  2
;  4319  1          end;                                             ! ROUTINE FILL_BUFF


                                .SBTTL  FILL.BUFF MULTI-DRIVE TEST ROUTINES
000000  004137  000000G       FILL.BUFF::
                                JSR     R1,$SAVE5                   ;                       4232
000004  005746                  TST     -(SP)
000006  032737  000100  000000G  BIT     #100,SWP.FLAGS              ;                       4266
000014  001403                  BEQ     1$
000016  012701  000000G         MOV     #SWP.UCNT,R1                ; *,DP.ADDR             4268
000022  000443                  BR      5$                          ;                       4266
000024  013700  000000G  1$:    MOV     SWP.DPAT,R0                 ;                       4272
000030  001402                  BEQ     2$
000032  010002                  MOV     R0,R2                       ; *,DP.NUM              4274
000034  000414                  BR      3$                          ;                       4266
000036  013700  000000G  2$:    MOV     L$LUN,R0                    ;                       4277
000042  062700  000050'         ADD     #DPST,R0
000046  005002                  CLR     R2                          ; DP.NUM
000050  151002                  BISB    (R0),R2                     ; *,DP.NUM
```

```
000052  105210                          INCB    (R0)                    ;                               4278
000054  121027  000025                  CMPB    (R0),#25                ;                               4280
000060  101402                          BLOS    3$
000062  112710  000001                  MOVB    #1,(R0)                 ;                               4282
000066  010200                  3$:     MOV     R2,R0                   ; DP.NUM,*                      4286
000070  006300                          ASL     R0
000072  016001  001072'                  MOV     DPA.TBL-2(R0),R1        ; *,DP.ADDR
000076  020227  000021                  CMP     R2,#21                  ; DP.NUM,*                      4288
000102  103413                          BLO     5$
000104  013700  000112'                  MOV     MAD1,R0                 ;                               4293
000110  006002                          ROR     R2                      ; DP.NUM                        4291
000112  103004                          BCC     4$
000114  016061  000046  000002          MOV     46(R0),2(R1)            ; *,*(DP.ADDR)                  4295
000122  000403                          BR      5$                      ;                               4291
000124  016061  000046  000004  4$:     MOV     46(R0),4(R1)            ; *,*(DP.ADDR)                  4295
000132  013700  000112'          5$:     MOV     MAD1,R0                 ;                               4299
000136  016004  000032                  MOV     32(R0),R4               ; *,IOB.ADDR
000142  011103                          MOV     (R1),R3                 ; DP.ADDR,COUNT                 4300
000144  012705  000002                  MOV     #2,R5                   ;                               4301
000150  060105                          ADD     R1,R5                   ; DP.ADDR,*
000152  010502                          MOV     R5,R2                   ; *,SRC.ADDR
000154  016046  000026                  MOV     26(R0),-(SP)            ;                               4303
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$:     MOV     (R2)+,(R4)+             ; SRC.ADDR,IOB.ADDR             4305
000204  005303                          DEC     R3                      ; COUNT                         4308
000206  001002                          BNE     7$                      ;                               4310
000210  011103                          MOV     (R1),R3                 ; DP.ADDR,COUNT                 4313
000212  010502                          MOV     R5,R2                   ; *,SRC.ADDR                    4314
000214  005200                  7$:     INC     R0                      ; N                             4303
000216  020066  000004                  CMP     R0,4(SP)                ; N,*
000222  003767                          BLE     6$
000224  062706  000006                  ADD     #6,SP                   ;                               4232
000230  000207                          RTS     PC

; Routine Size:  77 words,    Routine Base:  $CODE$ + 17506
; Maximum stack depth per invocation:  10 words


;   4320  1
;   4321  1
```

```
;  4322  1     GLOBAL ROUTINE PROC_RETPKT : NOVALUE =
;  4323  1
;  4324  1     !+
;  4325  1     !       THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" AND DUP_COMMAND TO CHECK FOR
;  4326  1     !       AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;  4327  1     !       PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;  4328  1     !       BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;  4329  1     !       REQUIRE PROCESSING.
;  4330  1     !
;  4331  1     !       UNDER THE MULTI-DRIVE TEST, RETURN PACKETS ORIGINATE FROM TWO SOURCES:
;  4332  1     !               1. MSCP         - THE MORE COMMON, DESCRIBING A COMPLETED I/O
;  4333  1     !                                 OPERATION.
;  4334  1     !               2. DUP          - THE LESS COMMON, DESCRIBING A PORTION OF I/O
;  4335  1     !                                 COMMUNICATIONS WITH THE CONTROLLER PROGRAM.
;  4336  1     !               3. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;  4337  1     !                                 COMMAND TIMEOUT.
;  4338  1     !-
;  4339  1
;  4340  1         while .IODQ_IN neq .IODQ_OUT do                         ! DO UNTIL I/O DONE QUEUE IS EMPTY
;  4341  2             begin
;  4342  2             RP_INDX = OUT_IODQ ();                              ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;  4343  2             RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);         ! CALCULATE RETPKT ADDRESS
;  4344  3             if NOT (.RP_ADDR [CONID] eql CID_DUP)               ! if not DUP then
;  4345  2             then (SET_CPAR (.RP_ADDR [CTLR]));                  ! SET UP CURRENT CONTROLLER PARAMETERS
;  4346  2
;  4347  2             selectoneu .RP_ADDR [CONID] of                      ! CONNECTION ID INDICATES PACKET SOURCE
;  4348  2                 set
;  4349  2
;  4350  2                 [CID_MSCP] :        IO_RETPKT ();               ! DISK MSCP (I/O TRANSFER DONE)
;  4351  2                 [CID_DUP] :         DIO_RETPKT ();              ! DUP (I/O TRANSFER DONE)
;  4352  2                 [CID_DRIVER] :      DR_RETPKT ();               ! MESSAGE FROM "DRIVER"
;  4353  2
;  4354  2                 [otherwise] :       PRINTF (DBM12, .RP_ADDR [CONID]);!"CONN ID = XXXXX RECEIVED"
;  4355  2                 tes;
;  4356  2
;  4357  1             end;                                                ! UNITL I/O DONE QUEUE IS EMPTY


                                .SBTTL  PROC.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                PROC.RETPKT::
                                MOV     R1,-(SP)                        ;                                       4322
000002  023737  000000G 000000G 1$:     CMP     IODQ.IN,IODQ.OUT        ;                                       4340
000010  001467                        BEQ     7$
000012  004737  000000G                JSR     PC,OUT.IODQ             ;                                       4342
000016  010037  000000G                MOV     R0,RP.INDX
000022  010046                        MOV     R0,-(SP)                ; RP.INDX,*                             4343
000024  012746  000054                MOV     #54,-(SP)
000030  004737  000000G                JSR     PC,BL$MUL
000034  062700  000000G                ADD     #RETPKT,R0
000040  010037  000000G                MOV     R0,RP.ADDR
000044  126027  000003  000002        CMPB    3(R0),#2                ;                                       4344
000052  001406                        BEQ     2$
000054  116016  000002                MOVB    2(R0),(SP)              ;                                       4345
```

```
000060  042716  177760                          BIC     #177760,(SP)
000064  004737  000000G                         JSR     PC,SET.CPAR
000070  013700  000000G             2$:         MOV     RP.ADDR,R0                      ;                       4347
000074  005001                                  CLR     R1
000076  156001  000003                          BISB    3(R0),R1
000102  005701                                  TST     R1                              ;                       4350
000104  001003                                  BNE     3$
000106  004737  000000V                         JSR     PC,IO.RETPKT
000112  000424                                  BR      6$                              ;                       4347
000114  020127  000002              3$:         CMP     R1,#2                           ;                       4351
000120  001003                                  BNE     4$
000122  004737  000000V                         JSR     PC,DIO.RETPKT
000126  000416                                  BR      6$                              ;                       4347
000130  020127  000003              4$:         CMP     R1,#3                           ;                       4352
000134  001003                                  BNE     5$
000136  004737  000000V                         JSR     PC,DR.RETPKT
000142  000410                                  BR      6$                              ;                       4347
000144  010116                      5$:         MOV     R1,(SP)                         ;                       4354
000146  012746  000000G                         MOV     #DBM12,-(SP)
000152  012746  000002                          MOV     #2,-(SP)
000156  010600                                  MOV     SP,R0                           ; SP,*
000160  104417                                  TRAP    17
000162  022626                                  CMP     (SP)+,(SP)+
000164  022626                      6$:         CMP     (SP)+,(SP)+                     ;                       4341
000166  000705                                  BR      1$                              ;                       4340
000170  012601                      7$:         MOV     (SP)+,R1                        ;                       4322
000172  000207                                  RTS     PC

; Routine Size:  62 words,      Routine Base:  $CODE$ + 17740
; Maximum stack depth per invocation:  7 words
```

```
;    4358  1      !+
;    4359  1      GLOBAL ROUTINE DIO_RETPKT : NOVALUE =
;    4360  1
;    4361  1      !+
;    4362  1      !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL DUP I/O TRANSFER
;    4363  1      !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;    4364  1      !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS.
;    4365  1      !
;    4366  1      !       IMPLICIT INPUTS:
;    4367  1      !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    4368  1      !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4369  1      !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    4370  1      !               DUOFF - CST OFFSET FOR THE CURRENT UNIT
;    4371  1      !               L$LUN - CURRENT UNIT NUMBER
;    4372  1      !               CCTLR - CURRENT CONTROLLER NUMBER
;    4373  1      !
;    4374  1      !       IMPLICIT OUTPUTS
;    4375  1      !               CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] - IF THIS BIT SET NO DUP EXERCISER
;    4376  1      !
;    4377  1      !-
;    4378  2      BEGIN
;    4379  2
;    4380  2      LOCAL   FLAG : BYTE INITIAL(BYTE(TRUE)),
;    4381  2              SUM2 : WORD,
;    4382  2              SUM : WORD;                                     ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;    4383  2      !PRINTX (DER18);
;    4384  2
;    4385  2      IF .RP_ADDR [STATUS] NEQU ST_SUC                        ! IF STATUS CODE INDICATES ERROR
;    4386  2      THEN
;    4387  3              BEGIN
;    4388  3              CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;       ! SET DUP ERROR FLAG
;    4389  3              HARD_ERROR ();
;    4390  3              IF .RP_ADDR [ENDCOD] EQLU (OP_ELP + OP_END) OR  ! IF ENDCODE IS EXECUTE LOCAL PROGRAM
;    4391  4                 .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END)     !                 OR GET DUST STATUS
;    4392  4              THEN    BEGIN
;    4393  4                      CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;             ! TURN OFF DUP EXERCISR
;    4394  3                      END;
;    4395  3              END
;    4396  2      ELSE                                                    ! ELSE - I/O WAS SUCCESSFUL
;    4397  3          BEGIN
;    4398  3
;    4399  4          IF .RP_ADDR [ENDCOD] EQLU (OP_GDS + OP_END) ! IF ENDCODE IS GET DUST STATUS
;    4400  3          THEN
;    4401  4              BEGIN
;    4402  4              IF .RP_ADDR [9,11,1,0] EQL 1
;    4403  4                      THEN CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = ACTIVE      ! CONTROLLER IN AN ACTIVE STAE
;    4404  4                      ELSE CST_ADDR [.DUOFF + OF_DBN, D_ACTIVE] = IDLE;       ! CONTROLLER IN AN IDLE STATE
;    4405  4              IF .RP_ADDR [9,9,1,0] NEQ 1 THEN        ! TEST TO SEE IF CONTROLLER LOCAL PROGRAMS(PG 18 OF DUP DOC)
;    4406  5                      BEGIN
;    4407  5                      HARD_ERROR ();
;    4408  5                      CST_ADDR [.DUOFF + OF_DBN, NODUPMEDIA] = 1;             ! TURN OFF DUP EXERCISR
;    4409  4                      END;
;    4410  3              END;
```

```
;    4411  3
;    4412  3
;    4413  3     IF (.RP_ADDR [ENDCOD] EQL (OP_RCO + OP_END)) AND
;    4414  3        (.DUPPKT [DUPTYPE] EQL 6) AND
;    4415  3        (.DUPPKT [DUPMESG] EQL 2) AND                    !IF IT IS A RECEIVE DBN COMMAND WITH TYPE 6 AND MESSAGE 2 THEN
;    4416  4        (.CST_ADDR [.DUOFF + OF_DBN, DUPWRITE] EQLU 1)       ! IF WRITE FLAG SET IN CST TABLE THEN COMPARE BLOCKS
;    4417  3             THEN DUP_COMPARE ();
;    4418  3
;    4419  2     END;                                                 ! COMPARE THE FOLLOWING 512 BYTES
;    4420  2
;    4421  2     PUT_RETPKT (.RP_INDX);
;    4422  1     END;                                                 ! ROUTINE DIO_RETPKT

                                        .SBTTL  DIO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146                    DIO.RETPKT::
                                        MOV     R1,-(SP)                        ;                                       4359
000002  112700  000001                  MOVB    #1,R0                           ; *,FLAG                                4378
000006  013701  000000G                 MOV     RP_ADDR,R1                      ;                                       4385
000012  005761  000016                  TST     16(R1)
000016  001435                          BEQ     2$
000020  013700  001156'                 MOV     DUOFF,R0                        ;                                       4388
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                   ;                                       4389
000044  013700  000000G                 MOV     RP_ADDR,R0                      ;                                       4390
000050  126027  000014  000203          CMPB    14(R0),#203
000056  001404                          BEQ     1$
000060  126027  000014  000201          CMPB    14(R0),#201                     ;                                       4391
000066  001112                          BNE     6$
000070  013700  001156'          1$:     MOV     DUOFF,R0                        ;                                       4393
000074  006300                          ASL     R0
000076  063700  000000G                 ADD     CST_ADDR,R0
000102  052760  100000  000020          BIS     #100000,20(R0)
000110  000501                          BR      6$                              ;                                       4385
000112  126127  000014  000201  2$:     CMPB    14(R1),#201                     ;                                       4399
000120  001036                          BNE     5$
000122  013700  001156'                 MOV     DUOFF,R0                        ;                                       4403
000126  006300                          ASL     R0
000130  063700  000000G                 ADD     CST_ADDR,R0
000134  032761  004000  000022          BIT     #4000,22(R1)                    ;                                       4402
000142  001404                          BEQ     3$
000144  052760  020000  000020          BIS     #20000,20(R0)                   ;                                       4403
000152  000403                          BR      4$                              ;                                       4402
000154  042760  020000  000020  3$:     BIC     #20000,20(R0)                   ;                                       4404
000162  032761  001000  000022  4$:     BIT     #1000,22(R1)                    ;                                       4405
000170  001012                          BNE     5$
000172  004737  000000V                 JSR     PC,HARD_ERROR                   ;                                       4407
000176  013700  001156'                 MOV     DUOFF,R0                        ;                                       4408
000202  006300                          ASL     R0
000204  063700  000000G                 ADD     CST_ADDR,R0
000210  052760  100000  000020          BIS     #100000,20(R0)
```

```
000216  013700  000000G                   5$:     MOV     RP.ADDR,R0                      ;                       4413
000222  126027  000014  000205                    CMPB    14(R0),#205
000230  001031                                    BNE     6$
000232  013700  000000G                           MOV     DUPPKT,R0                       ;                       4414
000236  042700  007777                            BIC     #7777,R0
000242  020027  060000                            CMP     R0,#60000
000246  001022                                    BNE     6$
000250  013700  000000G                           MOV     DUPPKT,R0                       ;                       4415
000254  042700  170000                            BIC     #170000,R0
000260  020027  000002                            CMP     R0,#2
000264  001013                                    BNE     6$
000266  013700  001156'                           MOV     DUOFF,R0                        ;                       4416
000272  006300                                    ASL     R0
000274  063700  000000G                           ADD     CST.ADDR,R0
000300  032760  010000  000020                    BIT     #10000,20(R0)
000306  001402                                    BEQ     6$
000310  004737  000000V                           JSR     PC,DUP.COMPARE                  ;                       4417
000314  013746  000000G                   6$:     MOV     RP.INDX,-(SP)                   ;                       4421
000320  004737  000000G                           JSR     PC,PUT.RETPKT
000324  005726                                    TST     (SP)+                           ;                       4378
000326  012601                                    MOV     (SP)+,R1                        ;                       4359
000330  000207                                    RTS     PC

; Routine Size:  109 words,    Routine Base:  $CODE$ + 20134
; Maximum stack depth per invocation:  3 words


;   4423  1
```

```
;  4424  1     GLOBAL ROUTINE DUP_COMPARE : NOVALUE =
;  4425  1
;  4426  1
;  4427  1     !+
;  4428  1     !       THIS ROUTINE IS CALLED BY DIO_RETPKT WHEN THE RECEIVE DATA COMMAND
;  4429  1     !       IS BEING PROCESSED. THIS COMMAND COMPARES THE WRITTEN BUFFER WITH
;  4430  1     !       THE PATERN WORD GIVEN IN SEND DATA COMMAND.  FOR EVERY WORD COMPARED
;  4431  1     !       THE ROUTINE INCREMENTS THE TALLY TABLE.  IF THE COMPARE SHOWS AN
;  4432  1     !       ERROR, THE DBN HARD ERROR COUNTER WILL BE INCREMENTED AND THE
;  4433  1     !       THE DBN NUMBER AND BYTE COUNT WILL BE PRINTED.
;  4434  1     !
;  4435  1     !       IMPLICIT INPUTS:
;  4436  1     !               S_PATTERN       ! THE SAVED PATTERN WRITTEN TO THE DBN'S
;  4437  1     !               S_DUPPKT        ! THE POINTER FOR DUP BUFFER
;  4438  1     !               T_ADDR          ! THE ADDRESS OF THE TALLY TABLE FOR THIS UNIT
;  4439  1     !               CST_ADDR        ! THE ADDRESS OF PRESENT CONTROLLER STATUS TABLE
;  4440  1     !-
;  4441  2     BEGIN
;  4442  2
;  4443  2     OWN
;  4444  2             COUNT : WORD;
;  4445  2
;  4446  2     !PRINTX (DER19);
;  4447  2     S_DUPPKT = 0;
;  4448  2     INCR COUNT FROM 1 TO 256 DO     !INDEX PIONTER FOR DATA STORED IN MSCP ENV PACKET
;  4449  3             BEGIN
;  4450  3             S_DUPPKT = .S_DUPPKT + 2;                               ! INITIALLY THIS SKIPS THE FIRST WORD OF DUPPKT
;  4451  3             IF .(DUPPKT + .S_DUPPKT) NEQ .S_PATTERN THEN            !IF THE CONTENTS OF DBN DOESN'T EQUAL PATTERN
;  4452  4                     BEGIN
;  4453  4                     CST_ADDR [.DUOFF + OF_DBN, DUPERROR] = 1;               ! SET DUP ERROR FLAG
;  4454  4                     ERRHRD (46, EH_10, EMS_22);                     !LIST ERROR
;  4455  4                     EXITLOOP;
;  4456  3                     END;
;  4457  2             END;                                                    !GO THROUGH ALL DBN WORDS
;  4458  1     END;                                                            !END ROUTINE DUP-COMPARE


001206                                  .PSECT  $GGG$,  RO
001206                          COUNT:  .BLKW   1


                                        .SBTTL  DUP.COMPARE MULTI-DRIVE TEST ROUTINES
020466                                  .PSECT  $CODE$,  RO

000000  010146                  DUP.COMPARE::
                                        MOV     R1,-(SP)                ;                               4424
000002  005037  000000G                 CLR     S.DUPPKT                ;                               4447
000006  012701  000400                  MOV     #400,R1                 ; #,COUNT                       4448
000012  062737  000002  000000G 1$:     ADD     #2,S.DUPPKT             ;                               4450
000020  013700  000000G                 MOV     S.DUPPKT,R0             ;                               4451
000024  026037  000000G 000000G         CMP     DUPPKT(R0),S.PATTERN
000032  001415                          BEQ     2$
```

```
000034  013700  001156'                        MOV     DUOFF,R0                        ;                       4453
000040  006300                                  ASL     R0
000042  063700  000000G                        ADD     CST.ADDR,R0
000046  052760  040000  000020                 BIS     #40000,20(R0)
000054  104456                                  TRAP    56                              ;                       4454
000056  000056                                  .WORD   56
000060  000000G                                 .WORD   EH.10
000062  000000G                                 .WORD   EMS.22
000064  000402                                  BR      3$                              ;                       4455
000066  005301                          2$:     DEC     R1                              ; COUNT                 4448
000070  001350                                  BNE     1$
000072  012601                          3$:     MOV     (SP)+,R1                        ;                       4424
000074  000207                                  RTS     PC

; Routine Size:  31 words,    Routine Base:  $CODE$ + 20466
; Maximum stack depth per invocation:  3 words


;    4459  1
;    4460  1
;    4461  1
```

```
;    4462  1     GLOBAL routine IO_RETPKT : novalue =
;    4463  1
;    4464  1     !+
;    4465  1     !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;    4466  1     !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;    4467  1     !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;    4468  1     !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;    4469  1     !
;    4470  1     !       IMPLICIT INPUTS:
;    4471  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    4472  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    4473  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4474  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    4475  1     !               L$LUN - CURRENT UNIT NUMBER
;    4476  1     !-
;    4477  1
;    4478  2         begin
;    4479  2
;    4480  2         local
;    4481  2             FLAG : byte initial (byte (TRUE));
;    4482  2
;    4483  2         FSET_UPAR ();                           ! FIND UNIT'S ENTRY IN CST AND SET UP UNIT-RELATED DATA
;    4484  2         ST_CODE = .RP_ADDR [STSCOD];            ! GET STATUS CODE FROM RETPKT
;    4485  2         SB_CODE = .RP_ADDR [SUBCOD];            ! GET SUB CODE, IF ANY
;    4486  2
;    4487  3         if (.ST_CODE neq ST_SUC)                ! IF STATUS CODE INDICATES ERROR
;    4488  2         then
;    4489  3             begin
;    4490  3             HARD_ERROR ();                      ! UPDATE ERROR COUNT
;    4491  3             COMPARE_DATA = FALSE;               ! NO POINT IN DOING HOST COMPARES ON ERRORS
;    4492  3
;    4493  3             if (.ST_CODE neq ST_OFL) and        ! DROP UNIT IF ERROR COUNTS EXCEEDS LIMIT
;    4494  3                (.ST_CODE neq ST_AVL) and
;    4495  4                (.T_ADDR [ERR_HARD] gequ .SWP_ERROR)
;    4496  3             then
;    4497  4                 begin
;    4498  4                 DUR [.L$LUN] = DU_HERR;         ! LOAD REASON FOR DROPPING UNIT
;    4499  4                 DODU (.L$LUN);                  ! DROP UNIT
;    4500  3                 end;
;    4501  3
;    4502  3             end
;    4503  2         else                                    ! IF I/O WAS SUCCESSFUL
;    4504  3             begin
;    4505  3             UPD_IO_TALLY ();                    ! UPDATE I/O TALLY (STATISTICS)
;    4506  3
;    4507  4             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)
;    4508  3             then
;    4509  3                 COMPARE_DATA = TRUE;            ! HOST COMPARES MAY BE ALLOWED IF NO FURTHER ERRORS
;    4510  3
;    4511  5             if (BIT_TST (SWP_FLAGS, SWP_HWC)) and   ! IF HOST IS DOING WRITE COMPARES
;    4512  4                (.COMPARE_DATA)
;    4513  5             then
;    4514  3                 FLAG = HOST_WRT_CHK ();         ! SAVE I/O PACKET OR DO WRITE-CHECK
```

```
;    4515  3
;    4516  2              end;
;    4517  2
;    4518  2          IF .FLAG                        ! IF FLAG IS STILL  TRUE
;    4519  2          then
;    4520  2              SWEEP ();                   ! DEALLOCATE BUFFER(S) AND RETPKT(S)
;    4521  2
;    4522  2          QIO [.CCTLR] = .QIO [.CCTLR] - 1;   ! DECREMENT NO. OF OUTSTANDING QIOs
;    4523  1          end;                            ! ROUTINE IO.RETPKT

                                        .SBTTL  IO.RETPKT MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          IO.RETPKT::
                                        JSR     R1,$SAVE2                       ;                       4462
000004  112701  000001                  MOVB    #1,R1                           ; #,FLAG                4478
000010  004737  000000V                  JSR     PC,FSET.UPAR                    ;                       4483
000014  013700  000000G                  MOV     RP.ADDR,R0                      ;                       4484
000020  116037  000016  000000G         MOVB    16(R0),ST.CODE
000026  042737  177740  000000G         BIC     #177740,ST.CODE
000034  016002  000016                  MOV     16(R0),R2                       ;                       4485
000040  006202                          ASR     R2
000042  006202                          ASR     R2
000044  006202                          ASR     R2
000046  006202                          ASR     R2
000050  006202                          ASR     R2
000052  042702  174000                  BIC     #174000,R2
000056  010237  000000G                  MOV     R2,SB.CODE
000062  005737  000000G                  TST     ST.CODE                         ;                       4487
000066  001431                          BEQ     1$
000070  004737  000000V                  JSR     PC,HARD.ERROR                   ;                       4490
000074  105037  001170'                  CLRB    COMPARE.DATA                    ;                       4491
000100  023727  000000G  000003         CMP     ST.CODE,#3                      ;                       4493
000106  001447                          BEQ     3$
000110  023727  000000G  000004         CMP     ST.CODE,#4                      ;                       4494
000116  001443                          BEQ     3$
000120  013700  000000G                  MOV     T.ADDR,R0                       ;                       4495
000124  026037  000014  000000G         CMP     14(R0),SWP.ERROR
000132  103435                          BLO     3$
000134  013700  000000G                  MOV     T$LUN,R0                        ;                       4498
000140  112760  000004  000000G         MOVB    #4,DUR(R0)
000146  104451                          TRAP    51                              ;                       4499
000150  000426                          BR      3$                              ;                       4487
000152  004737  000000V          1$:     JSR     PC,UPD.IO.TALLY                 ;                       4505
000156  013700  000000G                  MOV     RP.ADDR,R0                      ;                       4507
000162  126027  000014  000242         CMPB    14(R0),#242
000170  001003                          BNE     2$
000172  112737  000001  001170'         MOVB    #1,COMPARE.DATA                 ;                       4509
000200  032737  000040  000000G  2$:     BIT     #40,SWP.FLAGS                   ;                       4511
000206  001407                          BEQ     3$
000210  032737  000001  001170'         BIT     #1,COMPARE.DATA                 ;                       4512
000216  001403                          BEQ     3$
000220  004737  000000V                  JSR     PC,HOST.WRT.CHK                 ;                       4514
000224  110001                          MOVB    R0,R1                           ; R,FLAG
```

```
000226  006001                          3$:     ROR     R1                      ; FLAG                  4518
000230  103002                                  BCC     4$
000232  004737  000000V                        JSR     PC,SWEEP                ;                       4520
000236  013700  000000G                4$:     MOV     CCTLR,R0                ;                       4522
000242  105360  000000G                        DECB    QIO(R0)
000246  000207                                RTS     PC                      ;                       4462

; Routine Size:  84 words,      Routine Base:  $CODE$ + 20564
; Maximum stack depth per invocation:  5 words
```

```
;    4524  1     GLOBAL routine FSET_UPAR : novalue =
;    4525  1
;    4526  1     !+
;    4527  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;    4528  1     !       CONTROLLER STATUS TABLE (CST) FOR THE DISK ADDRESS WHICH IS
;    4529  1     !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;    4530  1     !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;    4531  1     !       DATA PARAMETERS.
;    4532  1     !
;    4533  1     !       IMPLICIT INPUTS:
;    4534  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    4535  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;    4536  1     !-
;    4537  1
;    4538  2         begin
;    4539  2
;    4540  2         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do  ! FOR EACH UNIT
;    4541  2
;    4542  2             if .CST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .RP_ADDR [DISK]        ! IF RETPKT UNIT MATCHES CST ENTRY
;    4543  2             then
;    4544  3                 beg
;    4545  3                 SET_UPAR (.OFFSET);                                             ! SET UP UNIT-RELATED DATA
;    4546  3                 return;                                                         ! DONE
;    4547  2                 end;
;    4548  2
;    4549  2         PRINTF (DBM19, .RP_ADDR [DISK], .CCTLR);                                ! "CAN'T FIND DISK XXX IN CST X"
;    4550  1         end;                                                                    ! ROUTINE FSET_UPAR
```

```
                        .SBTTL  FSET.UPAR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G   FSET.UPAR::
                        JSR     R1,$SAVE4                       ;                       4524
000004  012702  000003    MOV     #3,R2                           ; #,OFFSET              4540
000010  010201            1$:   MOV     R2,R1                   ; OFFSET,#              4542
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,#              4545
000044  004737  000000G   JSR     PC,SET.UPAR
000050  005726            TST     (SP)+                           ;                       4546
000052  000207            RTS     PC                              ;                       4544
000054  062702  000012    2$:   ADD     #12,R2                  ; #,OFFSET              4540
000060  020227  000041    CMP     R2,#41                          ; OFFSET,#
000064  003751            BLE     1$
000066  013746  000000G   MOV     CCTLR,-(SP)                     ;                       4549
000072  013700  000000G   MOV     RP.ADDR,R0
000076  016046  000010    MOV     10(R0),-(SP)
000102  012746  000000G   MOV     #DBM19,-(SP)
```

```
000106  012746  000003                    MOV     #3,-(SP)
000112  010600                            MOV     SP,R0                   ; SP,R
000114  104417                            TRAP    17
000116  062706  000010                    ADD     #10,SP                  ;                       4538
000122  000207                            RTS     PC                      ;                       4524

; Routine Size:  42 words,      Routine Base:  $CODE$ + 21034
; Maximum stack depth per invocation:  11 words
```

```
;  4551  1     GLOBAL routine HARD_ERROR : novalue =
;  4552  1
;  4553  1     !+
;  4554  1     !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO INCREMENT THE HARD
;  4555  1     !       ERROR STATISTIC FIELD FOR THE CURRENT UNIT.  IF THE HARD ERROR COUNT
;  4556  1     !       HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS DROPPED
;  4557  1     !       FROM TESTING.
;  4558  1     !
;  4559  1     !       IMPLICIT INPUTS:
;  4560  1     !               L$LUN - CURRENT UNIT NUMBER
;  4561  1     !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;  4562  1     !               CUOFF - CST OFFSET FOR CURRENT UNIT
;  4563  1     !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;  4564  1     !-
;  4565  1
;  4566  2         begin
;  4567  2         T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;                     ! INCREMENT UNIT'S HARD ERROR COUNT
;  4568  2
;  4569  2         selectoneu .ST_CODE of
;  4570  2             set
;  4571  2
;  4572  2             [ST_SUC]:       if .SB_CODE neq 0                           ! SUCCESS WITH NON-ZERO SUB-CODE
;  4573  2                             then
;  4574  3                                 begin
;  4575  3
;  4576  3                                 if .SB_CODE eql 4
;  4577  3                                 then
;  4578  3                                     T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1
;  4579  3                                 else
;  4580  3                                     T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  4581  3
;  4582  3                                 if .APT_MODE
;  4583  3                                 then
;  4584  3                                     ERR_HRD_RTNE_APT (44)
;  4585  3                                 else
;  4586  3                                     ERR_HRD_RTNE (44);
;  4587  3
;  4588  2                                 end;
;  4589  2
;  4590  3                 [ST_CMD]:       begin                                   ! INVALID COMMAND
;  4591  3                                 T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  4592  3
;  4593  3                                 if .APT_MODE
;  4594  3                                 then
;  4595  3                                     ERR_HRD_RTNE_APT (31)
;  4596  3                                 else
;  4597  3                                     ERR_HRD_RTNE (31);
;  4598  3
;  4599  2                                 end;
;  4600  2
;  4601  3                 [ST_ABO]:       begin                                   ! COMMMAND ABORTED
;  4602  3                                 T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4603  3
```

```
;  4604  3                        if .APT_MODE
;  4605  3                        then
;  4606  3                            ERR_HRD_RTNE_APT (32)
;  4607  3                        else
;  4608  3                            ERR_HRD_RTNE (32);
;  4609  3
;  4610  2                        end;
;  4611  2
;  4612  3            [ST_OFL] :  begin                                   ! OFFLINE
;  4613  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4614  3
;  4615  3                        if .APT_MODE
;  4616  3                        then
;  4617  4                            begin
;  4618  4                            .MAIL_BOX_TESTNUM = 1;
;  4619  4                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  4620  3                            end;
;  4621  3
;  4622  3                        ERRDF (18, EGD_18, EMS_18);             !
;  4623  3                        DUR [.L$LUN] = DU_DFATAL;               ! DEVICE FATAL ERROR
;  4624  3                        DODU (.L$LUN);                          ! DROP UNIT
;  4625  2                        end;
;  4626  2
;  4627  3            [ST_AVL]:   begin                                   ! WENT TO AVAILABLE STATE
;  4628  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4629  3
;  4630  3                        if .APT_MODE
;  4631  3                        then
;  4632  4                            begin
;  4633  4                            .MAIL_BOX_TESTNUM = 1;
;  4634  4                            .MAIL_BOX_SUBTST = .CST_ADDR [.CUOFF + OF_DATA, D_DISK_NUM];
;  4635  3                            end;
;  4636  3
;  4637  3                        ERRDF (24, EGD_18, EMS_24);             !
;  4638  3                        DUR [.L$LUN] = DU_DFATAL;               ! DEVICE FATAL ERROR
;  4639  3                        DODU (.L$LUN);                          ! DROP UNIT
;  4640  2                        end;
;  4641  2
;  4642  3            [ST_MFE]:   begin                                   ! MEDIA FORMAT ERROR
;  4643  3                        T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1;
;  4644  3
;  4645  3                        if .APT_MODE
;  4646  3                        then
;  4647  3                            ERR_HRD_RTNE_APT (35)
;  4648  3                        else
;  4649  3                            ERR_HRD_RTNE (35);
;  4650  3
;  4651  2                        end;
;  4652  2
;  4653  3            [ST_WPT]:   begin                                   ! DEVICE WRITE PROTECTED
;  4654  3
;  4655  3                        if .SB_CODE eql 128
;  4656  3                        then
```

```
;  4657  3                        T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1
;  4658  3                    else
;  4659  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4660  3
;  4661  3                    if .APT_MODE
;  4662  3                    then
;  4663  3                        ERR_HRD_RTNE_APT (36)
;  4664  3                    else
;  4665  3                        ERR_HRD_RTNE (36);
;  4666  3
;  4667  2                    end;
;  4668  2
;  4669  3            [ST_CMP]:       begin                                   ! COMPARE ERROR
;  4670  3                    T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  4671  3
;  4672  3                    if .APT_MODE
;  4673  3                    then
;  4674  3                        ERR_HRD_RTNE_APT (37)
;  4675  3                    else
;  4676  3                        ERR_HRD_RTNE (37);
;  4677  3
;  4678  2                    end;
;  4679  2
;  4680  3            [ST_DAT]:       begin                                   ! DATA ERROR
;  4681  3
;  4682  3                    if .SB_CODE eql 2
;  4683  3                    then
;  4684  3                        T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  4685  3                    else
;  4686  3                        T_ADDR [ERR_HRD_DAT] = .T_ADDR [ERR_HRD_DAT] + 1;
;  4687  3
;  4688  3                    if (.SB_CODE eql 0) and
;  4689  3                       (not .FORCED_ERROR) and
;  4690  4                       (BIT_TST (SWP_FLAGS, SWF_FER))
;  4691  3                    then
;  4692  4                        begin
;  4693  4                        FORCED_ERROR = TRUE;                        ! BLOCK WITH "FORCED ERROR" FOUND
;  4694  4                        FER_LBN = .RP_ADDR [LBN_LO];                !
;  4695  4                        FER_BC = .RP_ADDR [CBCNT_LO];               !
;  4696  3                        end;
;  4697  3
;  4698  3                    if .APT_MODE
;  4699  3                    then
;  4700  3                        ERR_HRD_RTNE_APT (38)
;  4701  3                    else
;  4702  3                        ERR_HRD_RTNE (38);
;  4703  3
;  4704  2                    end;
;  4705  2
;  4706  3            [ST_HST]:       begin                                   ! HOST ACCESS ERROR
;  4707  3                    T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;  4708  3
;  4709  3                    if .APT_MODE
```

```
;  4710  3                        then
;  4711  3                            ERR_HRD_RTNE_APT (39)
;  4712  3                        else
;  4713  3                            ERR_HRD_RTNE (39);
;  4714  3
;  4715  2                        end;
;  4716  2
;  4717  3            [ST_CNT]:   begin                                   ! CONTROLLER ERROR
;  4718  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4719  3
;  4720  3                        if .APT_MODE
;  4721  3                        then
;  4722  3                            ERR_HRD_RTNE_APT (40)
;  4723  3                        else
;  4724  3                            ERR_HRD_RTNE (40);
;  4725  3
;  4726  2                        end;
;  4727  2
;  4728  3            [ST_DRV]:   begin                                   ! DRIVE ERROR
;  4729  3
;  4730  3                        if .SB_CODE eql 3
;  4731  3                        then
;  4732  3                            T_ADDR [ERR_HRD_SEK] = .T_ADDR [ERR_HRD_SEK] + 1
;  4733  3                        else
;  4734  3                            T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4735  3
;  4736  3                        if .APT_MODE
;  4737  3                        then
;  4738  3                            ERR_HRD_RTNE_APT (41)
;  4739  3                        else
;  4740  3                            ERR_HRD_RTNE (41);
;  4741  3
;  4742  2                        end;
;  4743  2
;  4744  3            [ST_DIA]:   begin                                   ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  4745  3                        T_ADDR [ERR_HRD_DRV] = .T_ADDR [ERR_HRD_DRV] + 1;
;  4746  3
;  4747  3                        if .APT_MODE
;  4748  3                        then
;  4749  3                            ERR_HRD_RTNE_APT (43)
;  4750  3                        else
;  4751  3                            ERR_HRD_RTNE (43);
;  4752  3
;  4753  2                        end;
;  4754  2
;  4755  3            [otherwise]: begin                                  ! PRINT STATUS CODE IF NO MATCH
;  4756  3                        C_ERR_TBL [.CCTLR, C_ERR_HRD] = .C_ERR_TBL [.CCTLR, C_ERR_HRD] + 1;
;  4757  3
;  4758  3                        if .APT_MODE
;  4759  3                        then
;  4760  3                            ERR_HRD_RTNE_APT (45)
;  4761  3                        else
;  4762  3                            ERR_HRD_RTNE (45);
```

```
;   4763  3
;   4764  2                         end;
;   4765  2
;   4766  2              tes;
;   4767  2
;   4768  1          end;                                        ! ROUTINE HARD_ERROR

                                    .SBTTL  HARD.ERROR MULTI-DRIVE TEST ROUTINES
000000  004137  000000G             HARD.ERROR::
                                    JSR     R1,$SAVE3                   ;                   4551
000004  013701  000000G             MOV     T.ADDR,R1                   ;                   4567
000010  005261  000014              INC     14(R1)
000014  013702  000000G             MOV     ST.CODE,R2                  ;                   4569
000020  001030                      BNE     5$                          ;                   4572
000022  013703  000000G             MOV     SB.CODE,R3
000026  001001                      BNE     1$
000030  000207                      RTS     PC
000032  012700  000050              1$:     MOV     #50,R0              ;                   4578
000036  060100                      ADD     R1,R0
000040  020327  000004              CMP     R3,#4                       ;                   4576
000044  001002                      BNE     2$
000046  105210                      INCB    (R0)                        ;                   4578
000050  000402                      BR      3$                          ;                   4576
000052  105260  000001              2$:     INCB    1(R0)               ;                   4580
000056  032737  000001  001162'     3$:     BIT     #1,APT.MODE         ;                   4582
000064  001403                      BEQ     4$
000066  012746  000054              MOV     #54,-(SP)                   ;                   4584
000072  000556                      BR      14$
000074  012746  000054              4$:     MOV     #54,-(SP)           ;                   4586
000100  000556                      BR      16$
000102  020227  000001              5$:     CMP     R2,#1               ;                   4590
000106  001014                      BNE     7$
000110  105261  000051              INCB    51(R1)                      ;                   4591
000114  032737  000001  001162'     BIT     #1,APT.MODE                 ;                   4593
000122  001403                      BEQ     6$
000124  012746  000037              MOV     #37,-(SP)                   ;                   4595
000130  000567                      BR      20$
000132  012746  000037              6$:     MOV     #37,-(SP)           ;                   4597
000136  000567                      BR      22$
000140  020227  000002              7$:     CMP     R2,#2               ;                   4601
000144  001014                      BNE     9$
000146  105261  000050              INCB    50(R1)                      ;                   4602
000152  032737  000001  001162'     BIT     #1,APT.MODE                 ;                   4604
000160  001403                      BEQ     8$
000162  012746  000040              MOV     #40,-(SP)                   ;                   4606
000166  000567                      BR      24$
000170  012746  000040              8$:     MOV     #40,(SP)            ;                   4608
000174  000567                      BR      26$
000176  020227  000003              9$:     CMP     R2,#3               ;                   4610
000202  001036                      BNE     11$
000204  105261  000050              INCB    50(R1)                      ;                   4613
000210  032737  000001  001162'     BIT     #1,APT.MODE                 ;                   4615
```

```
000216  001415                                  BEQ     10$
000220  012777  000001  001164'                MOV     #1,@MAIL.BOX.TESTNUM            ;                4618
000226  013700  000000G                         MOV     CUOFF,R0                        ;                4619
000232  006300                                  ASL     R0
000234  063700  000000G                         ADD     CST.ADDR,R0
000240  111077  001166'                         MOVB    (R0),@MAIL.BOX.SUBTST
000244  042777  177760  001166'                BIC     #177760,@MAIL.BOX.SUBTST
000252  104455                          10$:    TRAP    55                              ;                4622
000254  000022                                  .WORD   22
000256  000000G                                 .WORD   EGD.18
000260  000000G                                 .WORD   EMS.18
000262  013700  000000G                         MOV     L$LUN,R0                        ;                4623
000266  112760  000005  000000G                MOVB    #5,DUR(R0)
000274  104451                                  TRAP    51                              ;                4624
000276  000207                                  RTS     PC                              ;                4569
000300  020227  000004                  11$:    CMP     R2,#4                           ;                4627
000304  001036                                  BNE     13$
000306  105261  000050                          INCB    50(R1)                          ;                4628
000312  032737  000001  001162'                BIT     #1,APT.MODE                     ;                4630
000320  001415                                  BEQ     12$
000322  012777  000001  001164'                MOV     #1,@MAIL.BOX.TESTNUM            ;                4633
000330  013700  000000G                         MOV     CUOFF,R0                        ;                4634
000334  006300                                  ASL     R0
000336  063700  000000G                         ADD     CST.ADDR,R0
000342  111077  001166'                         MOVB    (R0),@MAIL.BOX.SUBTST
000346  042777  177760  001166'                BIC     #177760,@MAIL.BOX.SUBTST
000354  104455                          12$:    TRAP    55                              ;                4637
000356  000030                                  .WORD   30
000360  000000G                                 .WORD   EGD.18
000362  000000G                                 .WORD   EMS.24
000364  013700  000000G                         MOV     L$LUN,R0                        ;                4638
000370  112760  000005  000000G                MOVB    #5,DUR(R0)
000376  104451                                  TRAP    51                              ;                4639
000400  000207                                  RTS     PC                              ;                4569
000402  020227  000005                  13$:    CMP     R2,#5                           ;                4642
000406  001014                                  BNE     17$
000410  105261  000046                          INCB    46(R1)                          ;                4643
000414  032737  000001  001162'                BIT     #1,APT.MODE                     ;                4645
000422  001403                                  BEQ     15$
000424  012746  000043                          MOV     #43,-(SP)                       ;                4647
000430  000562                          14$:    BR      35$
000432  012746  000043                  15$:    MOV     #43,-(SP)                       ;                4649
000436  000562                          16$:    BR      37$
000440  020227  000006                  17$:    CMP     R2,#6                           ;                4653
000444  001025                                  BNE     23$
000446  012700  000050                          MOV     #50,R0                          ;                4657
000452  060100                                  ADD     R1,R0
000454  023727  000000G 000200                  CMP     SB.CODE,#200                    ;                4655
000462  001003                                  BNE     18$
000464  105260  000001                          INCB    1(R0)                           ;                4657
000470  000401                                  BR      19$                             ;                4655
000472  105210                          15$:    INCB    (R0)                            ;                4659
000474  032737  000001  001162'          19$:    BIT     #1,APT.MODE                     ;                4661
```

```
000502  001403                                          BEQ     21$
000504  012746  000044                                  MOV     #44,-(SP)       ;                               4663
000510  000577                              20$:        BR      43$
000512  012746  000044                      21$:        MOV     #44,-(SP)       ;                               4665
000516  000577                              22$:        BR      45$
000520  020227  000007                      23$:        CMP     R2,#7           ;                               4669
000524  001014                                          BNE     27$
000526  105261  000047                                  INCB    47(R1)          ;                               4670
000532  032737  000001  001162'                         BIT     #1,APT.MODE     ;                               4672
000540  001403                                          BEQ     25$
000542  012746  000045                                  MOV     #45,-(SP)       ;                               4674
000546  000577                              24$:        BR      47$
000550  012746  000045                      25$:        MOV     #45,-(SP)       ;                               4676
000554  000560                              26$:        BR      45$
000556  020227  000010                      27$:        CMP     R2,#10          ;                               4680
000562  001053                                          BNE     32$
000564  012700  000046                                  MOV     #46,R0          ;                               4684
000570  060100                                          ADD     R1,R0
000572  023727  000000G 000002                          CMP     SB.CODE,#2      ;                               4682
000600  001002                                          BNE     28$
000602  105210                                          INCB    (R0)            ;                               4684
000604  000402                                          BR      29$             ;                               4682
000606  105260  000001                      28$:        INCB    1(R0)           ;                               4686
000612  005737  000000G                     29$:        TST     SB.CODE         ;                               4688
000616  001023                                          BNE     30$
000620  132737  0.0001  000000G                         BITB    #1,FORCED.ERROR ;                               4689
000626  001017                                          BNE     30$
000630  032737  004000  000000G                         BIT     #4000,SWP.FLAGS ;                               4690
000636  001413                                          BEQ     30$
000640  112737  000001  000000G                         MOVB    #1,FORCED.ERROR ;                               4693
000646  013700  000000G                               MOV     RP.ADDR,R0      ;                               4694
000652  016037  000050  000000G                         MOV     50(R0),FER.LBN
000660  016037  000044  000000G                         MOV     44(R0),FER.BC   ;                               4695
000666  032737  000001  001162'              30$:        BIT     #1,APT.MODE     ;                               4698
000674  001403                                          BEQ     31$
000676  012746  000046                                  MOV     #46,-(SP)       ;                               4700
000702  000521                                          BR      47$
000704  012746  000046                      31$:        MOV     #46,-(SP)       ;                               4702
000710  000523                                          BR      49$
000712  020227  000011                      32$:        CMP     R2,#11          ;                               4706
000716  001014                                          BNE     34$
000720  105261  000051                                  INCB    51(R1)          ;                               4707
000724  032737  000001  001162'                         BIT     #1,APT.MODE     ;                               4709
000732  001403                                          BEQ     33$
000734  012746  000047                                  MOV     #47,-(SP)       ;                               4711
000740  000502                                          BR      47$
000742  012746  000047                      33$:        MOV     #47,-(SP)       ;                               4713
000746  000504                                          BR      49$
000750  020227  000012                      34$:        CMP     R2,#12          ;                               4717
000754  001014                                          BNE     38$
000756  105261  000050                                  INCB    50(R1)          ;                               4718
000762  032737  000001  001162'                         BIT     #1,APT.MODE     ;                               4720
000770  001403                                          BEQ     36$
```

```
000772  012746  000050                        MOV     #50, (SP)                       ;                       47??
000776  000463                          35$:    BR      47$
001000  012746  000050                  36$:    MOV     #50, (SP)                       ;                       47??
001004  000465                          57$:    BR      49$
001006  020227  000013                  38$:    CMP     R2,#13                          ;                       47??
001012  001025                                  BNE     42$
001014  023727  000000G 000003                  CMP     SB.CODE,#3                      ;                       47??
001022  001003                                  BNE     39$
001024  105261  000046                          INCB    46(R1)                          ;                       47??
001050  000402                                  BR      40$                             ;                       47??
001052  105261  000050                  39$:    INCB    50(R1)                          ;                       47??
001056  032737  000001  001162'        40$:    BIT     #1,APT.MODE                     ;                       47??
001064  001403                                  BEQ     41$
001066  012746  000051                          MOV     #51, (SP)                       ;                       47??
001072  000435                                  BR      47$
001074  012746  000051                  41$:    MOV     #51, (SP)                       ;                       47??
001100  000437                                  BR      49$
001102  020227  000037                  42$:    CMP     R2,#37                          ;                       47??
001106  001014                                  BNE     46$
001110  105261  000050                          INCB    50(R1)                          ;                       47??
001114  032737  000001  001162'                BIT     #1,APT.MODE                     ;                       47??
001122  001403                                  BEQ     44$
001124  012746  000053                          MOV     #53, (SP)                       ;                       47??
001130  000416                                  BR      47$
001132  012746  000053                  44$:    MOV     #53, (SP)                       ;                       47??
001136  000420                          45$:    BR      49$
001140  013700  000000G                 46$:    MOV     CCTLR,R0                        ;                       47??
001144  006300                                  ASL     R0
001146  105260  000000G                          INCB    C.ERR.TBL(R0)
001152  032737  000001  001162'                BIT     #1,APT.MODE                     ;                       47??
001160  001405                                  BEQ     48$
001162  012746  000055                          MOV     #55, (SP)                       ;                       47??
001166  004737  000000V                 47$:    JSR     PC,ERR.HRD.RTNE.APT
001172  000404                                  BR      50$                             ;                       47??
001174  012746  000055                  48$:    MOV     #55, (SP)                       ;                       47??
001200  004737  000000V                 49$:    JSR     PC,ERR.HRD.RTNL
001204  005726                          50$:    TST     (SP)+                           ;                       47??
001206  000207                                  RTS     PC                              ;                       4561

; Routine Size:  516 words,    Routine Base:  $CODE$ + 21160
; Maximum stack depth per invocation:  6 words
```

```
;    4769  1     GLOBAL routine UPD_IO_TALLY : novalue =
;    4770  1
;    4771  1     !+
;    4772  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    4773  1     !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;    4774  1     !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;    4775  1     !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;    4776  1     !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS DROPPED.
;    4777  1     !
;    4778  1     !       IMPLICIT INPUTS:
;    4779  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;    4780  1     !               T_ADDR  - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;    4781  1     !               CST_ADDR  - ADDRESS OF THE CURRENT CONTROLLER'S CST
;    4782  1     !               CUOFF   - CST OFFSET FOR THE CURRENT UNIT
;    4783  1     !               L$LUN - CURRENT UNIT NUMBER
;    4784  1     !-
;    4785  1
;    4786  2         begin
;    4787  2
;    4788  2         local
;    4789  2             THOUSANDS : word,
;    4790  2             MILLIONS : word;                                        ! TOTAL NO. OF BYTES XFERRED TO/FROM A UNIT
;    4791  2
;    4792  3         if .RP_ADDR [ENDCOD] eql (OP_RD or OP_END)                  ! IF ENDCODE IS READ
;    4793  3         then
;    4794  3             begin
;    4795  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1;     ! INCREMENT NO. OF READS AND ADD BYTE COUNT
;    4796  3             T_ADDR [BYTES_READ_LO] = .T_ADDR [BYTES_READ_LO] + .RP_ADDR [BCNT_LO];
;    4797  3             T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + .RP_ADDR [BCNT_LO];
;    4798  3             OVF_CHK (T_ADDR [TOT_READS_LO]);                        ! CHECK FOR FIELD OVERFLOW
;    4799  3             OVF_CHK (T_ADDR [BYTES_READ_LO]);                       !
;    4800  3             OVF_CHK (T_ADDR [TOT_BYT_RED_LO]);                      !
;    4801  3             end
;    4802  2         else
;    4803  2
;    4804  3             if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)             ! IF ENDCODE IS WRITE
;    4805  3             then
;    4806  3                 begin
;    4807  3                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1;   ! INCREMENT NO. OF WRITES, ADD BYTE COUNT
;    4808  3                 T_ADDR [BYTES_WRIT_LO] = .T_ADDR [BYTES_WRIT_LO] + .RP_ADDR [BCNT_LO];
;    4809  3                 T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + .RP_ADDR [BCNT_LO];
;    4810  3                 OVF_CHK (T_ADDR [TOT_WRITES_LO]);                   ! CHECK FOR FIELD OVERFLOW
;    4811  3                 OVF_CHK (T_ADDR [BYTES_WRIT_LO]);                   !
;    4812  3                 OVF_CHK (T_ADDR [TOT_BYT_WRT_LO]);                  !
;    4813  3                 end;
;    4814  2
;    4815  2         if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) or
;    4816  3            (.RP_ADDR [ENDCOD] eql (OP_WRT or OP_END))
;    4817  2         then
;    4818  3             begin
;    4819  3             MILLIONS = .T_ADDR [MBYTES_READ] + .T_ADDR [MBYTES_WRIT];   ! TOTAL BYTES TRANSFERRED
;    4820  3             THOUSANDS = .T_ADDR [BYTES_READ_HI] + .T_ADDR [BYTES_WRIT_HI];
;    4821  3
```

```
;    4832  3              IF .THOUSANDS gequ 1000
;    4833  3              then
;    4834  4                  begin
;    4835  4                  MILLIONS = .MILLIONS + 1;                          ! COUNT THE LOWER OVERFLOW TOO!
;    4836  4                  THOUSANDS = .THOUSANDS - 1000;
;    4837  3                  end;
;    4838  3
;    4839  3          !
;    4840  3          ! THIS ADDED BECAUSE IT WILL TAKE FOREVER TO TRANSFER ON THE ORDER OF A MEGABYTE TO A FLOPPY
;    4841  3          ! BUT IT IS A MUCH MORE REASONABLE MEASURE FOR THE RD51/52 WINCHESTER.  THE QUESTION NOW REFERS TO
;    4842  3          ! THE TOTAL DATA TRANSFER TO THE CONTROLLER AND THIS IS PRETTY CLOSE SINCE THE FLOPPIES GET
;    4843  3          ! ABOUT 1/1000 THE DATA THE HARD DISK(S) GET.
;    4844  3          !
;    4845  3
;    4846  3              IF .SWP_XFER eql 0                                     ! IF THERE IS A TRANSFER LIMIT
;    4847  3              then
;    4848  4                  begin
;    4849  4
;    4850  4                  IF .THOUSANDS gtru 100
;    4851  4                  then
;    4852  4                      EOP_FLAG = TRUE;                               ! SET END-OF-PASS FLAG
;    4853  4
;    4854  4                  end
;    4855  3              else
;    4856  3
;    4857  3                  IF .MILLIONS gequ .SWP_XFER                        ! IF TRANSFER LIMIT IS REACHED
;    4858  3                  then
;    4859  3                      EOP_FLAG = TRUE;                               ! SET END OF PASS FLAG
;    4860  3
;    4861  2                  end;                                               ! IF UNIT IS STILL ACTIVE
;    4862  2
;    4863  2              ROUND_OUTPUT ();                                       ! ROUND TOTALS TO FIT PRINT POSITIONS
;    4864  1              end;                                                   ! ROUTINE UPD_IO_TALLY


                                        .SBTTL  UPD.IO.TALLY MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          UPD.IO.TALLY::
                                JSR     R1,$SAVE2                       ;
000004  013701  000000G                 MOV     RP.ADDR,R1              ;
000010  126127  000014  000241          CMPB    14(R1),#241
000016  001027                          BNE     1$
000020  013700  000000G                 MOV     T.ADDR,R0               ;
000024  005260  000016                  INC     16(R0)
000030  066110  000020                  ADD     20(R1),(R0)             ;
000034  066160  000020  000032          ADD     20(R1),32(R0)           ;
000042  012746  000016                  MOV     #16.,(SP)               ;
000046  060016                          ADD     R0,(SP)
000050  004737  000000V                 JSR     PC,OVF.CHK
000054  013716  000000G                 MOV     T.ADDR,(SP)             ;
000060  004737  000000V                 JSR     PC,OVF.CHK
000064  013716  000000G                 MOV     T.ADDR,(SP)             ;
000070  062716  000032                  ADD     #32,(SP)
000074  000435                          BR      2$
```

```
000076  126127  000014  000242          1$:     CMPB    14(R1),#0242            ;                       4804
000104  001034                                  BNE     3$
000106  013700  000000G                         MOV     T.ADDR,R0               ;                       4807
000112  005260  000024                          INC     24(R0)
000116  066160  000020  000006                 ADD     20(R1),6(R0)            ;                       4808
000124  066160  000020  000040                 ADD     20(R1),40(R0)           ;                       4809
000132  012746  000024                          MOV     #24,-(SP)               ;                       4810
000136  060016                                  ADD     R0,(SP)
000140  004737  000000V                         JSR     PC,OVF.CHK
000144  013716  000000G                         MOV     T.ADDR,(SP)             ;                       4811
000150  062716  000006                          ADD     #6,(SP)
000154  004737  000000V                         JSR     PC,OVF.CHK
000160  013716  000000G                         MOV     T.ADDR,(SP)             ;                       4812
000164  062716  000040                          ADD     #40,(SP)
000170  004737  000000V                 2$:     JSR     PC,OVF.CHK
000174  005726                                  TST     (SP)+                   ;                       4806
000176  013700  000000G                 3$:     MOV     RP.ADDR,R0              ;                       4815
000202  126027  000014  000241                 CMPB    14(R0),#0241
000210  001404                                  BEQ     4$
000212  126027  000014  000242                 CMPB    14(R0),#0242            ;                       4816
000220  001034                                  BNE     8$
000222  013700  000000G                 4$:     MOV     T.ADDR,R0               ;                       4819
000226  016002  000004                          MOV     4(R0),R2                ; *,MILLIONS
000232  066002  000012                          ADD     12(R0),R2               ; *,MILLIONS
000236  016001  000002                          MOV     2(R0),R1                ; *,THOUSANDS           4820
000242  066001  000010                          ADD     10(R0),R1               ; *,THOUSANDS
000246  020127  001750                          CMP     R1,#01750               ; THOUSANDS,*           4822
000252  103403                                  BLO     5$
000254  005202                                  INC     R2                      ; MILLIONS              4825
000256  162701  001750                          SUB     #1750,R1                ; *,THOUSANDS           4826
000262  013700  000000G                 5$:     MOV     SWP.XFER,R0             ;                       4836
000266  001004                                  BNE     6$
000270  020127  000144                          CMP     R1,#0144                ; THOUSANDS,*           4840
000274  101406                                  BLOS    8$
000276  000402                                  BR      7$                      ;                       4842
000300  020200                         6$:     CMP     R2,R0                   ; MILLIONS,*            4847
000302  103403                                  BLO     8$
000304  112737  000001  0000006         7$:     MOVB    #1,EOP.FLAG             ;                       4849
000312  004737  000000V                 8$:     JSR     PC,ROUND.OUTPUT         ;                       4855
000316  000207                                  RTS     PC                      ;                       4769

; Routine Size:  104 words,    Routine Base:  $CODE$ + 22350
; Maximum stack depth per invocation:  5 words
```

```
;    4855  1     GLOBAL routine OVF_CHK (ADDR) : novalue =
;    4856  1     !+
;    4857  1     !       THIS ROUTINE IS CALLED FROM UPD_IO_TALLY TO CHECK FOR OVERFLOW IN
;    4858  1     !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;    4859  1     !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;    4860  1     !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;    4861  1     !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IN INCREMENTED.
;    4862  1     !
;    4863  1     !       INPUTS:
;    4864  1     !               ADDR - ADDRESS OF THE BYTES_READ_LO OR BYTES_WRIT_LO FIELD FOR
;    4865  1     !                      THE CURRENT UNIT (SEE STATISTIC TABLE (TALLY) LAYOUT)
;    4866  1     !-
;    4867  2         begin
;    4868  2
;    4869  2         while ..ADDR gequ 1000 do                   ! IF LO-ORDER OVERFLOW
;    4870  3             begin
;    4871  3             .ADDR = ..ADDR - 1000;                  ! SUBTRACT 1000
;    4872  3             (.ADDR + 2) = .(.ADDR + 2) + 1;         ! INCR HI-ORDER
;    4873  2             end;
;    4874  2
;    4875  2         if .(.ADDR + 2) gequ 1000                   ! IF HI-ORDER OVERFLOW
;    4876  2         then
;    4877  3             begin
;    4878  3             (.ADDR + 2) = .(.ADDR + 2) - 1000;      ! SUBTRACT 1000
;    4879  3             (.ADDR + 4) = .(.ADDR + 4) + 1;         ! INCREMENT MBYTES
;    4880  2             end;
;    4881  2
;    4882  1         end;                                        ! ROUTINE OVF_CHK

                                .SBTTL  OVF.CHK MULTI-DRIVE TEST ROUTINES
000000  010146          OVF.CHK::
                                MOV     R1, (SP)                ;                       4855
000002  016600  000004          MOV     4(SP),R0                ; ADDR,*                4869
000006  012701  000002          MOV     #2,R1                   ;                       4871
000012  060001                  ADD     R0,R1
000014  021027  001750  1$:     CMP     (R0),#1750              ;                       4869
000020  103404                  BLO     2$
000022  162710  001750          SUB     #1750,(R0)              ;                       4871
000026  005211                  INC     (R1)                    ;                       4873
000030  000771                  BR      1$                      ;                       4869
000032  021127  001750  2$:     CMP     (R1),#1750              ;                       4875
000036  103404                  BLO     3$
000040  162711  001750          SUB     #1750,(R1)              ;                       4877
000044  005260  000004          INC     4(R0)                   ;                       4879
000050  012601          3$:     MOV     (SP)+,R1                ;                       4855
000052  000207                  RTS     PC

; Routine Size:  22 words,    Routine Base:  $CODE$ + 22670
; Maximum stack depth per invocation:  2 words
```

```
;    4883  1     GLOBAL routine ROUND_OUTPUT : novalue =
;    4884  1
;    4885  1     !+
;    4886  1     !       THIS ROUTINE ROUNDS THE TOTALS TO FIT PRINT POSITIONS.
;    4887  1     !-
;    4888  1
;    4889  2         begin
;    4890  2
;    4891  2         if .T_ADDR [TOT_READS_HI] gtru 9999
;    4892  2         then
;    4893  3             begin
;    4894  3
;    4895  3             if .T_ADDR [TOT_READS_LO] lssu 999
;    4896  3             then
;    4897  4                 begin
;    4898  4                 T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 1;
;    4899  4                 T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] + 1000;
;    4900  3                 end;
;    4901  3
;    4902  3             T_ADDR [TOT_READS_LO] = .T_ADDR [TOT_READS_LO] - 999;
;    4903  3             T_ADDR [TOT_READS_HI] = .T_ADDR [TOT_READS_HI] - 9999;
;    4904  2             end;
;    4905  2
;    4906  2         if .T_ADDR [TOT_WRITES_HI] gtru 9999
;    4907  2         then
;    4908  3             begin
;    4909  3
;    4910  3             if .T_ADDR [TOT_WRITES_LO] lssu 999
;    4911  3             then
;    4912  4                 begin
;    4913  4                 T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 1;
;    4914  4                 T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] + 1000;
;    4915  3                 end;
;    4916  3
;    4917  3             T_ADDR [TOT_WRITES_LO] = .T_ADDR [TOT_WRITES_LO] - 999;
;    4918  3             T_ADDR [TOT_WRITES_HI] = .T_ADDR [TOT_WRITES_HI] - 9999;
;    4919  2             end;
;    4920  2
;    4921  2         if .T_ADDR [MTOT_BYT_RED] gtru 999
;    4922  2         then
;    4923  3             begin
;    4924  3
;    4925  3             if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    4926  3             then
;    4927  4                 begin
;    4928  4                 T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    4929  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    4930  3                 end;
;    4931  3
;    4932  3             if .T_ADDR [TOT_BYT_RED_LO] lssu 999
;    4933  3             then
;    4934  4                 begin
;    4935  4                 T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 1;
```

```
;    4936  4                T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] + 1000;
;    4937  4
;    4938  4                if .T_ADDR [TOT_BYT_RED_HI] lssu 999
;    4939  4                then
;    4940  5                    begin
;    4941  5                    T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 1;
;    4942  5                    T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] + 1000;
;    4943  4                    end;
;    4944  3                end;
;    4945  3
;    4946  3            T_ADDR [TOT_BYT_RED_LO] = .T_ADDR [TOT_BYT_RED_LO] - 999;
;    4947  3            T_ADDR [TOT_BYT_RED_HI] = .T_ADDR [TOT_BYT_RED_HI] - 999;
;    4948  3            T_ADDR [MTOT_BYT_RED] = .T_ADDR [MTOT_BYT_RED] - 999;
;    4949  2            end;
;    4950  2
;    4951  2        if .T_ADDR [MTOT_BYT_WRT] gtru 999
;    4952  2        then
;    4953  3            begin
;    4954  3
;    4955  3            if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    4956  3            then
;    4957  4                begin
;    4958  4                T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    4959  4                T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    4960  3                end;
;    4961  3
;    4962  3            if .T_ADDR [TOT_BYT_WRT_LO] lssu 999
;    4963  3            then
;    4964  4                begin
;    4965  4                T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 1;
;    4966  4                T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] + 1000;
;    4967  4
;    4968  4                if .T_ADDR [TOT_BYT_WRT_HI] lssu 999
;    4969  4                then
;    4970  5                    begin
;    4971  5                    T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 1;
;    4972  5                    T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] + 1000;
;    4973  4                    end;
;    4974  3                end;
;    4975  3
;    4976  3            T_ADDR [TOT_BYT_WRT_LO] = .T_ADDR [TOT_BYT_WRT_LO] - 999;
;    4977  3            T_ADDR [TOT_BYT_WRT_HI] = .T_ADDR [TOT_BYT_WRT_HI] - 999;
;    4978  3            T_ADDR [MTOT_BYT_WRT] = .T_ADDR [MTOT_BYT_WRT] - 999;
;    4979  2            end;
;    4980  2
;    4981  1        end;


                                .SBTTL  ROUND.OUTPUT MULTI-DRIVE TEST ROUTINES
000000  004137  000006    ROUND.OUTPUT::
                            JSR     R1,$SAVE3                       ;                       4885
000004  013700  000006            MOV     T.ADDR,R0                       ;               4891
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                  ;                       4895
000030  060001                          ADD     R0,R1
000032  021127  001747                  CMP     (R1),#1747
000036  103003                          BHIS    1$
000040  005312                          DEC     (R2)                    ;                       4898
000042  062711  001750                  ADD     #1750,(R1)              ;                       4899
000046  162711  001747          1$:     SUB     #1747,(R1)              ;                       4902
000052  162712  023417                  SUB     #23417,(R2)             ;                       4903
000056  012702  000026          2$:     MOV     #26,R2                  ;                       4906
000062  060002                          ADD     R0,R2
000064  021227  023417                  CMP     (R2),#23417
000070  101415                          BLOS    4$
000072  012701  000024                  MOV     #24,R1                  ;                       4910
000076  060001                          ADD     R0,R1
000100  021127  001747                  CMP     (R1),#1747
000104  103003                          BHIS    3$
000106  005312                          DEC     (R2)                    ;                       4913
000110  062711  001750                  ADD     #1750,(R1)              ;                       4914
000114  162711  001747          3$:     SUB     #1747,(R1)              ;                       4917
000120  162712  023417                  SUB     #23417,(R2)             ;                       4918
000124  012703  000036          4$:     MOV     #36,R3                  ;                       4921
000130  060003                          ADD     R0,R3
000132  021327  001747                  CMP     (R3),#1747
000136  101436                          BLOS    7$
000140  012701  000034                  MOV     #34,R1                  ;                       4925
000144  060001                          ADD     R0,R1
000146  021127  001747                  CMP     (R1),#1747
000152  103003                          BHIS    5$
000154  005313                          DEC     (R3)                    ;                       4928
000156  062711  001750                  ADD     #1750,(R1)              ;                       4929
000162  012702  000032          5$:     MOV     #32,R2                  ;                       4932
000166  060002                          ADD     R0,R2
000170  021227  001747                  CMP     (R2),#1747
000174  103011                          BHIS    6$
000176  005311                          DEC     (R1)                    ;                       4935
000200  062712  001750                  ADD     #1750,(R2)              ;                       4936
000204  021127  001747                  CMP     (R1),#1747              ;                       4939
000210  103003                          BHIS    6$
000212  005313                          DEC     (R3)                    ;                       4942
000214  062711  001750                  ADD     #1750,(R1)              ;                       4943
000220  162712  001747          6$:     SUB     #1747,(R2)              ;                       4946
000224  162711  001747                  SUB     #1747,(R1)              ;                       4947
000230  162713  001747                  SUB     #1747,(R3)              ;                       4948
000234  012702  000044          7$:     MOV     #44,R2                  ;                       4951
000240  060002                          ADD     R0,R2
000242  021227  001747                  CMP     (R2),#1747
000246  101435                          BLOS    10$
000250  012701  000042                  MOV     #42,R1                  ;                       4955
000254  060001                          ADD     R0,R1
000256  021127  001747                  CMP     (R1),#1747
```

```
000262  103003                          BHIS    8$
000264  005312                          DEC     (R2)                    ;                       4958
000266  062711  001750                  ADD     #1750,(R1)              ;                       4959
000272  062700  000040          8$:     ADD     #40,R0                  ;                       4962
000276  021027  001747                  CMP     (R0),#1747
000302  103011                          BHIS    9$
000304  005311                          DEC     (R1)                    ;                       4965
000306  062710  001750                  ADD     #1750,(R0)              ;                       4966
000312  021127  001747                  CMP     (R1),#1747              ;                       4968
000316  103003                          BHIS    9$
000320  005312                          DEC     (R2)                    ;                       4971
000322  062711  001750                  ADD     #1750,(R1)              ;                       4972
000326  162710  001747          9$:     SUB     #1747,(R0)              ;                       4976
000332  162711  001747                  SUB     #1747,(R1)              ;                       4977
000336  162712  001747                  SUB     #1747,(R2)              ;                       4978
000342  000207                  10$:    RTS     PC                      ;                       4883

; Routine Size:  114 words,    Routine Base:  $CODE$ + 22744
; Maximum stack depth per invocation:  5 words
```

```
;    4982  1     GLOBAL routine HOST_WRT_CHK =
;    4983  1
;    4984  1     !+
;    4985  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;    4986  1     !       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
;    4987  1     !       OPTION WAS SELECTED BY THE OPERATOR.
;    4988  1     !
;    4989  1     !       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
;    4990  1     !       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
;    4991  1     !       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
;    4992  1     !       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A BYTE-BY-BYTE
;    4993  1     !       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
;    4994  1     !       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
;    4995  1     !
;    4996  1     !       IMPLICIT INPUTS:
;    4997  1     !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;    4998  1     !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;    4999  1     !-
;    5000  1
;    5001  2         begin
;    5002  2
;    5003  2         local
;    5004  2             BUFF1 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;    5005  2             BUFF2 : ref block [MAX_XFER_SIZE, byte],          ! I/O BUFFER ADDRESS
;    5006  2             BUFFW,                                            ! I/O BUFFER ADDRESS
;    5007  2             COUNT : word,                                     ! BYTE COUNT
;    5008  2             FLAG : byte initial (byte (TRUE)),
;    5009  2             index : signed word;
;    5010  2
;    5011  3         if .RP_ADDR [ENDCOD] eql (OP_WRT or OP_END)           ! IF WRITE OPERATION
;    5012  2         then
;    5013  2             FLAG = FALSE                                      ! DON'T CALL SWEEP FROM IO_RETPKT
;    5014  2         else
;    5015  2
;    5016  2             if (.RP_ADDR [ENDCOD] eql (OP_RD or OP_END)) and
;    5017  3                ((index = RPS_REM ()) geq 0)                   ! IF ASSOCIATED WRITE PACKET IS FOUND
;    5018  2             then
;    5019  3                 begin
;    5020  3                 BUFFW = RETPKT [.index, BUFF_O];              ! ADDR OF ADDR OF WRITE I/O BUFFER
;    5021  3                 BUFF1 = ..BUFFW;                              ! ADDR OF WRITE I/O BUFFER
;    5022  3                 BUFF2 = .RP_ADDR [BUFF_O];                    ! ADDR OF READ I/O BUFFER
;    5023  3                 COUNT = .RP_ADDR [BCNT_LO];                   ! BYTE COUNT
;    5024  3
;    5025  3                 incr I from 1 to .COUNT do                    ! FOR EACH BYTE IN BUFFERS
;    5026  3
;    5027  3                     if .(.BUFF1)<0, 8, 0> eql .(.BUFF2)<0, 8, 0>   ! IF BYTES COMPARE O.K.
;    5028  3                     then
;    5029  4                         begin
;    5030  4                         BUFF1 = .BUFF1 + 1;                   ! ADVANCE WRITE BUFFER ADDR
;    5031  4                         BUFF2 = .BUFF2 + 1;                   ! ADVANCE READ BUFFER ADDR
;    5032  4                         end
;    5033  3                     else
;    5034  4                         begin                                 ! ELSE - COMPARE ERROR
```

```
;    5035  4                      T_ADDR [ERR_HARD] = .T_ADDR [ERR_HARD] + 1;
;    5036  4                      T_ADDR [ERR_HRD_HST] = .T_ADDR [ERR_HRD_HST] + 1;
;    5037  4
;    5038  4                      if .APT_MODE
;    5039  4                      then
;    5040  4                          ERR_HRD_RTNE_APT (42)                    ! I/O REQUEST FAILED
;    5041  4                      else
;    5042  4                          ERR_HRD_RTNE (42);
;    5043  4
;    5044  4                      EMS_CMP (RETPKT + (.index * RP_LEN * 2));
;    5045  4
;    5046  4                      if .T_ADDR [ERR_HARD] gequ .SWP_ERROR
;    5047  4                      then
;    5048  5                          begin
;    5049  5                          DUR [.L$LUN] = DU_HERR;                  ! IF ERROR COUNT EXCEEDED
;    5050  5                          DODU (.L$LUN);                           ! DROP UNIT
;    5051  4                          end;
;    5052  4
;    5053  4                      exitloop;                                    ! NO NEED TO CONTINUE
;    5054  3                      end;                                         ! IF COMPARE ERROR
;    5055  3
;    5056  2                  end;                                             ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;    5057  2
;    5058  2          return (.FLAG);
;    5059  1          end;                                                     ! ROUTINE HOST_WRT_CHK

                                        .SBTTL  HOST.WRT.CHK MULTI-DRIVE TEST ROUTINES
000000  004137  000000G          HOST.WRT.CHK::
                                 JSR     R1,$SAVE5                ;                          4982
000004  005746                   TST     -(SP)
000006  112705  000001           MOVB    #1,R5                    ; *,FLAG                   5001
000012  013700  000000G          MOV     RP.ADDR,R0               ;                          5011
000016  126027  000014  000242   CMPB    14(R0),#242
000024  001002                   BNE     1$
000026  105005                   CLRB    R5                       ; FLAG                     5013
000030  000511                   BR      8$                       ;                          5011
000032  126027  000014  000241   1$:     CMPB    14(R0),#241      ;                          5016
000040  001105                   BNE     8$
000042  004737  000000V          JSR     PC,RPS.REM               ;                          5017
000046  005700                   TST     R0                       ; INDEX
000050  002501                   BLT     8$
000052  010046                   MOV     R0,-(SP)                 ; INDEX,*                  5020
000054  012746  000054           MOV     #54,-(SP)
000060  004737  000000G          JSR     PC,BL$MUL
000064  010066  000004           MOV     R0,4(SP)
000070  062700  000024G          ADD     #RETPKT+24,R0            ; *,BUFFW
000074  011001                   MOV     (R0),R1                  ; BUFFW,BUFF1              5021
000076  013700  000000G          MOV     RP.ADDR,R0               ;                          5022
000102  016002  000024           MOV     24(R0),R2                ; *,BUFF2
000106  016004  000020           MOV     20(R0),R4                ; *,COUNT                  5023
000112  005003                   CLR     R3                       ; I                        5025
000114  000453                   BR      6$
```

```
000116  121112                          2$:     CMPB    (R1),(R2)               ; BUFF1,BUFF2                   5027
000120  001003                                  BNE     3$
000122  005201                                  INC     R1                      ; BUFF1                         5030
000124  005202                                  INC     R2                      ; BUFF2                         5031
000126  000446                                  BR      6$                      ;                               5027
000130  013700  000000G                  3$:     MOV     T.ADDR,R0               ;                               5035
000134  005260  000014                          INC     14(R0)
000140  105260  000051                          INCB    51(R0)                  ;                               5036
000144  032737  000001  001162'                BIT     #1,APT.MODE             ;                               5038
000152  001405                                  BEQ     4$
000154  012716  000052                          MOV     #52,(SP)                ;                               5040
000160  004737  000000V                        JSR     PC,ERR.HRD.RTNE.APT
000164  000404                                  BR      5$                      ;                               5038
000166  012716  000052                  4$:     MOV     #52,(SP)                ;                               5042
000172  004737  000000V                        JSR     PC,ERR.HRD.RTNE
000176  016616  000004                  5$:     MOV     4(SP),(SP)              ;                               5044
000202  062716  000000G                        ADD     #REIPKT,(SP)
000206  004737  000000G                        JSR     PC,EMS.CMP
000212  013700  000000G                        MOV     T.ADDR,R0               ;                               5046
000216  026037  000014  000000G               CMP     14(R0),SWP.ERROR
000224  103412                                  BLO     7$
000226  013700  000000G                        MOV     L$LUN,R0                ;                               5049
000232  112760  000004  000000G               MOVB    #4,DUR(R0)
000240  104451                                  TRAP    51                      ;                               5050
000242  000403                                  BR      7$                      ;                               5034
000244  005203                  6$:     INC     R3                      ; I                             5025
000246  020304                                  CMP     R3,R4                   ; I,COUNT
000250  003722                                  BLE     2$
000252  022626                  7$:     CMP     (SP)+,(SP)+             ;                               5019
000254  005000                  8$:     CLR     R0                      ;                               5058
000256  150500                          BISB    R5,R0                   ; FLAG,+
000260  005726                          TST     (SP)+                   ;                               4982
000262  000207                          RTS     PC

; Routine Size:  90 words,      Routine Base:  $CODE$ + 23310
; Maximum stack depth per invocation:  11 words
```

```
;    5060  1     GLOBAL routine SWEEP : novalue =
;    5061  1
;    5062  1     !+
;    5063  1     !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;    5064  1     !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;    5065  1     !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;    5066  1     !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;    5067  1     !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;    5068  1     !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;    5069  1     !       DEALLOCATED.
;    5070  1     !
;    5071  1     !       IMPLICIT INPUTS:
;    5072  1     !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;    5073  1     !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;    5074  1     !-
;    5075  1
;    5076  2         begin
;    5077  2
;    5078  2         local
;    5079  2             index : signed word;
;    5080  2
;    5081  2         if (.RP_ADDR [ENDCOD] and OP_MSK) eql OP_RD          ! IF READ OPCODE OR ENDCODE
;    5082  2         then
;    5083  2
;    5084  3             if BIT_TST (SWP_FLAGS, SWF_HWC)                  ! IF HOST IS DOING WRITE-COMPARES
;    5085  2             then
;    5086  2
;    5087  2                 if (index = RPS_REM ()) geq 0                ! IF ASSOCIATED WRITE RETPKT IS FOUND
;    5088  2                 then
;    5089  3                     begin
;    5090  3                     PUT_IO_BUFF (RETPKT [.index, BUFF_O]);   ! RETURN WRITE I/O BUFFER TO POOL
;    5091  3                     PUT_RETPKT (.index);                     ! RETURN WRITE PACKET TO POOL
;    5092  2                     end;
;    5093  2
;    5094  2         PUT_IO_BUFF (RP_ADDR [BUFF_O]);                      ! RETURN CURRENT I/O BUFFER TO POOL
;    5095  2         PUT_RETPKT (.RP_INDX);                               ! RETURN CURRENT RETPKT TO POOL
;    5096  1         end;                                                 ! ROUTINE SWEEP

                                    .SBTTL  SWEEP MULTI-DRIVE TEST ROUTINES
000000  010146                SWEEP:: MOV     R1, (SP)                    ;                       5060
000002  013700  000000G                MOV     RP_ADDR,R0                  ;                       5081
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               ;                       5084
000032  001422                        BEQ     1$
000034  004737  000000V                JSR     PC,RPS_REM                  ;                       5087
000040  010001                        MOV     R0,R1                       ; *,INDEX
000042  002416                        BLT     1$
000044  010146                        MOV     R1, (SP)                    ; INDEX,*               5090
000046  012746  000054                MOV     #54,-(SP)
```

```
00005C  004737  0000006                    JSR     PC,BL$MUL
000062  062700  000024                    ADD     #RETPKT+24,R0
000066  010016                            MOV     R0,(SP)
000064  004737  0000006                    JSR     PC,PUT.IO.BUFF
000070  010116                            MOV     R1,(SP)                 ; INDEX,*                       5091
000074  004737  0000006                    JSR     PC,PUT.RETPKT
000100  026626                            CMP     (SP)+,(SP)+             ;                               5093
000102  013746  0000004                1$: MOV     RP.ADDR, (SP)           ;                               5094
000106  062716  000024                    ADD     #24,(SP)
000112  004737  0000006                    JSR     PC,PUT.IO.BUFF
000116  013716  0000004                    MOV     RP.INDX,(SP)            ;                               5095
000122  004737  0000006                    JSR     PC,PUT.RETPKT
000126  005726                            TST     (SP)+                   ;                               5096
000130  012601                            MOV     (SP)+,R1                ;                               5060
000132  000207                            RTS     PC

; Routine Size:  45 words,    Routine Base:  $CODE$ + 2357A
; Maximum stack depth per invocation:  4 words
```

```
;    5097  1      GLOBAL routine RPS_REM =
;    5098  1
;    5099  1      !+
;    5100  1      !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP SAVE AREA FOR A
;    5101  1      !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;    5102  1      !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;    5103  1      !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;    5104  1      !       RP SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;    5105  1      !       OPERATION IS RETURNED TO THE CALLER.
;    5106  1      !
;    5107  1      !       IMPLICIT INPUTS:
;    5108  1      !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;    5109  1      !
;    5110  1      !       OUTPUTS:
;    5111  1      !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;    5112  1      !                       A CRN WHICH IS ONE LESS THAN THE CURRENT
;    5113  1      !-
;    5114  1
;    5115  2          begin
;    5116  2
;    5117  2          local
;    5118  2              index : signed word initial (-1);                       ! ASSUME NOT FOUND
;    5119  2
;    5120  2          incr COUNT from 0 to RP_CNT - 1 do                          ! FOR EACH ENTRY IN RP SAVE
;    5121  2              if (.RP_USE [.COUNT] eql .CCTLR) and                    ! IF THIS IS A VALID RETPKT INDEX
;    5122  3                 (.RETPKT [.COUNT, ENDCOD] eql (OP_WRT or OP_END))
;    5123  2              then
;    5124  3
;    5125  3                  if ((.RETPKT [.COUNT, CRF_LO] eql (.RP_ADDR [CRF_LO] - 1)) and      ! IF CORRECT CRN
;    5126  3                      (.RETPKT [.COUNT, CRF_HI] eql .RP_ADDR [CRF_HI])) or
;    5127  3                     ((.RETPKT [.COUNT, CRF_HI] eql (.RP_ADDR [CRF_HI] - 1)) and
;    5128  3                      (.RETPKT [.COUNT, CRF_LO] eql %o'177777') and
;    5129  3                      (.RP_ADDR [CRF_LO] eql 0))
;    5130  3                  then
;    5131  4                      begin
;    5132  4                      index = .COUNT;                                ! INDEX TO BE RETURNED
;    5133  4                      exitloop;                                      ! DONE
;    5134  4                      end;
;    5135  2
;    5136  2          return .index;
;    5137  1          end;                                                        ! ROUTINE RPS_REM
```

```
                        .SBTTL  RPS.REM MULTI DRIVE TEST ROUTINES
000000  004157  000000G  RPS.REM::
                        JSR     R1,$SAVE4                       ;
000004  012704  177777          MOV     #-1,R4                  ; #,INDEX
000010  005005                  CLR     R5                      ; COUNT
000012  116500  000000G  1$:     MOVB    RP_USE(R5),R0           ; #(COUNT),#
000016  020057  000000G          CMP     R0,CCTLR
000022  001055                  BNE     4$
000024  010346                  MOV     R3,-(SP)                ; COUNT,#
```

```
000026  012746  000054                        MOV     #54, (SP)
000032  004737  0000006                       JSR     PC,BL$MUL
000036  022626                                CMP     (SP)+,(SP)+
000040  126027  0000146 000242                CMPB    RETPKT+14(R0),#242
000046  001041                                BNE     4$
000050  010346                                MOV     R3, (SP)                ; COUNT,+                       5126
000052  012746  000054                        MOV     #54, -(SP)
000056  004737  0000006                       JSR     PC,BL$MUL
000062  022626                                CMP     (SP)+,(SP)+
000064  013701  0000006                       MOV     RP,ADDR,R1
000070  016102  000004                        MOV     4(R1),R2
000074  005302                                DEC     R2
000076  026002  0000046                       CMP     RETPKT+4(R0),R2
000102  001004                                BNE     2$
000104  026061  0000066 000006               CMP     RETPKT+6(R0),6(R1)      ;                       5127
000112  001415                                BEQ     3$
000114  016102  000006              2$:       MOV     6(R1),R2                ;                       5128
000120  005302                                DEC     R2
000122  026002  0000066                       CMP     RETPKT+6(R0),R2
000126  001011                                BNE     4$
000130  026027  0000046 177777               CMP     RETPKT+4(R0),#-1        ;                       5129
000136  001005                                BNE     4$
000140  005761  000004                        TST     4(R1)                   ;                       5130
000144  001002                                BNE     4$
000146  010504                      3$:       MOV     R5,R4                   ; COUNT,INDEX           5133
000150  000404                                BR      5$                      ;                       5134
000152  005205                      4$:       INC     R5                      ; COUNT                 5120
000154  020527  000007                        CMP     R5,#7                   ; COUNT,#
000160  003714                                BLE     1$
000162  010400                      5$:       MOV     R4,R0                   ; INDEX,#               5115
000164  000207                                RTS     PC                      ;                       5097

; Routine Size:  59 words,    Routine Base:  $CODE$ + 23726
; Maximum stack depth per invocation:  8 words
```

```
;    5139  1     GLOBAL routine DR_RETPKT : novalue =
;    5140  1
;    5141  1     !+
;    5142  1     !       THIS ROUTINE IS CALLED BY PROC RETPKT FOR ALL PACKETS ORIGINATING AT
;    5143  1     !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;    5144  1     !       FATAL DEVICE ERRORS.
;    5145  1     !
;    5146  1     !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;    5147  1     !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;    5148  1     !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED.
;    5149  1     !
;    5150  1     !       IMPLICIT INPUTS:
;    5151  1     !               RP_INDX   INDEX OF THE CURRENT RETURN PACKET
;    5152  1     !               RP_ADDR   ADDRESS OF THE CURRENT RETURN PACKET
;    5153  1     !               CST_ADDR   ADDRESS OF THE CURRENT CONTROLLER'S CST
;    5154  1     !               CCTLR - CURRENT CONTROLLER NUMBER
;    5155  1     !-
;    5156  1
;    5157  2         begin
;    5158  2
;    5159  2
;    5160  2         PUTA_BUFF ();                           ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;    5161  2
;    5162  2         incr index from 0 to RP_CNT - 1 do      ! FOR EACH ENTRY IN CONTROLLER'S RP SAVE
;    5163  2
;    5164  2             if .RP_USE [.index] eql .CCTLR      ! IF VALID RETPKT INDEX
;    5165  2             then
;    5166  2                 PUT_RETPKT (.index);            ! RETURN RETPKT TO POOL
;    5167  2
;    5168  2         QIO [.CCTLR] = 0;                       ! CLEAR NO. OF OUTSTANDING QIOs
;    5169  2         CST_ADDR [STATE] = OFFLINE;             ! MARK CST OFFLINE
;    5170  2         DROP_CTLR (.CCTLR, DU_CFATAL);          ! DROP CONTROLLER'S UNITS
;    5171  2         PUT_RETPKT (.RP_INDX);                  ! PUT BACK RETPKT
;    5172  1         end;                                    ! ROUTINE DR_RETPKT

                                .SBTTL  DR.RETPKT MULTI-DRIVE TEST ROUTINES
000000  010146          DR.RETPKT::
                        MOV     R1, -(SP)                       ;                               5139
000002  004737  000000G         JSR     PC,PUTA.BUFF            ;                               5160
000006  005001                  CLR     R1                      ; INDEX                         5162
000010  116100  000000G 1$:     MOVB    RP.USE(R1),R0           ; *(INDEX),*                    5164
000014  020037  000000G         CMP     R0,CCTLR
000020  001004                  BNE     2$
000022  010146                  MOV     R1, -(SP)               ; INDEX,*                       5166
000024  004737  000000G         JSR     PC,PUT.RETPKT
000030  005726                  TST     (SP)+
000032  005201          2$:     INC     R1                      ; INDEX                         5162
000034  020127  000007          CMP     R1,#7                   ; INDEX,*
000040  003765                  BLE     1$
000042  013701  000000G         MOV     CCTLR,R1                ;                               5168
000046  105061  000000G         CLRB    DIO(R1)
000052  013700  000000G         MOV     CST.ADDR,R0             ;                               5169
```

```
000056  042760  100000  000002          BIC     #100000,2(R0)
000064  010146                          MOV     R1,-(SP)                ;                5170
000066  012746  000006                  MOV     #6,-(SP)
000072  004737  0000006                 JSR     PC,DROP.CTLR
000076  013716  0000006                 MOV     RP.INDX,(SP)            ;                5171
000102  004737  0000006                 JSR     PC,PUT.RETPKT
000106  022626                          CMP     (SP)+,(SP)+             ;                5157
000110  012601                          MOV     (SP)+,R1                ;                5139
000112  000207                          RTS     PC

; Routine Size:  58 words,    Routine Base:  $CODE$ + 24114
; Maximum stack depth per invocation:  4 words
```

```
;    5173  1      %sbttl 'RDRX INTERRUPT SERVICE ROUTINES'
;    5174  1
;    5175  1      !+
;    5176  1      !       THERE EXISTS AN RDRX INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;    5177  1      !       CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;    5178  1      !       APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
;    5179  1      !       BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;    5180  1      !
;    5181  1
;    5182  2      BGNSRV (AZINTO);
;    5183  2      ICTLR = 0;
;    5184  2      AZINT ();
;    5185  1      ENDSRV;


                                        .SBTTL  AZINTO RDRX INTERRUPT SERVICE ROUTINES
000000  010046                  AZINTO::MOV     R0,-(SP)                        ;                       5182
000002  005037  000104'                 CLR     ICTLR                           ;                       5183
000006  004737  000000V                 JSR     PC,AZINT                        ;                       5184
000012  012600                          MOV     (SP)+,R0                        ;                       5182
000014  000002                          RTI

; Routine Size:  7 words,       Routine Base:  $CODE$ + 24230
; Maximum stack depth per invocation:  2 words
```

```
;    5186  1     GLOBAL routine AZINT : novalue =
;    5187  1
;    5188  1     !+
;    5189  1     !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RDRX CONTROLLERS.
;    5190  1     !       AFTER CALCULATING THE DCT ADDRESS FOR THE INTERRUPTING DEVICE, THIS
;    5191  1     !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;    5192  1     !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;    5193  1     !       OTHERWISE, THE SA VALUE IS CHECKED FOR A FATAL ERROR, AND THE COMMAND
;    5194  1     !       AND RESPONSE RINGS ARE POLLED.
;    5195  1     !
;    5196  1
;    5197  2         begin
;    5198  2         IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);          ! GET DCT ADDRESS
;    5199  2         ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);          ! GET CST ADDRESS
;    5200  2         IRDRX_ADDR = .ICST_ADDR [IP_ADDR];                 ! GET RDRX ADDRESS
;    5201  2         ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2);   ! GET COMM AREA ADDR
;    5202  2         IDCT_ADDR [SA_SAVE] = .IRDRX_ADDR [RCSA, RC_ALL];  ! SAVE SA REGISTER
;    5203  2
;    5204  2         IF .IDCT_ADDR [IG_INT]                             ! IGNORE INTERRUPT?
;    5205  2         then
;    5206  2             return;                                        ! RETURN IF INTERRUPTS IGNORED
;    5207  2
;    5208  3         IF BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)           ! IF FATAL ERROR
;    5209  2         then
;    5210  2             FATAL_ERROR ()
;    5211  2         else
;    5212  3             begin
;    5213  3             POLL_CRING ();                                 ! POLL COMMAND RING
;    5214  3             POLL_RRING ();                                 ! POLL RESPONSE RING
;    5215  2             end;
;    5216  2
;    5217  1         end;
```

```
                                .SBTTL  AZINT RDRX INTERRUPT SERVICE ROUTINES
000000  010146                  AZINT:: MOV     R1, -(SP)                       ;                       5186
000002  005746                          TST     -(SP)
000004  013701  000104'                MOV     ICTLR,R1                        ;                       5198
000010  010146                          MOV     R1,-(SP)
000012  012746  000022                  MOV     #22, -(SP)
000016  004737  000000G                 JSR     PC,@#$MUL
000022  062700  000000G                 ADD     #DCT,R0
000026  010037  000100'                MOV     R0,IDCT_ADDR
000032  010116                          MOV     R1,(SP)                         ;                       5199
000034  012746  000126                  MOV     #126, -(SP)
000040  004737  000000G                 JSR     PC,@#$MUL
000044  062700  000000G                 ADD     #CST,R0
000050  010037  000076'                MOV     R0,ICST_ADDR
000054  011037  000000G                 MOV     (R0),IRDRX_ADDR                 ; ICST_ADDR,*           5200
000060  010116                          MOV     R1,(SP)                         ;                       5201
000062  012746  000050                  MOV     #50, -(SP)
000066  004737  000000G                 JSR     PC,@#$MUL
000072  062700  000000'                ADD     #COMM_AREA,R0
```

```
000076  010037  000074'                       MOV     R0,ICOM.ADDR
000102  013701  000100'                       MOV     IDCT.ADDR,R1            ;                                5202
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                    5204
000132  001016                                BNE     2$                      ;                                5186
000134  016601  000010                        MOV     10(SP),R1               ;                                5208
000140  042701  077777                        BIC     #77777,R1
000144  020127  100000                        CMP     R1,#-100000
000150  001003                                BNE     1$
000152  004737  000000V                       JSR     PC,FATAL.ERROR          ;                                5210
000156  000404                                BR      2$                      ;                                5208
000160  004737  000000V             1$:       JSR     PC,POLL.CRING           ;                                5213
000164  004737  000000V                       JSR     PC,POLL.RRING           ;                                5214
000170  062706  000012             2$:       ADD     #12,SP                  ;                                5186
000174  012601                                MOV     (SP)+,R1
000176  000207                                RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 24246
; Maximum stack depth per invocation:  7 words


;   5218  1
```

```
;    5219  1
;    5220  1      !+
;    5221  1      GLOBAL ROUTINE DUP_RSP : NOVALUE =                                          !ZZZ
;    5222  1
;    5223  1      !+
;    5224  1      !       THIS ROUTINE IS CALLED BY POLLARRING FOR EACH DUP RESPONSE
;    5225  1      !       ITS GENERAL PURPOSE IS TO ACT ON A DATAGRAM OR SEQUENTIAL MESSAGE.
;    5226  1      !       IF THE MESSAGE TYPE IS SEQUENTIAL, THE ROUTINE COPIES THE
;    5227  1      !       CONTENTS OF THE MESSAGE ENVELOPE INTO A RETURN PACKET SO THAT THE
;    5228  1      !       ENVELOPE CAN BE RETURNED TO THE CONTROLLER.
;    5229  1      !
;    5230  1      !       IMPLICIT INPUTS:
;    5231  1      !               ICTLR   INTERRUPTING CONTROLLER NUMBER
;    5232  1      !               IPKT_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;    5233  1      !-
;    5234  2          begin
;    5235  2
;    5236  2          local
;    5237  2              R_INDEX : signed word,
;    5238  2              SRC_ADDR,
;    5239  2              DST_ADDR,
;    5240  2              R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    5241  2          !PRINTX (DER34);
;    5242  2
;    5243  2          incr COUNT from 0 to PKT_CNT - 1 do
;    5244  2
;    5245  2              if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;    5246  2                 (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5247  2                 (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;    5248  2                 ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;    5249  2                 (.MSCP_PKT [.COUNT, CONNID] eql CID_DUP) and
;    5250  3                 ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END)
;    5251  2              then
;    5252  3                  begin
;    5253  3                  P_INDEX = .COUNT;                       ! SET PKT NUMBER
;    5254  3                  exitloop;
;    5255  3                  end;
;    5256  2
;    5257  2          if .P_INDEX lss 0                               ! IF COMMAND NOT FOUND
;    5258  2          then
;    5259  3              begin
;    5260  3              PRINTF (DBM108, .IPKT_ADDR [CRN_LO]);       ! UNKNOWN COMMAND REF. NUMBER
;    5261  3              return;
;    5262  3              end;
;    5263  2
;    5264  2          IF (R_INDEX = GET_RETPKT (.ICTLR)) lss 0        ! IF RETPKT IS NOT AVAILABLE
;    5265  2          then
;    5266  3              PRINTF (DBM112)                             ! "DUP RSP: RETPKT NOT AVAILABLE"
;    5267  2          else
;    5268  3              begin
;    5269  3              SRC_ADDR = .IPKT_ADDR + 6;                  ! SET UP COPY (SKIP OVER PKT DESC)
;    5270  3              R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);   ! START OF ALLOCATED RETPKT
;    5271  3
```

```
;    5272  3          incr COUNT from 1 to RP_LEN do
;    5273  4              begin
;    5274  4              .DST_ADDR = ..SRC_ADDR;          ! COPY 1 WORD
;    5275  4              DST_ADDR = .DST_ADDR + 2;        ! ADVANCE DESTINATION ADDR
;    5276  4              SRC_ADDR = .SRC_ADDR + 2;        ! ADVANCE SOURCE ADDR
;    5277  3              end;                             ! COPY LOOP
;    5278  3
;    5279  3          IN_IODQ (.R_INDEX);                  ! PUT RETPKT INDEX INTO IODQ
;    5280  2          end;                                 ! IF RETPKT WAS ALLOCATED
;    5281  2
;    5282  2
;    5283  2      if .P_INDEX geq 0                        ! IF ASSOC CMD PKT WAS FOUND
;    5284  2      then
;    5285  2          PUT_PKT (.P_INDEX);                  ! RETURN COMMAND PACKET TO POOL
;    5286  2
;    5287  1      end;                                     ! ROUTINE DUP-RSP

                                        .SBTTL  DUP.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G          DUP.RSP::
                                        JSR     R1,$SAVE3                       ;                       5221
000004  013701  000000G                 MOV     IPKT.ADDR,R1                    ;                       5244
000010  005002                          CLR     R2                              ; COUNT                 5243
000012  010246                  1$:     MOV     R2,-(SP)                        ; COUNT,*               5245
000014  012746  000106                  MOV     #106,-(SP)
000020  004737  000000G                 JSR     PC,@#$MUL
000024  022626                          CMP     (SP)+,(SP)+
000026  026061  000012G 000012          CMP     MSCP.PKT+12(R0),12(R1)
000034  001024                          BNE     2$
000036  026061  000014G 000014          CMP     MSCP.PKT+14(R0),14(R1)          ;                       5246
000044  001020                          BNE     2$
000046  026011  000000G                 CMP     MSCP.PKT(R0),(R1)               ;                       5247
000052  001415                          BEQ     2$
000054  105760  000022G                 TSTB    MSCP.PKT+22(R0)                 ;                       5248
000060  100412                          BMI     2$
000062  126027  000011G 000002          CMPB    MSCP.PKT+11(R0),#2              ;                       5249
000070  001006                          BNE     2$
000072  105761  000022                  TSTB    22(R1)                          ;                       5250
000076  100003                          BPL     2$
000100  010237  000000G                 MOV     R2,P.INDEX                      ; COUNT,*               5253
000104  000406                          BR      3$                              ;                       5251
000106  005202                  2$:     INC     R2                              ; COUNT                 5243
000110  020227  000015                  CMP     R2,#15                          ; COUNT,*
000114  003756                          BLE     1$
000116  005737  000000G                 TST     P.INDEX                         ;                       5255
000122  002015                  3$:     BGE     4$
000124  016146  000012                  MOV     12(R1),-(SP)                    ;                       5260
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                           ;                       5261
000150  000207                          RTS     PC                              ;                       5259
```

```
000152  013746  000104'          4$:     MOV     ICTLR,-(SP)                     ;                               5264
000156  004737  000000G                  JSR     PC,GET.RETPKT
000162  010001                          MOV     R0,R1                           ; *,R.INDEX
000164  005726                          TST     (SP)+
000166  005701                          TST     R1                              ; R.INDEX
000170  002007                          BGE     5$
000172  012746  000000G                 MOV     #DBM112,-(SP)                   ;                               5266
000176  012746  000001                  MOV     #1,-(SP)
000202  010600                          MOV     SP,R0                           ; SP,*
000204  104417                          TRAP    17
000206  000424                          BR      7$                              ;                               5264
000210  013702  000000G          5$:     MOV     IPKT,ADDR,R2                    ; *,SRC.ADDR                    5269
000214  062702  000006                  ADD     #6,R2                           ; *,SRC.ADDR
000220  010146                          MOV     R1,-(SP)                        ; R.INDEX,*                     5270
000222  012746  000054                  MOV     #54,-(SP)
000226  004737  000000G                 JSR     PC,BL$MUL
000232  062700  000000G                 ADD     #RETPKT,R0
000236  010003                          MOV     R0,R3                           ; *,DST.ADDR
000240  012700  000026                  MOV     #26,R0                          ; *,COUNT                       5273
000244  012223                  6$:     MOV     (R2)+,(R3)+                     ; SRC.ADDR,DST.ADDR             5274
000246  005300                          DEC     R0                              ; COUNT                         5272
000250  001375                          BNE     6$
000252  010116                          MOV     R1,(SP)                         ; R.INDEX,*                     5279
000254  004737  000000G                 JSR     PC,IN.IODQ
000260  013700  000000G          7$:     MOV     P.INDEX,R0                      ;                               5283
000264  002403                          BLT     8$
000266  010016                          MOV     R0,(SP)                         ;                               5285
000270  004737  000000G                 JSR     PC,PUT.PKT
000274  022626                  8$:     CMP     (SP)+,(SP)+                     ;                               5284
000276  000207                          RTS     PC                              ;                               5221

; Routine Size:  96 words,      Routine Base:  $CODE$ + 24446
; Maximum stack depth per invocation:  9 words

;   5288  1
```

```
;    5289  1     GLOBAL routine FATAL_ERROR : novalue =
;    5290  1
;    5291  1     !+
;    5292  1     !       THIS ROUTINE IS CALLED BY THE INTERRUPT SERVICE ROUTINE (AZINT) UPON
;    5293  1     !       DETECTING AN UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER.
;    5294  1     !       ITS PURPOSE IS TO CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF
;    5295  1     !       THE EXERCISER, AND TO INFORM THE "PROGRAM" PORTION OF THE EVENT VIA
;    5296  1     !       RETURN PACKET.
;    5297  1     !
;    5298  1     !       IMPLICIT INPUTS:
;    5299  1     !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;    5300  1     !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5301  1     !               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S CST
;    5302  1     !               IRDRX_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S IP REGISTER
;    5303  1     !-
;    5304  1
;    5305  2         begin
;    5306  2
;    5307  2         local
;    5308  2             index : signed word,
;    5309  2             U_SAVE : word;
;    5310  2
;    5311  2         SA_REG = .IDCT_ADDR [SA_SAVE];
;    5312  2         U_SAVE = .L$LUN;                                    ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
;    5313  2         C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    5314  2
;    5315  2         if .APT_MODE
;    5316  2         then
;    5317  3             begin
;    5318  3             .MAIL_BOX_TESTNUM = 1;
;    5319  3             .MAIL_BOX_SUBTST = 0;
;    5320  2             end;
;    5321  2
;    5322  2         L$LUN = .ICST_ADDR [OF_UN + OF_DATA, D_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
;    5323  2         ERRDF (14, EGD_14, EMS_14);                         ! FATAL CONTROLLER ERROR
;    5324  2         L$LUN = .U_SAVE;                                    ! RESTORE PRE INTERRUPT CURRENT UNIT
;    5325  2         DRV_CTLERR (.ICTLR);                                ! CLEAN UP DRIVER DATA FOR CONTROLLER
;    5326  2
;    5327  2         if (index = GET_RETPKT (.ICTLR)) lss 0              ! TRY TO GET A RETPKT; IF FAILURE
;    5328  2         then
;    5329  3             PRINTF (DBM18)                                  ! "FATAL_ERROR: RETPKT NOT AVAILABLE"
;    5330  2         else
;    5331  3             begin                                           ! IF RETPKT WAS ALLOCATED
;    5332  3             RETPKT [.index, CONID] = CID_DRIVER;            ! SET CONNECTION ID TO "DRIVER"
;    5333  3             RETPKT [.index, MESTYP] = MT_FATAL;             ! FATAL ERROR
;    5334  3             RETPKT [.index, CTLR] = .ICTLR;                 ! CONTROLLER NUMBER
;    5335  3             IN_IODQ (.index);                               ! LOAD RETPKT INDEX INTO IODQ
;    5336  2             end;                                            ! IF RETPKT WAS ALLOCATED
;    5337  2
;    5338  1         end;                                                ! ROUTINE FATAL_ERR

                                .SBTTL  FATAL_ERROR RDRX INTERRUPT SERVICE ROUTINES
```

```
000000  004137  000000G                FATAL.ERROR::
                                        JSR     R1,$SAVE2                       ;                               5289
000004  013700  000100'                MOV     IDCT.ADDR,R0                    ;                               5311
000010  016037  000002  000000G        MOV     2(R0),SA.REG
000016  013701  000000G                MOV     L$LUN,R1                        ; *,U.SAVE                      5312
000022  013700  000104'                MOV     ICTLR,R0                        ;                               5313
000026  006300                         ASL     R0
000030  105260  000000G                INCB    C.ERR.TBL(R0)
000034  032737  000001  001162'        BIT     #1,APT.MODE                     ;                               5315
000042  001405                         BEQ     1$
000044  012777  000001  001164'        MOV     #1,@MAIL.BOX.TESTNUM            ;                               5318
000052  005077  001166'                CLR     @MAIL.BOX.SUBTST                ;                               5319
000056  013700  000076'        1$:     MOV     ICST.ADDR,R0                    ;                               5322
000062  016002  000006                MOV     6(R0),R2
000066  000302                         SWAB    R2
000070  042702  177760                BIC     #177760,R2
000074  010237  000000G                MOV     R2,L$LUN
000100  104455                         TRAP    55                              ;                               5323
000102  000016                         .WORD   16
000104  000000G                        .WORD   EGD.14
000106  000000G                        .WORD   EMS.14
000110  010137  000000G                MOV     R1,L$LUN                        ; U.SAVE,*                      5324
000114  013746  000104'                MOV     ICTLR,-(SP)                     ;                               5325
000120  004737  000000G                JSR     PC,DRV.CTLERR
000124  013716  000104'                MOV     ICTLR,(SP)                      ;                               5327
000130  004737  000000G                JSR     PC,GET.RETPKT
000134  010001                         MOV     R0,R1                           ; *,INDEX
000136  002007                         BGE     2$
000140  012716  000000G                MOV     #DBM18,(SP)                     ;                               5329
000144  012746  000001                MOV     #1,-(SP)
000150  010600                         MOV     SP,R0                           ; SP,*
000152  104417                         TRAP    17
000154  000424                         BR      3$                              ;                               5331
000156  010116        2$:     2$:     MOV     R1,(SP)                         ; INDEX,*                       5332
000160  012746  000054                MOV     #54,-(SP)
000164  004737  000000G                JSR     PC,BL$MUL
000170  062700  000002G                ADD     #RETPKT+2,R0
000174  112760  000003  000001        MOVB    #3,1(R0)
000202  013702  000104'                MOV     ICTLR,R2                        ;                               5334
000206  042702  177760                BIC     #177760,R2
000212  112710  000060                MOVB    #60,(R0)
000216  150210                         BISB    R2,(R0)
000220  010116                         MOV     R1,(SP)                         ; INDEX,*                       5335
000222  004737  000000G                JSR     PC,IN.IDDQ
000226  022626        3$:     3$:     CMP     (SP)+,(SP)+                     ;                               5336
000230  000207                         RTS     PC                              ;                               5289

; Routine Size:  77 words,    Routine Base:  $CODE$ + 24746
; Maximum stack depth per invocation:  7 words
```

```
;     1    GLOBAL routine POLL_CRING : novalue =
;     1
;     1    !+
;     1    !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (A/INT)
;     1    !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;     1    !       ITS PURPOSE IS TO SCAN THE DEVICE'S COMMAND RING AND CHECK FOR ANY
;     1    !       COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY THE CONTROLLER. SUCH SLOTS
;     1    !       HAVE BEEN RETURNED TO THE HOST, INDICATED BY A ZERO OWNERSHIP BIT. FOR
;     1    !       EACH SLOT THAT HAS BEEN RETURNED TO THE HOST, THE CRING COUNT IS
;     1    !       DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO THE NEXT SLOT IN
;     1    !       THE COMMAND RING.
;     1    !
;     1    !       IMPLICIT INPUTS:
;     1    !               ICTLR     INTERRUPTING CONTROLLER NUMBER
;     1    !               IDCT_ADDR   ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;     1    !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM AREA
;     1    !
;     1    !-
;     1
;     2        begin
;     2
;     3        while ((.IDCT_ADDR [CRING_CNT] gtru 0) and                     ! WHILE # OF COMMANDS IN CRING > 0 AND
;     2            not (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) do  ! CURRENT SLOT IS HOST OWNED
;     3            begin
;     3            IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1;       ! DECREMENT # CMDs IN CRING
;     3            IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;           ! ADVANCE TO NEXT SLOT TO POLL
;     3
;     3            if .IDCT_ADDR [CR_POLL] gtra .IDCT_ADDR [CR_END]          ! IF BEYOND END OF RING
;     3            then
;     3                IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];            ! SET POINTER TO TOP OF CRING
;     3
;     2            end;
;     2
;     2        ICOM_ADDR [CMD_INT] = 0;                                      ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER
;     1        end;


                                        .SBTTL  POLL_CRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G         POLL_CRING::
                                        JSR     R1,$SAVE2                       ;               5557
000004  016701  000100'                 MOV     IDCT_ADDR,R1                    ;               5559
000010  012702  000016                  MOV     #16,R2                          ;               5562
000014  060102                          ADD     R1,R2
000016  105711                  1$:     TSTB    (R1)                            ;               5559
000020  001422                          BEQ     2$
000022  016100  000016                  MOV     16(R1),R0                       ;               5560
000026  016000  000002                  MOV     2(R0),R0
000032  042700  077777                  BIC     #77777,R0
000036  020027  100000                  CMP     R0,#100000
000042  001411                          BEQ     2$
000044  105311                          DECB    (R1)                            ;               5562
000046  062712  000004                  ADD     #4,(R2)                         ;               5563
000052  021261  000012                  CMP     (R2),12(R1)                     ;               5565
000056  101757                          BLOS    1$
```

```
000060  016111  000010                    MOV     10(R1),(R1)             ;                       5367
000064  000754                            BR      1$                      ;                       5369
000066  015700  000004'          2$:     MOV     ICOM,ADDR,R0            ;                       5371
000072  005060  000004                    CLR     4(R0)
000076  000207                            RTS     PC                      ;                       5339

; Routine Size:  30 words,    Routine Base:  $CODE$ + 25200
; Maximum stack depth per invocation:  4 words
```

```
;    5377  1      GLOBAL routine POLL_RRING : novalue =
;    5378  1
;    5379  1
;    5380  1      !+
;    5381  1      !       THIS ROUTINE IS CALLED BY THE RDRX INTERRUPT SERVICE ROUTINE (AZINT)
;    5382  1      !       FOR EACH DEVICE INTERRUPT EXCEPT DURING INITIALIZATION OR FATAL ERROR.
;    5383  1      !       ITS PURPOSE IS TO SCAN THE DEVICE'S RESPONSE RING AND CHECK FOR ANY
;    5384  1      !       SLOTS WHICH HAVE BEEN RETURNED TO THE HOST (OWNERSHIP BIT = 0). FOR
;    5385  1      !       EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS PROCESSED BASED ON ITS
;    5386  1      !       CONNECTION ID (DISK OR DUP). AFTER PROCESSING, THE MESSAGE PACKET
;    5387  1      !       IS RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET
;    5388  1      !       TO 1).
;    5389  1      !
;    5390  1      !       IMPLICIT INPUTS:
;    5391  1      !               ICTLR   NUMBER OF INTERRUPTING CONTROLLER
;    5392  1      !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;    5393  1      !-
;    5394  1
;    5395  1
;    5396  2          begin
;    5397  2
;    5398  2
;    5399  2          while not (BIT_TST (.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) do          ! WHILE O = 0
;    5400  3              begin
;    5401  3              IPKT_ADDR = ..IDCT_ADDR [RR_POLL] - 10;                         ! ADDRESS OF RESPONSE PACKET
;    5402  3
;    5403  3
;    5404  4      IF NOT (.IPKT_ADDR [CONNID] EQL CID_DUP)                                !                  !!!
;    5405  3              THEN                                                            !                  !!!
;    5406  3              (CREDIT_BAL = .CREDIT_BAL + .IPKT_ADDR [CREDITS]);              !                  !!!
;    5407  3      !IT WAS NOTICED THAT DUP WAS SENDING BACK CREDITS WHICH IT SHOULD NOT.   !                  !!!
;    5408  3              selectoneu .IPKT_ADDR [CONNID] of
;    5409  3                  set
;    5410  3
;    5411  3                  [CID_DISK] :        DISK_RSP ();
;    5412  3
;    5413  3                  [CID_DUP]  :        DUP_RSP  ();                            !!!
;    5414  3
;    5415  3                  [otherwise] :       PRINTF (DBM,PO, .IPKT_ADDR [CONNID], .IRDRX_ADDR);
;    5416  3                                                                              ! "BAD CONN ID = XXXXX FROM XXXXXX
;    5417  3                  tes;
;    5418  3
;    5419  3
;    5420  3              IPKT_ADDR [MSGLEN] = MSG_LEN + 2;                               ! RE-INIT PKT FIELDS; MESSAGE LENGTH
;    5421  3              IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;                 ! ADVANCE TO HI ORDER WORD OF RING SLOT
;    5422  3              .IDCT_ADDR [RR_POLL] = .IPKT_ADDR [PKT_HI];                     ! RETURN SLOT TO CONTROLLER
;    5423  3              .IDCT_ADDR [RR_POLL] = ..IDCT_ADDR [RR_POLL] or ED_OWN or ED_FLAG; ! OWNERSHIP TOO
;    5424  3              IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;                 ! ADVANCE TO NEXT RRING SLOT
;    5425  3
;    5426  3              IF .IDCT_ADDR [RR_POLL] gtrm .IDCT_ADDR [RR_END]                ! IF BEYOND END OF RING
;    5427  3
;    5428  4              then
;    5429  4                  IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG];                  ! CYCLE TO TOP OF RING
```

```
;    5426  3
;    5427  2            end;                                          ! WHILE LOOP
;    5428  1
;    5429  1        ICOM_ADDR [RSP_INT] = 0;                          ! CLR RESPONSE INTERRUPT WRD IN RING HEADER
;    5430  1        end;


                                .SBTTL  POLL_RRING RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G       POLL_RRING::
                                JSR     R1,$SAVE3                     ;                                5373
000004  013701  000100'         MOV     IDCT_ADDR,R1                  ;                                5395
000010  062701  000014          ADD     #14,R1
000014  011100                1$: MOV     (R1),R0
000016  016000  000002          MOV     2(R0),R0
000022  042700  077777          BIC     #77777,R0
000026  020027  100000          CMP     R0,#100000
000032  001504                  BEQ     6$
000034  017137  000000  000000G   MOV     @0(R1),IPKT_ADDR              ;                                5397
000042  162737  000012  000000G   SUB     #12,IPKT_ADDR
000050  013700  000000G         MOV     IPKT_ADDR,R0                  ;                                5400
000054  005002                  CLR     R2
000056  156000  000011          BISB    11(R0),R2
000062  020227  000002          CMP     R2,#2
000066  001406                  BEQ     2$
000070  116005  000010          MOVB    10(R0),R5                     ;                                5402
000074  042705  177760          BIC     #177760,R5
000100  060537  000000G         ADD     R5,CREDIT_BAL
000104  005702                2$: TST     R2                            ;                                5407
000106  001003                  BNE     5$
000110  004737  000000V         JSR     PC,DISK_RSP
000114  000421                  BR      5$                            ;                                5404
000116  020227  000002        3$: CMP     R2,#2                         ;                                5409
000122  001003                  BNE     4$
000124  004737  004446'         JSR     PC,DUP_RSP
000130  000413                  BR      5$                            ;                                5404
000132  013746  000000G       4$: MOV     IRDRX_ADDR,-(SP)              ;                                5411
000136  010246                  MOV     R2,-(SP)
000140  012746  000000G         MOV     #DBM,0,-(SP)
000144  012746  000003          MOV     #3,-(SP)
000150  010600                  MOV     SP,R0                         ; SP,*
000152  104417                  TRAP    17
000154  062706  000010          ADD     #10,SP
000160  013700  000000G       5$: MOV     IPKT_ADDR,R0                  ;                                5416
000164  012760  000074  000006    MOV     #74,6(R0)
000172  013702  000100'         MOV     IDCT_ADDR,R2                  ;                                5417
000176  010201                  MOV     R2,R1
000200  062701  000014          ADD     #14,R1
000204  062711  000002          ADD     #2,(R1)
000210  016071  000002  000000    MOV     2(R0),@0(R1)                  ;                                5418
000216  052771  140000  000000    BIS     #140000,@0(R1)                ;                                5419
000224  062711  000002          ADD     #2,(R1)                       ;                                5420
000230  021162  000006          CMP     (R1),6(R2)                    ;                                5422
000234  101667                  BLOS    1$
```

```
000236  016211  000004                        MOV     4(R2),(R1)              ;                       5425
000242  000664                                BR      1$                      ;                       5395
000244  013700  000074'              6$:     MOV     ICOM,ADDR,R0            ;                       5429
000250  005060  000006                        CLR     6(R0)
000254  000207                                RTS     PC                      ;                       5373

; Routine Size:  87 words,    Routine Base:  $CODE$ + 25300
; Maximum stack depth per invocation:  10 words


;  5431  1
```

```
;    5452  1
;    5453  1
;    5454  1     GLOBAL routine DISK_RSP : novalue =
;    5455  1
;    5456  1
;    5457  1     !+
;    5458  1     !       THIS ROUTINE IS CALLED BY POLL_RRING FOR EACH RESPONSE MESSAGE
;    5459  1     !       WHICH HAS A CONNECTION ID INDICATING A DISK MSCP ORIGINATOR
;    5460  1     !       (I.E., ALL EXCEPT DUP RESPONSES). ITS PURPOSE IS TO PASS
;    5461  1     !       CONTROL TO THE APPROPRIATE ROUTINE BASED ON THE MESSAGE TYPE
;    5462  1     !       FIELD (SEQUENTIAL, DATAGRAM, OR CREDIT NOTIFICATION).
;    5463  1     !
;    5464  1     !       IMPLICIT INPUTS:
;    5465  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    5466  1     !                           MESSAGE
;    5467  1     !-
;    5468  1
;    5469  1
;    5470  1
;    5471  1         selectoneu .IPKT_ADDR [MSGTYP] of
;    5472  1
;    5473  1
;    5474  1             set
;    5475  1
;    5476  1             [MT_SEQ] :      SEQUEN ();
;    5477  1
;    5478  1             [MT_DG] :       DATAGM ();
;    5479  1
;    5480  1             [otherwise] :   PRINTF (DBM21, .IPKT_ADDR [MSGTYP]);    ! "MESSAGE TYPE XX RECEIVED"
;    5481  1             tes;

                                        .SBTTL  DISK.RSP RDRX INTERRUPT SERVICE ROUTINES
000000  010146                 DISK.RSP::
                                        MOV     R1, -(SP)                       ;                       5454
000002  013700  000006                  MOV     IPKT.ADDR,R0                    ;                       5471
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  001005                          BNE     1$                              ;                       5475
000030  004737  000000V                 JSR     PC,SEQUEN
000034  000417                          BR      5$                              ;                       5476
000036  020127  000001          1$:     CMP     R1,#1                           ;                       5478
000042  001005                          BNE     .+12
000044  004737  000000V                 JSR     PC,DATAGM
000050  000411                          BR      5$                              ;                       5481
000052  010146                  2$:     MOV     R1, -(SP)                       ;                       5480
000054  012746  000006                  MOV     #DBM21, -(SP)
000060  012746  000002                  MOV     #2, -(SP)
000064  010600                          MOV     SP,R0                           ; SP,*
```

```
000066  104417                          TRAP    17
000070  062706  000006                  ADD     #6,SP
000074  012601                  33:     MOV     (SP)+,R1                ;                       5434
000076  000207                          RTS     PC

; Routine Size:  30 words,    Routine Base:  $CODE$ + 25556
; Maximum stack depth per invocation:  6 words


;    5460  1
;    5462  1
;    5464  1
```

```
;    5465  1      GLOBAL routine SEQUEN : novalue =
;    5466  1
;    5467  1      !+
;    5468  1      !       THIS ROUTINE IS CALLED BY DISK RSP FOR EACH DISK MSCP RESPONSE MESSAGE
;    5469  1      !       WITH THE "SEQUENTIAL" MESSAGE TYPE.  ITS GENERAL PURPOSE IS TO COPY THE
;    5470  1      !       CONTENTS OF THE MESSAGE PACKET INTO A RETURN PACKET SO THAT THE
;    5471  1      !       PACKET CAN BE RETURNED TO THE CONTROLLER.    IN ADDITION,
;    5472  1      !       IF THE COMMAND WAS AN I/O TRANSFER (READ, WRITE, OR ACCESS), THEN SOME
;    5473  1      !       FIELDS OF THE COMMAND PACKET ARE COPIED INTO THE RETURN PACKET.
;    5474  1      !
;    5475  1      !       IMPLICIT INPUTS:
;    5476  1      !               ICTLR   INTERRUPTING CONTROLLER NUMBER
;    5477  1      !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING RESPONSE
;    5478  1      !-
;    5479  1
;    5480  2          begin
;    5481  2
;    5482  2          local
;    5483  2              P_INDEX : signed word initial (-1),                         ! ASSUME NO ASSOCIATED COMMAND PKT
;    5484  2              R_INDEX : signed word,
;    5485  2              SRC_ADDR,
;    5486  2              DST_ADDR,
;    5487  2              R_ADDR : ref block [RP_LEN, word] field (RP_FIELDS);
;    5488  2
;    5489  2          incr COUNT from 0 to PKT_CNT - 1 do
;    5490  2
;    5491  2              if (.MSCP_PKT [.COUNT, CRN_LO] eql .IPKT_ADDR [CRN_LO]) and     ! IF THIS IS THE ASSOC CMD
;    5492  2                 (.MSCP_PKT [.COUNT, CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5493  2                 (.MSCP_PKT [.COUNT, PKT_LO] neqa .IPKT_ADDR [PKT_LO]) and
;    5494  2                 ((.MSCP_PKT [.COUNT, OPCODE] and OP_END) neq OP_END) and
;    5495  2                 (.MSCP_PKT [.COUNT, MSGTYP] eql MT_SEQ) and
;    5496  2                 ((.IPKT_ADDR [OPCODE] and OP_END) eql OP_END) and
;    5497  3                 (.PKT_USE [.COUNT] eql .ICTLR)
;    5498  2              then
;    5499  3                  begin
;    5500  3                  P_INDEX = .COUNT;                                      ! SET PKT NUMBER
;    5501  3                  exitloop;
;    5502  2                  end;
;    5503  2
;    5504  2          if .P_INDEX lss 0                                              ! IF COMMAND NOT FOUND
;    5505  2          then
;    5506  3              begin
;    5507  3              PRINTF (DBM108, .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);  ! UNKNOWN COMMAND REF. NUMBER
;    5508  3              return;
;    5509  2              end;
;    5510  2
;    5511  3          if .MSCP_PKT [.P_INDEX, OPCODE] neq (.IPKT_ADDR [OPCODE] and (not OP_END))  ! IF OPCODE MISMATCH
;    5512  2          then
;    5513  2              PRINTF (DBM111, .MSCP_PKT [.P_INDEX, OPCODE], .IPKT_ADDR [OPCODE], .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO])
;
;    5514  2
;    5515  3          if ((.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or
;    5516  2              (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END))) and
;    5517  3             ((.IPKT_ADDR [STATUS_CODE] eql ST_SUC) and
```

```
;    5518  2              (.IPKT_ADDR [STATUS_SUBCODE] eql 0)) and
;    5519  3             ((.MSCP_PKT [.P_INDEX, BC_LO] neq .IPKT_ADDR [BC_LO]) or
;    5520  3              (.MSCP_PKT [.P_INDEX, BC_HI] neq .IPKT_ADDR [BC_HI]))
;    5521  2          then
; P  5522  2              PRINTF (DBM112,
; P  5523  2                      .MSCP_PKT [.P_INDEX, BC_HI], .MSCP_PKT [.P_INDEX, BC_LO], .IPKT_ADDR [BC_HI], .IPKT_ADDR [BC_LO],
;    5524  2                      .IPKT_ADDR [CRN_HI], .IPKT_ADDR [CRN_LO]);
;    5525  2
;    5526  2          if .MSCP_PKT [.P_INDEX, RSP_RECEIVED]
;    5527  2          then
;    5528  3              begin
;    5529  3              PRINTF (DBM120, .MSCP_PKT [.P_INDEX, CRN_HI], .MSCP_PKT [.P_INDEX, CRN_LO]);
;    5530  3              PUT_PKT (.P_INDEX);
;    5531  3              return;
;    5532  3              end
;    5533  2          else
;    5534  2              MSCP_PKT [.P_INDEX, RSP_RECEIVED] = TRUE;                          ! MARK RESPONSE RECEIVED
;    5535  2
;    5536  2          if (R_INDEX = GET_RETPKT (.ICTLR)) lss 0                           ! IF RETPKT IS NOT AVAILABLE
;    5537  2          then
;    5538  3              begin
;    5539  3              PRINTF (DBM22);                                                ! "SEQUEN: RETPKT NOT AVAILABLE"
;    5540  3              PUT_PKT (.P_INDEX);
;    5541  3              return;
;    5542  3              end
;    5543  2          else
;    5544  3              begin
;    5545  3              SRC_ADDR = .IPKT_ADDR + 6;                                     ! SET UP COPY (SKIP OVER PKT DESC)
;    5546  3              R_ADDR = DST_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);          ! START OF ALLOCATED RETPKT
;    5547  3
;    5548  4              incr COUNT from 1 to RP_LEN do
;    5549  4                  begin
;    5550  4                  .DST_ADDR = ..SRC_ADDR;                                    ! COPY 1 WORD
;    5551  4                  DST_ADDR = .DST_ADDR + 2;                                  ! ADVANCE DESTINATION ADDR
;    5552  4                  SRC_ADDR = .SRC_ADDR + 2;                                  ! ADVANCE SOURCE ADDR
;    5553  4
;    5554  4                  if .IPKT_ADDR [OPCODE] eql (OP_ONL or OP_END)              ! IF THIS IS THE ONLINE END MESSAGE
;    5555  4                  then
;    5556  4                      if .COUNT eql 10                                       ! SKIP OVER RESERVED WORDS
;    5557  4                      then
;    5558  4                          SRC_ADDR = .SRC_ADDR + 4;                          !       IN ONLINE END   MESSAGE
;    5559  3                  end;                                                       ! COPY LOOP
;    5560  3
;    5561  3              R_ADDR [CTLR] = .ICTLR;                                        ! LOAD CONTROLLER NUMBER INTO PKT
;    5562  3
;    5563  3              if .P_INDEX geq 0                                              ! IF ASSOC. CMD PKT WAS FOUND
;    5564  3              then
;    5565  3
;    5566  3                  if (.IPKT_ADDR [OPCODE] eql (OP_RD or OP_END)) or          ! IF END MESSAGE IS
;    5567  3                     (.IPKT_ADDR [OPCODE] eql (OP_WRT or OP_END)) or         !       READ, WRITE, OR
;    5568  4                     (.IPKT_ADDR [OPCODE] eql (OP_ACC or OP_END))            !       ACCESS
;    5569  4                  then
;    5570  4                      begin
```

```
;    5571  4                  R_ADDR [CMDMOD] = .MSCP_PKT [.P_INDEX, MODIFY];          ! COPY
;    5572  4                  R_ADDR [CBCNT_LO] = .MSCP_PKT [.P_INDEX, BC_LO];         ! RELEVANT
;    5573  4                  R_ADDR [CBCNT_HI] = .MSCP_PKT [.P_INDEX, BC_HI];         ! FIELDS
;    5574  4                  R_ADDR [LBN_LO] = .MSCP_PKT [.P_INDEX, LBN_L];           ! FROM
;    5575  4                  R_ADDR [LBN_HI] = .MSCP_PKT [.P_INDEX, LBN_H];           ! COMMAND
;    5576  4                  R_ADDR [BUFF_0] = .MSCP_PKT [.P_INDEX, BUF_0];           ! PACKET
;    5577  4                  R_ADDR [BUFF_1] = .MSCP_PKT [.P_INDEX, BUF_1];           ! TO RETPKT
;    5578  3                  end;                                                     ! IF ENDCODE WAS READ/WRITE/ACCESS
;    5579  3
;    5580  3              IN_IODQ (.R_INDEX);                                          ! PUT RETPKT INDEX INTO IODQ
;    5581  2              end;                                                         ! IF RETPKT WAS ALLOCATED
;    5582  2
;    5583  2          if (.IPKT_ADDR [STATUS_CODE] neq  ST_SUC) or
;    5584  3             (.IPKT_ADDR [STATUS_SUBCODE] neq 0)
;    5585  2          then
;    5586  2              LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_OCCURED                ! SAVE ERROR CONDITION
;    5587  2          else
;    5588  2              LAST_PKT [.ICTLR, LAST_HRD_ERR] = HRD_NOT_OCCURED;           !
;    5589  2
;    5590  2          LAST_PKT [.ICTLR, LAST_CRN_LO] = .IPKT_ADDR [CRN_LO];            ! SAVE COMMAND REFERENCE NUMBER
;    5591  2          LAST_PKT [.ICTLR, LAST_CRN_HI] = .IPKT_ADDR [CRN_HI];            !
;    5592  2          SCAN_ERRLOG ();                                                  ! PRINT ANY ASSOCIATED ERROR LOGS
;    5593  2
;    5594  2          if .P_INDEX geq 0                                                ! IF ASSOC CMD PKT WAS FOUND
;    5595  2          then
;    5596  2              PUT_PKT (.P_INDEX);                                          ! RETURN COMMAND PACKET TO POOL
;    5597  2
;    5598  1          end;                                                             ! ROUTINE DISK_RSP
```

```
                        .SBTTL  SEQUEN RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G         SEQUEN::JSR     R1,$SAVE5                       ;                                5465
000004  005746                    TST     -(SP)
000006  012746  177777            MOV     #-1,-(SP)                       ; *,P.INDEX                      5480
000012  013701  000000G           MOV     IPKT.ADDR,R1                    ;                                5491
000016  005002                    CLR     R2                              ; COUNT                          5489
000020  010246                 1$:MOV     R2,-(SP)                        ; COUNT,*                        5491
000022  012746  000106            MOV     #106,-(SP)
000026  004737  000000G           JSR     PC,BL$MUL
000032  022626                    CMP     (SP)+,(SP)+
000034  026061  000012G 000012    CMP     MSCP.PKT+12(R0),12(R1)
000042  001030                    BNE     2$
000044  026061  000014G 000014    CMP     MSCP.PKT+14(R0),14(R1)          ;                                5492
000052  001024                    BNE     2$
000054  026011  000000G           CMP     MSCP.PKT(R0),(R1)               ;                                5493
000060  001421                    BEQ     2$
000062  105760  000022G           TSTB    MSCP.PKT+22(R0)                 ;                                5494
000066  100416                    BMI     2$
000070  132760  000360  000010G   BITB    #360,MSCP.PKT+10(R0)            ;                                5495
000076  001013                    BNE     2$
000100  105761  000022            TSTB    22(R1)                          ;                                5496
000104  100007                    BPL     2$
000106  116200  000000G           MOVB    PKT.USE(R2),R0                  ; *(COUNT),*                     5497
```

```
000112  020037  000104'                        CMP     R0,ICTLR
000116  001002                                 BNE     2$
000120  010216                                 MOV     R2,(SP)          ; COUNT,P.INDEX                 5500
000122  000405                                 BR      3$               ;                               5499
000124  005202                          2$:    INC     R2               ; COUNT                         5489
000126  020227  000013                         CMP     R2,#13           ; COUNT,*
000132  003732                                 BLE     1$
000134  005716                                 TST     (SP)             ; P.INDEX                       5504
000136  002013                          3$:    BGE     4$
000140  016146  000012                         MOV     12(R1),-(SP)     ;                               5507
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$               ;                               5508
000166  011646                          4$:    MOV     (SP),-(SP)       ; P.INDEX,*                     5511
000170  012746  000106                         MOV     #106,-(SP)
000174  004737  000000G                        JSR     PC,BL$MUL
000200  010001                                 MOV     R0,R1
000202  022626                                 CMP     (SP)+,(SP)+
000204  013700  000000G                        MOV     IPKT.ADDR,R0
000210  116003  000022                         MOVB    22(R0),R3
000214  042703  177600                         BIC     #177600,R3
000220  005002                                 CLR     R2
000222  156102  000022G                        BISB    MSCP.PKT+22(R1),R2
000226  020203                                 CMP     R2,R3
000230  001422                                 BEQ     5$
000232  016046  000012                         MOV     12(R0),-(SP)     ;                               5513
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$:    MOV     IPKT.ADDR,R0     ;                               5515
000302  126027  000022  000241                CMPB    22(R0),#241
000310  001404                                 BEQ     6$
000312  126027  000022  000242                CMPB    22(R0),#242      ;                               5516
000320  001045                                 BNE     8$
000322  012702  000024                 6$:    MOV     #24,R2           ;                               5517
000326  060002                                 ADD     R0,R2
000330  132712  000037                         BITB    #37,(R2)
000334  001037                                 BNE     8$
000336  032712  177740                         BIT     #177740,(R2)     ;                               5518
000342  001034                                 BNE     8$
000344  026160  000026G 000026                 CMP     MSCP.PKT+26(R1),26(R0)   ;                      5519
000352  001004                                 BNE     7$
000354  026160  000030G 000030                 CMP     MSCP.PKT+30(R1),30(R0)   ;                      5520
```

M16


```
000362  001424                          BEQ     8$
000364  016046  0C0012          7$:     MOV     12(R0),-(SP)                    ;                               5524
000370  016046  000014                  MOV     14(R0),-(SP)
000374  016046  000026                  MOV     26(R0),-(SP)
000400  016046  000030                  MOV     30(R0),-(SP)
000404  016146  000026G                 MOV     MSCP.PKT+26(R1),-(SP)
000410  016146  000030G                 MOV     MSCP.PKT+30(R1),-(SP)
000414  012746  000000G                 MOV     #DBM112,-(SP)
000420  012746  000007                  MOV     #7,-(SP)
000424  010600                          MOV     SP,R0                           ; SP,*
000426  104417                          TRAP    17
000430  062706  000020                  ADD     #20,SP
000434  132761  000400  000005G  8$:     BITB    #400,MSCP.PKT+5(R1)             ;                               5526
000442  001422                          BEQ     10$
000444  016146  000012G                 MOV     MSCP.PKT+12(R1),-(SP)           ;                               5529
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,*                     5530
000474  004737  000000G                 JSR     PC,PUT.PKT
000500  062706  000010          9$:     ADD     #10,SP                          ;                               5531
000504  000137  027016'                 JMP     21$                             ;                               5528
000510  112761  000001  000005G  10$:    MOVB    #1,MSCP.PKT+5(R1)               ;                               5534
000516  013746  000104'                 MOV     ICTLR,-(SP)                     ;                               5536
000522  004737  000000G                 JSR     PC,GET.RETPKT
000526  010066  000004                  MOV     R0,4(SP)                        ; *,R.INDEX
000532  005726                          TST     (SP)+
000534  005766  000002                  TST     2(SP)                           ; R.INDEX
000540  002007                          BGE     11$
000542  012746  000000G                 MOV     #DBM22,-(SP)                    ;                               55'9
000546  012746  000001                  MOV     #1,-(SP)
000552  010600                          MOV     SP,R0                           ; SP,*
000554  104417                          TRAP    17
000556  000563                          BR      19$                             ;                               5540
000560  013704  000000G          11$:    MOV     IPKT.ADDR,R4                    ; *,SRC.ADDR                    5545
000564  062704  000006                  ADD     #6,R4                           ; *,SRC.ADDR
000570  016646  000002                  MOV     2(SP),-(SP)                     ; R.INDEX,*                     5546
000574  012746  000054                  MOV     #54,-(SP)
000600  004737  000000G                 JSR     PC,RL$MUL
000604  062700  000000G                 ADD     #RETPKT,R0
000610  010005                          MOV     R0,R5                           ; *,DST.ADDR
000612  013702  000000G                 MOV     IPKT.ADDR,R2                    ;                               5554
000616  012703  000001                  MOV     #1,R3                           ; *,COUNT                       5548
000622  012425                  12$:    MOV     (R4)+,(R5)+                     ; SRC.ADDR,DST.ADDR             5550
000624  126227  000022  000211          CMPB    22(R2),#211                     ;                               5554
000632  001005                          BNE     13$
000634  020327  000012                  CMP     R3,#12                          ; COUNT,*                       5556
000640  001002                          BNE     13$
000642  062704  000004                  ADD     #4,R4                           ; *,SRC.ADDR                    5558
000646  005203                  13$:    INC     R3                              ; COUNT                         5548
000650  020327  000026                  CMP     R3,#26                          ; COUNT,*
```

ZRQAM2	RD/R....B1	ZRQAM3	RD/R....B5	ZRQAM3	RD/R....B9	ZRQAM3	RD/R....B13
ZRQAM2	RD/R....C1	ZRQAM3	RD/R....C5	ZRQAM3	RD/R....C9	ZRQAM3	RD/R....C13
ZRQAM2	RD/R....D1	ZRQAM3	RD/R....D5	ZRQAM3	RD/R....D9	ZRQAM3	RD/R....D13
ZRQAM2	RD/R....E1	ZRQAM3	RD/R....E5	ZRQAM3	RD/R....E9	ZRQAM3	RD/R....E13
ZRQAM2	RD/R....F1	ZRQAM3	RD/R....F5	ZRQAM3	RD/R....F9	ZRQAM3	RD/R....F13
ZRQAM2	RD/R....G1	ZRQAM3	RD/R....G5	ZRQAM3	RD/R....G9	ZRQAM3	RD/R....G13
ZRQAM2	RD/R....H1	ZRQAM3	RD/R....H5	ZRQAM3	RD/R....H9	ZRQAM3	RD/R....H13
ZRQAM2	RD/R....I1	ZRQAM3	RD/R....I5	ZRQAM3	RD/R....I9	ZRQAM3	RD/R....I13
ZRQAM2	RD/R....J1	ZRQAM3	RD/R....J5	ZRQAM3	RD/R....J9	ZRQAM3	RD/R....J13
ZRQAM2	RD/R....K1	ZRQAM3	RD/R....K5	ZRQAM3	RD/R....K9	ZRQAM3	RD/R....K13
ZRQAM2	RD/R....L1	ZRQAM3	RD/R....L5	ZRQAM3	RD/R....L9	ZRQAM3	RD/R....L13
ZRQAM2	RD/R....M1	ZRQAM3	RD/R....M5	ZRQAM3	RD/R....M9	ZRQAM3	RD/R....M13
ZRQAM2	RD/R....N1	ZRQAM3	RD/R....N5	ZRQAM3	RD/R....N9	ZRQAM3	RD/R....N13
ZRQAM2	RD/R....B2	ZRQAM3	RD/R....B6	ZRQAM3	RD/R....B10	ZRQAM3	RD/R....B14
ZRQAM2	RD/R....C2	ZRQAM3	RD/R....C6	ZRQAM3	RD/R....C10	ZRQAM3	RD/R....C14
ZRQAM2	RD/R....D2	ZRQAM3	RD/R....D6	ZRQAM3	RD/R....D10	ZRQAM3	RD/R....D14
ZRQAM2	RD/R....E2	ZRQAM3	RD/R....E6	ZRQAM3	RD/R....E10	ZRQAM3	RD/R....E14
ZRQAM2	RD/R....F2	ZRQAM3	RD/R....F6	ZRQAM3	RD/R....F10	ZRQAM3	RD/R....F14
ZRQAM2	RD/R....G2	ZRQAM3	RD/R....G6	ZRQAM3	RD/R....G10	ZRQAM3	RD/R....G14
ZRQAM2	RD/R....H2	ZRQAM3	RD/R....H6	ZRQAM3	RD/R....H10	ZRQAM3	RD/R....H14
ZRQAM2	RD/R....I2	ZRQAM3	RD/R....I6	ZRQAM3	RD/R....I10	ZRQAM3	RD/R....I14
ZRQAM2	RD/R....J2	ZRQAM3	RD/R....J6	ZRQAM3	RD/R....J10	ZRQAM3	RD/R....J14
ZRQAM2	RD/R....K2	ZRQAM3	RD/R....K6	ZRQAM3	RD/R....K10	ZRQAM3	RD/R....K14
ZRQAM2	RD/R....L2	ZRQAM3	RD/R....L6	ZRQAM3	RD/R....L10	ZRQAM3	RD/R....L14
ZRQAM3	....M2	ZRQAM3	RD/R....M6	ZRQAM3	RD/R....M10	ZRQAM3	RD/R....M14
ZRQAM3	RD/R....N2	ZRQAM3	RD/R....N6	ZRQAM3	RD/R....N10	ZRQAM3	RD/R....N14
ZRQAM3	RD/R....B3	ZRQAM3	RD/R....B7	ZRQAM3	RD/R....B11	ZRQAM3	RD/R....B15
ZRQAM3	RD/R....C3	ZRQAM3	RD/R....C7	ZRQAM3	RD/R....C11	ZRQAM3	RD/R....C15
ZRQAM3	RD/R....D3	ZRQAM3	RD/R....D7	ZRQAM3	RD/R....D11	ZRQAM3	RD/R....D15
ZRQAM3	RD/R....E3	ZRQAM3	RD/R....E7	ZRQAM3	RD/R....E11	ZRQAM3	RD/R....E15
ZRQAM3	RD/R....F3	ZRQAM3	RD/R....F7	ZRQAM3	RD/R....F11	ZRQAM3	RD/R....F15
ZRQAM3	RD/R....G3	ZRQAM3	RD/R....G7	ZRQAM3	RD/R....G11	ZRQAM3	RD/R....G15
ZRQAM3	RD/R....H3	ZRQAM3	RD/R....H7	ZRQAM3	RD/R....H11	ZRQAM3	RD/R....H15
ZRQAM3	RD/R....I3	ZRQAM3	RD/R....I7	ZRQAM3	RD/R....I11	ZRQAM3	RD/R....I15
ZRQAM3	RD/R....J3	ZRQAM3	RD/R....J7	ZRQAM3	RD/R....J11	ZRQAM3	RD/R....J15
ZRQAM3	RD/R....K3	ZRQAM3	RD/R....K7	ZRQAM3	RD/R....K11	ZRQAM3	RD/R....K15
ZRQAM3	RD/R....L3	ZRQAM3	RD/R....L7	ZRQAM3	RD/R....L11	ZRQAM3	RD/R....L15
ZRQAM3	RD/R....M3	ZRQAM3	RD/R....M7	ZRQAM3	RD/R....M11	ZRQAM3	RD/R....M15
ZRQAM3	RD/R....N3	ZRQAM3	RD/R....N7	ZRQAM3	RD/R....N11	ZRQAM3	RD/R....N15
ZRQAM3	RD/R....B4	ZRQAM3	RD/R....B8	ZRQAM3	RD/R....B12	ZRQAM3	RD/R....B16
ZRQAM3	RD/R....C4	ZRQAM3	RD/R....C8	ZRQAM3	RD/R....C12	ZRQAM3	RD/R....C16
ZRQAM3	RD/R....D4	ZRQAM3	RD/R....D8	ZRQAM3	RD/R....D12	ZRQAM3	RD/R....D16
ZRQAM3	RD/R....E4	ZRQAM3	RD/R....E8	ZRQAM3	RD/R....E12	ZRQAM3	RD/R....E16
ZRQAM3	RD/R....F4	ZRQAM3	RD/R....F8	ZRQAM3	RD/R....F12	ZRQAM3	RD/R....F16
ZRQAM3	RD/R....G4	ZRQAM3	RD/R....G8	ZRQAM3	RD/R....G12	ZRQAM3	RD/R....G16
ZRQAM3	RD/R....H4	ZRQAM3	RD/R....H8	ZRQAM3	RD/R....H12	ZRQAM3	RD/R....H16
ZRQAM3	RD/R....I4	ZRQAM3	RD/R....I8	ZRQAM3	RD/R....I12	ZRQAM3	RD/R....I16
ZRQAM3	RD/R....J4	ZRQAM3	RD/R....J8	ZRQAM3	RD/R....J12	ZRQAM3	RD/R....J16
ZRQAM3	RD/R....K4	ZRQAM3	RD/R....K8	ZRQAM3	RD/R....K12	ZRQAM3	RD/R....K16
ZRQAM3	RD/R....L4	ZRQAM3	RD/R....L8	ZRQAM3	RD/R....L12	ZRQAM3	RD/R....L16
ZRQAM3	RD/R....M4	ZRQAM3	RD/R....M8	ZRQAM3	RD/R....M12	ZRQAM3	RD/R....M16
ZRQAM3	RD/R....N4	ZRQAM3	RD/R....N8	ZRQAM3	RD/R....N12		

```
000654  003762                          BLE     12$
000656  013703  000104'                 MOV     ICTLR,R3                ;                               5561
000662  042703  177760                  BIC     #177760,R3
000666  142760  000017  000002          BICB    #17,2(R0)               ; *,*(R.ADDR)
000674  150360  000002                  BISB    R3,2(R0)                ; *,*(R.ADDR)
000700  005766  000004                  TST     4(SP)                   ; P.INDEX                       5563
000704  002441                          BLT     15$
000706  005003                          CLR     R3                      ;                               5566
000710  156203  000022                  BISB    22(R2),R3
000714  020327  000241                  CMP     R3,#241
000720  001406                          BEQ     14$
000722  020327  000242                  CMP     R3,#242                 ;                               5567
000726  001403                          BEQ     14$
000730  020327  000220                  CMP     R3,#220                 ;                               5568
000734  001025                          BNE     15$
000736  016160  000024G 000012  14$:    MOV     MSCP.PKT+24(R1),12(R0)  ; *,*(R.ADDR)                   5571
000744  016160  000026G 000044          MOV     MSCP.PKT+26(R1),44(R0)  ; *,*(R.ADDR)                   5572
000752  016160  000030G 000046          MOV     MSCP.PKT+30(R1),46(R0)  ; *,*(R.ADDR)                   5573
000760  016160  000046G 000050          MOV     MSCP.PKT+46(R1),50(R0)  ; *,*(R.ADDR)                   5574
000766  016160  000050G 000052          MOV     MSCP.PKT+50(R1),52(R0)  ; *,*(R.ADDR)                   5575
000774  016160  000032G 000024          MOV     MSCP.PKT+32(R1),24(R0)  ; *,*(R.ADDR)                   5576
001002  016160  000034G 000026          MOV     MSCP.PKT+34(R1),26(R0)  ; *,*(R.ADDR)                   5577
001010  016616  000006          15$:    MOV     6(SP),(SP)              ; R.INDEX,*                     5580
001014  004737  000000G                 JSR     PC,IN.IODQ
001020  005726                          TST     (SP)+                   ;                               5544
001022  013716  000104'                 MOV     ICTLR,(SP)              ;                               5586
001026  012746  000006                  MOV     #6,-(SP)
001032  004737  000000G                 JSR     PC,BL$MUL
001036  013701  000000G                 MOV     IPKT.ADDR,R1            ;                               5583
001042  012703  000024                  MOV     #24,R3
001046  060103                          ADD     R1,R3
001050  132713  000037                  BITB    #37,(R3)
001054  001003                          BNE     16$
001056  032713  177740                  BIT     #177740,(R3)            ;                               5584
001062  001404                          BEQ     17$
001064  012760  000001  000116'  16$:    MOV     #1,LAST.PKT(R0)         ;                               5586
001072  000402                          BR      18$                     ;                               5583
001074  005060  000116'          17$:    CLR     LAST.PKT(R0)            ;                               5588
001100  016160  000012  000120'  18$:    MOV     12(R1),LAST.PKT+2(R0)   ;                               5590
001106  016160  000014  000122'          MOV     14(R1),LAST.PKT+4(R0)   ;                               5591
001114  004737  000000V                 JSR     PC,SCAN.ERRLOG          ;                               5592
001120  005766  000004                  TST     4(SP)                   ; P.INDEX                       5594
001124  002404                          BLT     20$
001126  016616  000004          19$:    MOV     4(SP),(SP)              ; P.INDEX,*                     5596
001132  004737  000000G                 JSR     PC,PUT.PKT
001136  022626                  20$:    CMP     (SP)+,(SP)+             ;                               5480
001140  022626                  21$:    CMP     (SP)+,(SP)+             ;                               5465
001142  000207                          RTS     PC

; Routine Size:  306 words,    Routine Base:  $CODE$ + 25656
; Maximum stack depth per invocation:  18 words
```

```
;    5599  1     GLOBAL routine SCAN_ERRLOG : novalue =
;    5600  1
;    5601  1     !+
;    5602  1     ! THIS ROUTINE SCANS THE ERROR-LOG SAVE AREA AND PRINTS ANY ERROR-LOGS RECEIVED FOR THE ASSOCIATED RESPONSE
;    5603  1     !-
;    5604  1
;    5605  2         begin
;    5606  2
;    5607  2         local
;    5608  2             TEMP_UNIT,
;    5609  2             SFT_ERR_PRINTED : byte initial (byte (FALSE));
;    5610  2
;    5611  2         incr index from 0 to EP_CNT do                                          ! SCAN ERROR-LOG PACKET SAVE AREA
;    5612  3             begin
;    5613  3
;    5614  3             if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;    5615  3                (.ELOG_PKT [.index, EL_CRN_LO] eql .IPKT_ADDR [CRN_LO]) and
;    5616  3                (.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5617  4                (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;    5618  3             then
;    5619  4                 begin                                                       ! ERROR-LOG PENDING THIS RESPONSE
;    5620  4                 if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED     ! IF SOFT ERROR OCCURED
;    5621  4                 then
;    5622  4
;    5623  4                     if .ELOG_PKT [.index, EL_FORMAT] lequ 4
;    5624  4                     then
;    5625  4                         begin
;    5626  5                         SOFT_ERROR (.index);                                ! UPATE SOFT ERROR COUNT
;    5627  5                         TEMP_UNIT = .L$LUN;                                 ! SAVE UNIT NUMBER AS KNOWN TO DRS
;    5628  5
;    5629  5                         incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;    5630  5
;    5631  5                             if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_PKT [.index, EL_DK_NUM]) and
;    5632  5                                (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;    5633  6                             then
;    5634  5                                 begin
;    5635  6                                 L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];     ! CORECT UNIT NO. FOR ERROR MESSAGE
;    5636  6                                 exitloop;
;    5637  6                                 end;
;    5638  5
;    5639  5                         case .ELOG_PKT [.index, EL_FORMAT] from 0 to 4 of
;    5640  5                             set
;    5641  5
;    5642  5                             [0]:    if .APT_MODE                            ! CONTROLLER ERROR
;    5643  5                                     then
;    5644  5                                         ERR_SOFT_RTNE_APT (50, .index)
;    5645  5                                     else
;    5646  5                                         ERR_SOFT_RTNE (50);
;    5647  5
;    5648  5                             [1]:    if .APT_MODE                            ! HOST MEMORY ACCESS ERROR
;    5649  5                                     then
;    5650  5                                         ERR_SOFT_RTNE_APT (51, .index)
;    5651  5
```

```
;    5652  5                                    else
;    5653  5                                        ERR_SOFT_RTNE (51);
;    5654  5
;    5655  5                                [2]:    if .APT_MODE                                    ! DISK TRANSFER ERROR
;    5656  5                                    then
;    5657  5                                        ERR_SOFT_RTNE_APT (52, .index)
;    5658  5                                    else
;    5659  5                                        ERR_SOFT_RTNE (52);
;    5660  5
;    5661  5                                [3]:    if .APT_MODE                                    ! SDI ERROR
;    5662  5                                    then
;    5663  5                                        ERR_SOFT_RTNE_APT (53, .index)
;    5664  5                                    else
;    5665  5                                        ERR_SOFT_RTNE (53);
;    5666  5
;    5667  5                                [4]:    if .APT_MODE                                    ! SMALL DISK ERROR
;    5668  5                                    then
;    5669  5                                        ERR_SOFT_RTNE_APT (54, .index)
;    5670  5                                    else
;    5671  5                                        ERR_SOFT_RTNE (54);
;    5672  5                                tes;
;    5673  5
;    5674  5                                    L$LUN = .TEMP_UNIT;                                 ! RESTORE UNIT NUMBER
;    5675  5                                    SFT_ERR_PRINTED = TRUE;                             ! SOFT ERROR PRINTOUT OCCURED
;    5676  5                                    end
;    5677  4                                else
;    5678  4                                    PRINTF (DBM109, .ELOG_PKT [.index, EL_FORMAT]);     ! UNKNOWN ERROR-LOG FORMAT
;    5679  4
;    5680  5                            if not (.SFT_ERR_PRINTED)
;    5681  4                            then
;    5682  4                                PRINTB (CRLF);                                          ! EXTRA CARRIAGE-RETURN/LINE-FEED
;    5683  4
;    5684  4                            EMS_EL (.index);                                            ! PRINT ERROR-LOG CONTENTS
;    5685  4                            end
;    5686  3                        else
;    5687  3
;    5688  3                            if (.ELOG_PKT [.index, EL_CNTR] eql .ICTLR) and
;    5689  4                               ((.ELOG_PKT [.index, EL_CRN_HI] lssu .IPKT_ADDR [CRN_HI]) or
;    5690  5                                ((.ELOG_PKT [.index, EL_CRN_HI] eql .IPKT_ADDR [CRN_HI]) and
;    5691  3                                 (.ELOG_PKT [.index, EL_CRN_LO] lssu .IPKT_ADDR [CRN_LO]))) and
;    5692  4                               (.ELOG_PKT [.index, EL_CONTENTS] eql FULL)
;    5693  3                            then
;    5694  4                                begin
;    5695  4                                PRINTB (CRLF);                                          ! CARRIAGE-RETURN/LINE-FEED
;    5696  4                                EMS_EL (.index);                                        ! PRINT ERROR-LOG CONTENTS
;    5697  3                                end;
;    5698  3
;    5699  2                        end;                                                            ! ERROR-LOG SAVE AREA SCAN
;    5700  2
;    5701  1            end;

                                                .SBTTL  SCAN.ERRLOG RDRX INTERRUPT SERVICE ROUTINES
```

```
000000  004137  000000G                SCAN.ERRLOG::
                                                JSR     R1,$SAVE5                       ;                               5599
000004  005746                                  TST     -(SP)
000006  105005                                  CLRB    R5                              ; SFT.ERR.PRINTED               5605
000010  005002                                  CLR     R2                              ; INDEX                         5611
000012  010246                          1$:     MOV     R2,-(SP)                        ; INDEX,*                       5614
000014  012746  000102                          MOV     #102,-(SP)
000020  004737  000000G                        JSR     PC,BL$MUL
000024  010001                                  MOV     R0,R1
000026  022626                                  CMP     (SP)+,(SP)+
000030  012703  000000G                        MOV     #ELOG.PKT,R3
000034  060103                                  ADD     R1,R3
000036  005004                                  CLR     R4
000040  005000                                  CLR     R0
000042  151300                                  BISB    (R3),R0
000044  020037  000104'                        CMP     R0,ICTLR
000050  001016                                  BNE     2$
000052  005204                                  INC     R4
000054  013700  000000G                        MOV     IPKT.ADDR,R0                    ;                               5615
000060  026160  000006G 000012                 CMP     ELOG.PKT+6(R1),12(R0)
000066  001007                                  BNE     2$
000070  026160  000010G 000014                 CMP     ELOG.PKT+10(R1),14(R0)          ;                               5616
000076  001003                                  BNE     2$
000100  126327  000001  000001                 CMPB    1(R3),#1                        ;                               5617
000106  001402                          2$:     BEQ     3$
000110  000137  027562'                        JMP     25$
000114  013746  000104'                3$:     MOV     ICTLR,-(SP)                     ;                               5621
000120  012746  000006                          MOV     #6,-(SP)
000124  004737  000000G                        JSR     PC,BL$MUL
000130  022626                                  CMP     (SP)+,(SP)+
000132  005760  000116'                        TST     LAST.PKT(R0)
000136  001161                                  BNE     23$
000140  126127  000016G 000004                 CMPB    ELOG.PKT+16(R1),#4              ;                               5624
000146  101142                                  BHI     21$
000150  010246                                  MOV     R2,-(SP)                        ; INDEX,*                       5627
000152  004737  000000V                        JSR     PC,SOFT.ERROR
000156  013766  000000G 000002                 MOV     L$LUN,2(SP)                     ; *,TEMP.UNIT                   5628
000164  012703  000006                          MOV     #6,R3                           ; *,OFFSET                      5630
000170  010300                          4$:     MOV     R3,R0                           ; OFFSET,*                      5632
000172  063700  000076'                        ADD     ICST.ADDR,R0
000176  016146  000012G                        MOV     ELOG.PKT+12(R1),-(SP)
000202  111004                                  MOVB    (R0),R4
000204  042704  177760                        BIC     #177760,R4
000210  020426                                  CMP     R4,(SP)+
000212  001012                                  BNE     5$
000214  032710  040000                        BIT     #40000,(R0)                     ;                               5633
000220  001407                                  BEQ     5$
000222  011004                                  MOV     (R0),R4                         ;                               5636
000224  000304                                  SWAB    R4
000226  042704  177760                        BIC     #177760,R4
000232  010437  000000G                        MOV     R4,L$LUN
000236  000405                                  BR      6$                              ;                               5635
000240  062703  000024                  5$:     ADD     #24,R3                          ; *,OFFSET                      5630
```

```
000244  020327  000102                      CMP     R3,#102                 ; OFFSET,*
000250  003747                              BLE     4$
000252  005000                      6$:     CLR     R0                      ;                       5643
000254  153700  001162'                      BISB    APT.MODE,R0
000260  116101  000016G                      MOVB    ELOG.PKT+16(R1),R1      ;                       5640
000264  042701  177400                      BIC     #177400,R1
000270  006301                              ASL     R1
000272  066107  000000'                      ADD     P.AAA(R1),PC            ; Case dispatch
000276  032700  000001              8$:     BIT     #1,R0                   ;                       5643
000302  001403                              BEQ     9$
000304  012716  000062                      MOV     #62,(SP)                ;                       5645
000310  000442                              BR      17$
000312  012716  000062              9$:     MOV     #62,(SP)                ;                       5647
000316  000446                              BR      19$
000320  032700  000001              10$:    BIT     #1,R0                   ;                       5649
000324  001403                              BEQ     11$
000326  012716  000063                      MOV     #63,(SP)                ;                       5651
000332  000431                              BR      17$
000334  012716  000063              11$:    MOV     #63,(SP)                ;                       5653
000340  000435                              BR      19$
000342  032700  000001              12$:    BIT     #1,R0                   ;                       5655
000346  001403                              BEQ     13$
000350  012716  000064                      MOV     #64,(SP)                ;                       5657
000354  000420                              BR      17$
000356  012716  000064              13$:    MOV     #64,(SP)                ;                       5659
000362  000424                              BR      19$
000364  032700  000001              14$:    BIT     #1,R0                   ;                       5661
000370  001403                              BEQ     15$
000372  012716  000065                      MOV     #65,(SP)                ;                       5663
000376  000407                              BR      17$
000400  012716  000065              15$:    MOV     #65,(SP)                ;                       5665
000404  000413                              BR      19$
000406  006000              16$:    ROR     R0                      ;                       5667
000410  103007                              BCC     18$
000412  012716  000066                      MOV     #66,(SP)                ;                       5669
000416  010246              17$:    MOV     R2,-(SP)                ; INDEX,*
000420  004737  000000V                      JSR     PC,ERR.SOFT.RTNE.APT
000424  005726                              TST     (SP)+
000426  000404                              BR      20$
000430  012716  000066              18$:    MOV     #66,(SP)                ;                       5667
000434  004737  000000V              19$:    JSR     PC,ERR.SOFT.RTNE        ;                       5671
000440  016637  000002  000000G     20$:    MOV     2(SP),L$LUN             ; TEMP.UNIT,*           5674
000446  112705  000001                      MOVB    #1,R5                   ; *,SFT.ERR.PRINTED     5675
000452  000412                              BR      22$                     ;                       5624
000454  005046              21$:    CLR     -(SP)                   ;                       5678
000456  116116  000016G                      MOVB    ELOG.PKT+16(R1),(SP)
000462  012746  000000G                      MOV     #DBM109,-(SP)
000466  012746  000002                      MOV     #2,-(SP)
000472  010600                              MOV     SP,R0                   ; SP,*
000474  104417                              TRAP    17
000476  022626                              CMP     (SP)+,(SP)+
000500  005726              22$:    TST     (SP)+                   ;                       5624
000502  032705  000001              23$:    BIT     #1,R5                   ; *,SFT.ERR.PRINTED     5680
```

```
000506  001007                          BNE     24$
000510  012746  000000G                 MOV     #CRLF,(SP)              ;                               5682
000514  012746  000001                  MOV     #1,-(SP)
000520  010600                          MOV     SP,R0                   ; SP,*
000522  104414                          TRAP    14
000524  022626                          CMP     (SP)+,(SP)+
000526  010246                  24$:    MOV     R2,-(SP)                ; INDEX,*                       5684
000530  004737  000000G                 JSR     PC,EMS.EL
000534  005726                          TST     (SP)+                   ;                               5619
000536  000433                          BR      27$                     ;                               5614
000540  006004                  25$:    ROR     R4                      ;                               5688
000542  103031                          BCC     27$
000544  013700  000000G                 MOV     IPKT.ADDR,R0            ;                               5689
000550  026160  000010G 000014          CMP     ELOG.PKT+10(R1),14(R0)
000556  103405                          BLO     26$
000560  001022                          BNE     27$                     ;                               5690
000562  026160  000006G 000012          CMP     ELOG.PKT+6(R1),12(R0)   ;                               5691
000570  103016                          BHIS    27$
000572  126327  000001  000001  26$:    CMPB    1(R3),#1                ;                               5692
000600  001012                          BNE     27$
000602  012746  000000G                 MOV     #CRLF,-(SP)             ;                               5695
000606  012746  000001                  MOV     #1,-(SP)
000612  010600                          MOV     SP,R0                   ; SP,*
000614  104414                          TRAP    14
000616  010216                          MOV     R2,(SP)                 ; INDEX,*                       5696
000620  004737  000000G                 JSR     PC,EMS.EL
000624  022626                          CMP     (SP)+,(SP)+             ;                               5694
000626  005202                  27$:    INC     R2                      ; INDEX                         5611
000630  020227  000014                  CMP     R2,#14                  ; INDEX,*
000634  003002                          BGT     28$
000636  000137  027034'                 JMP     1$
000642  005726                  28$:    TST     (SP)+                   ;                               5599
000644  000207                          RTS     PC

; Routine Size:  211 words,    Routine Base:  $CODE$ + 27022
; Maximum stack depth per invocation:  12 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:
000000  000000                  7$:     .WORD   0                       ; CASE Table for SCAN.ERRLOG+0272       5640
000002  000022                          .WORD   22                      ; [8$]
000004  000044                          .WORD   44                      ; [10$]
000006  000066                          .WORD   66                      ; [12$]
000010  000110                          .WORD   110                     ; [14$]
                                        .WORD   110                     ; [16$]
```

```
;    5702  1     GLOBAL routine DATAGM : novalue =
;    5703  1
;    5704  1     !+
;    5705  1     !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;    5706  1     !       THE RDRX
;    5707  1     !
;    5708  1     !       IMPLICIT INPUTS:
;    5709  1     !               IPKT_ADDR - ADDRESS OF MSCP PACKET CONTAINING ERROR LOG
;    5710  1     !                           MESSAGE
;    5711  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;    5712  1     !-
;    5713  1
;    5714  2         begin
;    5715  2
;    5716  2         local
;    5717  2             index : signed word initial (-1),
;    5718  2             SAVE_ADDR : ref block [EP_LEN, word] field (EP_FIELDS),
;    5719  2             SRC_ADDR,
;    5720  2             DST_ADDR,
;    5721  2             TEMP_UNIT,
;    5722  2             SFT_ERR_PRINTED : byte initial (byte (FALSE)),
;    5723  2             PACKET_LEN : word;
;    5724  2
;    5725  2         !
;    5726  2         ! FIND AN EMPTY SLOT IN THE ERROR-LOG PACKET SAVE AREA
;    5727  2         !
;    5728  2
;    5729  2         incr COUNT from 0 to EP_CNT - 1 do
;    5730  2
;    5731  2             if .ELOG_PKT [.COUNT, EL_CONTENTS] eql EMPTY                    ! IF EMPTY SLOT FOUND
;    5732  2             then
;    5733  3                 begin
;    5734  3                 index = .COUNT;                                             ! SAVE INDEX INTO THE SAVE AREA
;    5735  3                 exitloop;
;    5736  2                 end;
;    5737  2
;    5738  2             if .index lss 0
;    5739  2             then
;    5740  2                 index = EP_CNT;                                             ! IF NO SLOT FOUND, USE LAST SPARE SLOT
;    5741  2
;    5742  2         !
;    5743  2         ! SAVE THE PACKET CONTENTS
;    5744  2         !
;    5745  2
;    5746  2             SAVE_ADDR = ELOG_PKT + (.index * EP_LEN * 2);               ! ADDRESS OF THE SAVE AREA
;    5747  2             SAVE_ADDR [EL_CONTENTS ] = FULL;                            ! MARK IT FULL
;    5748  2             SAVE_ADDR [EL_CNTR] = .ICTLR;                               ! OWNERSHIP
;    5749  2             SRC_ADDR = .IPKT_ADDR + 6;                                  ! SETUP COPY ADDRESSES
;    5750  2             DST_ADDR = .SAVE_ADDR + 2;                                  !
;    5751  2             PACKET_LEN = ((.IPKT_ADDR [MSCP_LEN] + 1) / 2) + 2;         ! LENGTH OF ERROR-LOG INCLUDING ENVELOPE
;    5752  2
;    5753  2             if .PACKET_LEN gtru EP_LEN - 1
;    5754  2             then
```

```
;  5755  2              PACKET_LEN = ZP_LEN   1;                                ! ADJUST LENGTH, IF TOO LONG
;  5756  2
;  5757  2          incr COUNT from 1 to .PACKET_LEN do
;  5758  3              begin
;  5759  3              .DST_ADDR = ..SRC_ADDR;                                 ! COPY A WORD
;  5760  3              SRC_ADDR = .SRC_ADDR + 2;                               ! UPDATE ADDRESS POINTERS
;  5761  3              DST_ADDR = .DST_ADDR + 2;                               !
;  5762  3              end;
;  5763  2
;  5764  2          !
;  5765  2          ! CHECK IF THE CORRESPONDING RESPONSE HAS ALREADY BEEN RECEIVED
;  5766  2          !
;  5767  2
;  5768  2          if (.SAVE_ADDR [EL_CRN_LO] eql .LAST_PKT [.ICTLR, LAST_CRN_LO]) and
;  5769  3             (.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI])
;  5770  2          then
;  5771  3              begin                                                   ! LOG REFERS TO THE LAST RESPONSE RECEIVED
;  5772  3
;  5773  3              if .LAST_PKT [.ICTLR, LAST_HRD_ERR] eql HRD_NOT_OCCURED ! IF SOFT ERROR HAD OCCURED
;  5774  3              then
;  5775  3
;  5776  3                  if .SAVE_ADDR [EL_FORMAT] lequ 4
;  5777  3                  then
;  5778  4                      begin
;  5779  4                      SOFT_ERROR (.index);                            ! UPDATE SOFT ERROR COUNT
;  5780  4                      TEMP_UNIT = .L$LUN;                             ! SAVE UNIT NUMBER AS KNOWN TO DRS
;  5781  4
;  5782  4                  incr OFFSET from (0 + OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE + OF_UN) by UNIT_SIZE do
;  5783  4
;  5784  4                      if (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .SAVE_ADDR [EL_DK_NUM]) and
;  5785  5                         (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT)
;  5786  4                      then
;  5787  5                          begin
;  5788  5                          L$LUN = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT]; ! CORRECT UNIT NUMBER FOR ERROR MESSAGE
;  5789  5                          exitloop;
;  5790  4                          end;
;  5791  4
;  5792  4                  case .SAVE_ADDR [EL_FORMAT] from 0 to 4 of
;  5793  4                      set
;  5794  4
;  5795  4
;  5796  4                      [0] :       if .APT_MODE                        ! CONTROLLER ERROR
;  5797  4                                  then
;  5798  4                                      ERR_SOFT_RTNE_APT (50, .index)
;  5799  4                                  else
;  5800  4                                      ERR_SOFT_RTNE (50);
;  5801  4
;  5802  4                      [1] :       if .APT_MODE                        ! HOST MEMORY ACCESS ERROR
;  5803  4                                  then
;  5804  4                                      ERR_SOFT_RTNE_APT (51, .index)
;  5805  4                                  else
;  5806  4                                      ERR_SOFT_RTNE (51);
;  5807  4
```

```
;    5808  4                      [2] :       if .APT_MODE                                    ! DISK TRANSFER ERROR
;    5809  4                                  then
;    5810  4                                      ERR_SOFT_RTNE_APT (52, .index)
;    5811  4                                  else
;    5812  4                                      ERR_SOFT_RTNE (52);
;    5813  4
;    5814  4                      [3] :       if .APT_MODE                                    ! SDI ERROR
;    5815  4                                  then
;    5816  4                                      ERR_SOFT_RTNE_APT (53, .index)
;    5817  4                                  else
;    5818  4                                      ERR_SOFT_RTNE (53);
;    5819  4
;    5820  4                      [4] :       if .APT_MODE                                    ! SMALL DISK ERROR
;    5821  4                                  then
;    5822  4                                      ERR_SOFT_RTNE_APT (54, .index)
;    5823  4                                  else
;    5824  4                                      ERR_SOFT_RTNE (54);
;    5825  4                      tes;
;    5826  4
;    5827  4
;    5828  4                  L$LUN = .TEMP_UNIT;                                             ! RESTORE UNIT NUMBER
;    5829  4                  SFT_ERR_PRINTED = TRUE;                                         ! SOFT ERROR PRINTOUT OCCURED
;    5830  4                  end
;    5831  4
;    5832  3              else
;    5833  3                  PRINTF (DBM109, .SAVE_ADDR [EL_FORMAT]);                        ! ERROR LOG FORMAT UNKNOWN
;    5834  3
;    5835  4                  if not (.SFT_ERR_PRINTED)
;    5836  4
;    5837  3                  then
;    5838  3                      PRINTB (CRLF);                                              ! EXTRA CARRIEGE-RETURN/LINE-FEED
;    5839  3
;    5840  3                  EMS_EL (.index);                                                ! PRINT PACKET CONTENTS
;    5841  3                  end                                                             ! CORRESPONDING RESPONSE RECEIVED
;    5842  3
;    5843  2          else
;    5844  2
;    5845  2              if (.SAVE_ADDR [EL_CRN_HI] lssu .LAST_PKT [.ICTLR, LAST_CRN_HI]) or
;    5846  3                 ((.SAVE_ADDR [EL_CRN_HI] eql .LAST_PKT [.ICTLR, LAST_CRN_HI]) and
;    5847  3                  (.SAVE_ADDR [EL_CRN_LO] lssu .LAST_PKT [.ICTLR, LAST_CRN_LO]))
;    5848  3
;    5849  2              then
;    5850  3                  begin                                                           ! LOG REFERS TO SOME PREVIOUS RESPONSE
;    5851  3                  PRINTB (CRLF);                                                  ! CARRIAGE-RETURN/LINE-FEED
;    5852  3                  EMS_EL (.index);                                                ! PRINT PACKET CONTENTS
;    5853  2                  end;
;    5854  2
;    5855  1          end;


027670                                  .SBTTL  DATAGM RDRx INTERRUPT SERVICE ROUTINES
                                        .PSECT  $CODE$,  RO
```

```
000000  004137  000000G            DATAGM::JSR     R1,$SAVE5                       ;                                       5702
000004  012704  177777                     MOV     #-1,R4                          ; *,INDEX                               5714
000010  105046                       CLRB    -(SP)                           ; SFT.ERR.PRINTED
000012  005001                       CLR     R1                              ; COUNT                                 5729
000014  010146                 1$:   MOV     R1,-(SP)                        ; COUNT,*                               5731
000016  012746  000102               MOV     #102,-(SP)
000022  004737  000000G              JSR     PC,BL$MUL
000026  022626                       CMP     (SP)+,(SP)+
000030  105760  000001G              TSTB    ELOG.PKT+1(R0)
000034  001002                       BNE     2$
000036  010104                       MOV     R1,R4                           ; COUNT,INDEX                           5734
000040  000405                       BR      3$                              ;                                       5733
000042  005201                 2$:   INC     R1                              ; COUNT                                 5729
000044  020127  000013               CMP     R1,#13                          ; COUNT,*
000050  003761                       BLE     1$
000052  005704                       TST     R4                              ; INDEX                                 5738
000054  002002                 3$:   BGE     4$
000056  012704  000014               MOV     #14,R4                          ; *,INDEX                               5740
000062  010446                 4$:   MOV     R4,-(SP)                        ; INDEX,*                               5746
000064  012746  000102               MOV     #102,-(SP)
000070  004737  000000G              JSR     PC,BL$MUL
000074  062700  000000G              ADD     #ELOG.PKT,R0
000100  010001                       MOV     R0,R1                           ; *,SAVE.ADDR
000102  112761  000001               MOVB    (PC),1(R1)                      ; *,*(SAVE.ADDR)                        5747
000106  113711  000104'              MOVB    ICTLR,(R1)                      ; *,SAVE.ADDR                           5748
000112  013700  000000G              MOV     IPKT.ADDR,R0                    ;                                       5749
000116  012705  000006               MOV     #6,R5                           ; *,SRC.ADDR
000122  060005                       ADD     R0,R5                           ; *,SRC.ADDR
000124  012703  000002               MOV     #2,R3                           ; *,DST.ADDR                            5750
000130  060103                       ADD     R1,R3                           ; SAVE.ADDR,DST.ADDR
000132  016016  000006               MOV     6(R0),(SP)                      ;                                       5751
000136  005216                       INC     (SP)
000140  012746  000002               MOV     #2,-(SP)
000144  004737  000000G              JSR     PC,BL$DIV
000150  062700  000002               ADD     #2,R0
000154  020027  000040               CMP     R0,#40                          ; PACKET.LEN,*                          5753
000160  101402                       BLOS    5$
000162  012700  000040               MOV     #40,R0                          ; *,PACKET.LEN                          5755
000166  005002                 5$:   CLR     R2                              ; COUNT                                 5757
000170  000401                       BR      7$
000172  012523                 6$:   MOV     (R5)+,(R3)+                     ; SRC.ADDR,DST.ADDR                     5759
000174  005202                 7$:   INC     R2                              ; COUNT                                 5757
000176  020200                       CMP     R2,R0                           ; COUNT,PACKET.LEN
000200  003774                       BLE     6$
000202  013716  000104'              MOV     ICTLR,(SP)                      ;                                       5768
000206  012746  000006               MOV     #6,-(SP)
000212  004737  000000G              JSR     PC,BL$MUL
000216  005726                       TST     (SP)+
000220  026160  000006  000120'       CMP     6(R1),LAST.PKT+2(R0)            ; *(SAVE.ADDR),*
000226  001402                       BEQ     8$
000230  000137  030532'              JMP     30$
000234  026160  000010  000122'  8$:  CMP     10(R1),LAST.PKT+4(R0)           ; *(SAVE.ADDR),*                        5769
000242  001402                       BEQ     9$
```

```
000244  000137  030540'                    JMP     31$
000250  005760  000116'            9$:     TST     LAST.PKT(R0)                ;                                5773
000254  001153                              BNE     28$
000256  005003                              CLR     R3                          ;                                5776
000260  156103  000016                      BISB    16(R1),R3                   ; *(SAVE.ADDR),*
000264  020327  000004                      CMP     R3,#4
000270  101135                              BHI     27$
000272  010416                              MOV     R4,(SP)                     ; INDEX,*                        5779
000274  004737  000000V                     JSR     PC,SOFT.ERROR
000300  013705  000000G                     MOV     L$LUN,R5                    ; *,TEMP.UNIT                    5780
000304  012702  000006                      MOV     #6,R2                       ; *,OFFSET                       5782
000310  010200                      10$:    MOV     R2,R0                       ; OFFSET,*                       5784
000312  063700  000076'                     ADD     ICST.ADDR,R0
000316  016146  000012                      MOV     12(R1),-(SP)                ; *(SAVE.ADDR),*
000322  111046                              MOVB    (R0),-(SP)
000324  042716  177760                      BIC     #177760,(SP)
000330  022626                              CMP     (SP)+,(SP)+
000332  001012                              BNE     11$
000334  032710  040000                      BIT     #40000,(R0)                 ;                                5785
000340  001407                              BEQ     11$
000342  011046                              MOV     (R0),-(SP)                  ;                                5788
000344  000316                              SWAB    (SP)
000346  042716  177760                      BIC     #177760,(SP)
000352  012637  000000G                     MOV     (SP)+,L$LUN
000356  000405                              BR      12$                         ;                                5787
000360  062702  000024            11$:    ADD     #24,R2                      ; *,OFFSET                       5782
000364  020227  000102                      CMP     R2,#102                     ; OFFSET,*
000370  003747                              BLE     10$
000372  005000                      12$:    CLR     R0                          ;                                5796
000374  153700  001162'                     BISB    APT.MODE,R0
000400  006303                              ASL     R3                          ;                                5792
000402  066307  000012'                     ADD     P.AAB(R3),PC                ; Case dispatch
000406  032700  000001            14$:    BIT     #1,R0                       ;                                5796
000412  001403                              BEQ     15$
000414  012716  000062                      MOV     #62,(SP)                    ;                                5798
000420  000442                              BR      23$
000422  012716  000062            15$:    MOV     #62,(SP)                    ;                                5800
000426  000446                              BR      25$
000430  032700  000000            16$:    BIT     #1,R0                       ;                                5802
000434  001403                              BEQ     17$
000436  012716  000063                      MOV     #63,(SP)                    ;                                5804
000442  000431                              BR      23$
000444  012716  000063            17$:    MOV     #63,(SP)                    ;                                5806
000450  000435                              BR      25$
000452  032700  000001            18$:    BIT     #1,R0                       ;                                5808
000456  001403                              BEQ     19$
000460  012716  000064                      MOV     #64,(SP)                    ;                                5810
000464  000420                              BR      23$
000466  012716  000064            19$:    MOV     #64,(SP)                    ;                                5812
000472  000424                              BR      25$
000474  032700  000001            20$:    BIT     #1,R0                       ;                                5814
000500  001403                              BEQ     21$
000502  012716  000065                      MOV     #65,(SP)                    ;                                5816
```

```
000506  000407                                  BR      23$
000510  012716  000065                 21$:    MOV     #65,(SP)                ;                               5818
000514  000413                                  BR      25$
000516  006000                 22$:    ROR     R0                              ;                               5820
000520  103007                                  BCC     24$
000522  012716  000066                         MOV     #66,(SP)                ;                               5822
000526  010446                 23$:    MOV     R4,-(SP)                        ; INDEX,*
000530  004737  000000V                         JSR     PC,ERR.SOFT.RTNE.APT
000534  005726                                  TST     (SP)+
000536  000404                                  BR      26$                     ;                               5820
000540  012716  000066                 24$:    MOV     #66,(SP)                ;                               5824
000544  004737  000000V                 25$:    JSR     PC,ERR.SOFT.RTNE
000550  010537  000000G                 26$:    MOV     R5,L$LUN                ; TEMP.UNIT,*                   5828
000554  112766  000001  000006                 MOVB    #1,6(SP)                ; *,SFT.ERR.PRINTED             5829
000562  000410                                  BR      28$                     ;                               5776
000564  010316                 27$:    MOV     R3,(SP)                         ;                               5833
000566  012746  000000G                         MOV     #DBM109,-(SP)
000572  012746  000002                         MOV     #2,-(SP)
000576  010600                                  MOV     SP,R0                   ; SP,*
000600  104417                                  TRAP    17
000602  022626                                  CMP     (SP)+,(SP)+
000604  032766  000001  000006         28$:    BIT     #1,6(SP)                ; *,SFT.ERR.PRINTED             5835
000612  001007                                  BNE     29$
000614  012716  000000G                         MOV     #CRLF,(SP)              ;                               5838
000620  012746  000001                         MOV     #1,-(SP)
000624  010600                                  MOV     SP,R0                   ; SP,*
000626  104414                                  TRAP    14
000630  005726                                  TST     (SP)+
000632  010416                 29$:    MOV     R4,(SP)                         ; INDEX,*                       5840
000634  004737  000000G                         JSR     PC,EMS.EL
000640  000426                                  BR      33$                     ;                               5768
000642  026160  000010  000122'         30$:    CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*                5845
000650  103410                 31$:    BLO     32$                             ;                               5846
000652  026160  000010  000122'                 CMP     10(R1),LAST.PKT+4(R0)   ; *(SAVE.ADDR),*
000660  001016                                  BNE     33$                     ;                               5847
000662  026160  000006  000120'                 CMP     6(R1),LAST.PKT+2(R0)    ; *(SAVE.ADDR),*
000670  103012                                  BHIS    33$                     ;                               5851
000672  012716  000000G                 32$:    MOV     #CRLF,(SP)              ;
000676  012746  000001                         MOV     #1,(SP)
000702  010600                                  MOV     SP,R0                   ; SP,*
000704  104414                                  TRAP    14
000706  010416                                  MOV     R4,(SP)                 ; INDEX,*                       5852
000710  004737  000000G                         JSR     PC,EMS.EL
000714  005726                                  TST     (SP)+                   ;                               5850
000716  062706  000010                 33$:    ADD     #10,SP                  ;                               5702
000722  000207                                  RTS     PC

; Routine Size:  234 words,    Routine Base:  $CODE$ + 27670
; Maximum stack depth per invocation:  14 words

000012                                          .PSECT  $PLIT$,  RO ,  D
```

```
                                P.AAB:
000012  000000                13$:    .WORD   0               ; CASE Table for DATAGM+0402            5792
000014  000022                        .WORD   22              ; [14$]
000016  000044                        .WORD   44              ; [16$]
000020  000066                        .WORD   66              ; [18$]
000022  000110                        .WORD   110             ; [20$]
                                                               ; [22$]


;    5856  1
;    5857  1
;    5858  1
```

```
;  5859  1     GLOBAL routine SOFT_ERROR (index) : novalue =
;  5860  1
;  5861  1     !+
;  5862  1     !       THIS ROUTINE UPDATES THE SOFT ERROR COUNT IN THE TALLY TABLE FOR EACH
;  5863  1     !       ERROR LOG MESSAGE RECEIVED
;  5864  1     !
;  5865  1     !       IMPLICIT INPUTS:
;  5866  1     !               ICST_ADDR - ADDRESS OF THE INTERRUPTING CONTROLLER'S CST
;  5867  1     !-
;  5868  1
;  5869  2         begin
;  5870  2
;  5871  2         local
;  5872  2             FOUND: byte initial (byte (FALSE)),
;  5873  2             SOFT_OCCURED : byte initial (byte (FALSE)),
;  5874  2             UNIT: word,
;  5875  2             ERROR_CODE : byte,
;  5876  2             ERROR_SUB : word,
;  5877  2             RETRIES : word,
;  5878  2             TALLY_ADDR : ref block [TALLY_LEN, word] field (T_FIELDS),
;  5879  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;  5880  2
;  5881  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);                          ! ADDR OF ERROR PKT
;  5882  2         ERROR_CODE = .ELOG_ADDR [EL_CODE];                                     ! ERROR CODE
;  5883  2         ERROR_SUB = .ELOG_ADDR [EL_SUBCODE];                                   ! ERROR SUBCODE
;  5884  2
;  5885  2         if (BIT_TST (SWP_FLAGS, SWF_TRY)) and
;  5886  3            (.ELOG_ADDR [EL_FORMAT] eql 2)
;  5887  2         then
;  5888  2             RETRIES = .ELOG_ADDR [EL_RETRY]                                    ! COUNT EACH RETRY
;  5889  2         else
;  5890  2             RETRIES = 1;                                                       ! IGNORE RETRIES
;  5891  2
;  5892  2         if .RETRIES eql 0                                                      ! IN CASE OF A BUG
;  5893  2         then
;  5894  2             RETRIES = 1;
;  5895  2
;  5896  2         incr OFFSET from (0 * OF_UN) to ((UNITS_PER_CNTR - 1) * UNIT_SIZE * OF_UN) by UNIT_SIZE do
;  5897  2
;  5898  2             if (.ICST_ADDR [.OFFSET + OF_DATA, D_PRES] eql PRESENT) and        ! DISK TO UNIT NO.
;  5899  3                (.ICST_ADDR [.OFFSET + OF_DATA, D_DISK_NUM] eql .ELOG_ADDR [EL_DK_NUM])
;  5900  2             then
;  5901  3                 begin
;  5902  3                 FOUND = TRUE;
;  5903  3                 UNIT = .ICST_ADDR [.OFFSET + OF_DATA, D_UNIT];             ! DISK'S UNIT NO.
;  5904  3                 exitloop;
;  5905  2                 end;
;  5906  2
;  5907  2     !   if (.ELOG_ADDR [EL_SUCCESS]) or
;  5908  2     !      (.ELOG_ADDR [EL_CONTINUE])
;  5909  2     !   then
;  5910  2             SOFT_OCCURED = TRUE;                                           ! SOFT ERROR FLAG
;  5911  2
```

```
;    5912  2          if .FOUND
;    5913  2          then                                                                    ! IF UNIT FOUND
;    5914  3              begin
;    5915  3              TALLY_ADDR = TALLY + (.UNIT * TALLY_LEN * 2);                       ! ADDR OF TALLY TBL
;    5916  3
;    5917  3              if .SOFT_OCCURED                                                    ! FOR SOFT ERRORRS
;    5918  3              then
;    5919  3                  selectoneu .ERROR_CODE of
;    5920  3                      set
;    5921  3
;    5922  3                      [ST_MFE]:       TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES;! SOFT-MEDIA FORMAT
;    5923  3
;    5924  3                      [ST_DAT]:       if .ERROR_SUB eql 2                         ! SOFT-DATA
;    5925  3                                      then
;    5926  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;    5927  3                                      else
;    5928  3                                          TALLY_ADDR [ERR_SFT_DAT] = .TALLY_ADDR [ERR_SFT_DAT] + .RETRIES;
;    5929  3
;    5930  3                      [ST_HST]:       TALLY_ADDR [ERR_SFT_HST] = .TALLY_ADDR [ERR_SFT_HST] + .RETRIES;! SOFT-HOST ACCESS
;    5931  3
;    5932  3                      [ST_CNT]:       C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + .RETRIES;
;    5933  3                                                                                  ! SOFT-CONTROLLER
;    5934  3
;    5935  3                      [ST_DRV]:       if .ERROR_SUB eql 3                         ! SOFT_DRIVE
;    5936  3                                      then
;    5937  3                                          TALLY_ADDR [ERR_SFT_SEK] = .TALLY_ADDR [ERR_SFT_SEK] + .RETRIES
;    5938  3                                      else
;    5939  3                                          TALLY_ADDR [ERR_SFT_DRV] = .TALLY_ADDR [ERR_SFT_DRV] + .RETRIES;
;    5940  3                      tes
;    5941  3              else
;    5942  3
;    5943  3                  if (.ELOG_ADDR [EL_CRN_LO] eql 0) and
;    5944  4                     (.ELOG_ADDR [EL_CRN_HI] eql 0)
;    5945  3                  then
;    5946  3                      selectoneu .ERROR_CODE of
;    5947  3                          set
;    5948  3
;    5949  3                          [ST_MFE]:   TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1;       ! HARD-MEDIA FORMAT
;    5950  3
;    5951  3                          [ST_DAT]:   if .ERROR_SUB eql 2                         ! HARD-DATA
;    5952  3                                      then
;    5953  3                                          TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
;    5954  3                                      else
;    5955  3                                          TALLY_ADDR [ERR_HRD_DAT] = .TALLY_ADDR [ERR_HRD_DAT] + 1;
;    5956  3
;    5957  3                          [ST_HST]:   TALLY_ADDR [ERR_HRD_HST] = .TALLY_ADDR [ERR_HRD_HST] + 1;       ! HARD-HOST ACCESS
;    5958  3
;    5959  3                          [ST_CNT]:   C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    5960  3                                                                                  ! HARD CONTROLLER
;    5961  3
;    5962  3                          [ST_DRV]:   if .ERROR_SUB eql 3                         ! HARD-DRIVE
;    5963  3                                      then
;    5964  3                                          TALLY_ADDR [ERR_HRD_SEK] = .TALLY_ADDR [ERR_HRD_SEK] + 1
```

```
;    5965  3                                    else
;    5966  3                                        TALLY_ADDR [ERR_HRD_DRV] = .TALLY_ADDR [ERR_HRD_DRV] + 1;
;    5967  3                            tes;
;    5968  3
;    5969  3            end
;    5970  2        else                                                                  ! UNIT NOT FOUND
;    5971  2
;    5972  2            if .SOFT_OCCURED
;    5973  2            then
;    5974  2                C_ERR_TBL [.ICTLR, C_ERR_SFT] = .C_ERR_TBL [.ICTLR, C_ERR_SFT] + 1
;    5975  2            else
;    5976  2                C_ERR_TBL [.ICTLR, C_ERR_HRD] = .C_ERR_TBL [.ICTLR, C_ERR_HRD] + 1;
;    5977  2
;    5978  1        end;                                                                  ! RTNE SOFT_ERROR

                                            .SBTTL  SOFT.ERROR RDRX INTERRUPT SERVICE ROUTINES
030614                                      .PSECT  $CODE$,  RO

000000  004137  000000G             SOFT.ERROR::
000004  005746                      JSR     R1,$SAVE5                 ;                                5859
000006  105046                      TST     -(SP)
000010  105046                      CLRB    -(SP)                     ; FOUND                          5869
000012  016646  000024              CLRB    -(SP)                     ; SOFT.OCCURED
000016  012746  000102              MOV     24(SP),-(SP)              ; INDEX,*                        5881
000022  004737  000000G             MOV     #102,-(SP)
000026  062700  000000G             JSR     PC,BL$MUL
000032  010001                      ADD     #ELOG.PKT,R0
000034  116100  000020              MOV     R0,R1
000040  042700  177740              MOVB    20(R1),R0                 ; *,ELOG.ADDR
000044  105003                      BIC     #177740,R0                ; *(ELOG.ADDR),*                 5882
000046  050003                      CLRB    R3                        ; ERROR.CODE
000050  016105  000020              BIS     R0,R3                     ; *,ERROR.CODE
000054  006205                      MOV     20(R1),R5                 ; *(ELOG.ADDR),ERROR.SUB         5883
000056  006205                      ASR     R5                        ; ERROR.SUB
000060  006205                      ASR     R5                        ; ERROR.SUB
000062  006205                      ASR     R5                        ; ERROR.SUB
000064  006205                      ASR     R5                        ; ERROR.SUB
000066  042705  174000              ASR     R5                        ; ERROR.SUB
000072  013700  000000G             BIC     #174000,R5                ; *,ERROR.SUB
000076  042700  077777              MOV     SWP.FLAGS,R0              ;                                5885
000102  020027  100000              BIC     #077777,R0
000106  001010                      CMP     R0,#-100000
000110  126127  000016  000002      BNE     1$
000116  001004                      CMPB    16(R1),#2                 ; *(ELOG.ADDR),*                 5886
000120  005004                      BNE     1$
000122  156104  000051              CLR     R4                        ; RETRIES                        5888
000126  000402                      BISB    51(R1),R4                 ; *(ELOG.ADDR),RETRIES
000130  012704  000001      1$:     BR      2$                        ;                                5885
000134  005704              2$:     MOV     #1,R4                     ; *,RETRIES                      5890
000136  001002                      TST     R4                        ; RETRIES                        5892
000140  012704  000001              BNE     3$
                                    MOV     #1,R4                     ; *,RETRIES                      5894
```

```
000144  012702  000006              3$:     MOV     #6,R2                   ; *,OFFSET                      5896
000150  010200                      4$:     MOV     R2,R0                   ; OFFSET,*                      5898
000152  063700  000076'                      ADD     ICST.ADDR,R0
000156  032710  040000                      BIT     #40000,(R0)
000162  001421                              BEQ     5$
000164  016146  000012                      MOV     12(R1),-(SP)            ; *(ELOG.ADDR),*                5899
000170  111046                              MOVB    (R0),-(SP)
000172  042716  177760                      BIC     #177760,(SP)
000176  022626                              CMP     (SP)+,(SP)+
000200  001012                              BNE     5$
000202  112766  000001  000006              MOVB    #1,6(SP)                ; *,FOUND                       5902
000210  011046                              MOV     (R0),-(SP)              ;                               5903
000212  000316                              SWAB    (SP)
000214  042716  177760                      BIC     #177760,(SP)
000220  012666  000010                      MOV     (SP)+,10(SP)            ; *,UNIT
000224  000405                              BR      6$                      ;                               5901
000226  062702  000024              5$:     ADD     #24,R2                  ; *,OFFSET                      5896
000232  020227  000102                      CMP     R2,#102                 ; OFFSET,*
000236  003744                              BLE     4$
000240  112766  000001  000004      6$:     MOVB    #1,4(SP)                ; *,SOFT.OCCURED                5910
000246  032766  000001  000006              BIT     #1,6(SP)                ; *,FOUND                       5912
000254  001002                              BNE     7$
000256  000137  031502'                      JMP     22$
000262  016616  000010              7$:     MOV     10(SP),(SP)             ; UNIT,*                        5915
000266  012746  000066                      MOV     #66,-(SP)
000272  004737  000000G                      JSR     PC,BL$MUL
000276  062700  000000G                      ADD     #TALLY,R0
000302  032766  000001  000006              BIT     #1,6(SP)                ; *,SOFT.OCCURED                5917
000310  001503                              BEQ     14$
000312  120327  000005                      CMPB    R3,#5                   ; ERROR.CODE,*                  5922
000316  001462                              BEQ     12$
000320  120327  000010                      CMPB    R3,#10                  ; ERROR.CODE,*                  5924
000324  001022                              BNE     9$
000326  012702  000052                      MOV     #52,R2                  ;                               5926
000332  060002                              ADD     R0,R2                   ; TALLY.ADDR,*
000334  020527  000002                      CMP     R5,#2                   ; ERROR.SUB,*                   5924
000340  001005                              BNE     8$
000342  005001                              CLR     R1                      ;                               5926
000344  151201                              BISB    (R2),R1
000346  060401                              ADD     R4,R1                   ; RETRIES,*
000350  110112                              MOVB    R1,(R2)
000352  000543                              BR      21$                     ;                               5924
000354  005001              8$:     CLR     R1                      ;                               5928
000356  156201  000001                      BISB    1(R2),R1
000362  060401                              ADD     R4,R1                   ; RETRIES,*
000364  110162  000001                      MOVB    R1,1(R2)
000370  000534                              BR      21$                     ;                               5919
000372  120327  000011              9$:     CMPB    R3,#11                  ; ERROR.CODE,*                  5930
000376  001007                              BNE     10$
000400  005001                              CLR     R1
000402  156001  000055                      BISB    55(R0),R1               ; *(TALLY.ADDR),*
000406  060401                              ADD     R4,R1                   ; RETRIES,*
000410  110160  000055                      MOVB    R1,55(R0)               ; *,*(TALLY.ADDR)
```

```
000414  000522                          BR      21$                             ;                               5919
000416  120327  000012        10$:      CMPB    R3,#12                          ; ERROR.CODE,*                  5932
000422  001012                          BNE     11$
000424  013702  000104'                  MOV     ICTLR,R2
000430  006302                          ASL     R2
000432  005001                          CLR     R1
000434  156201  000001G                 BISB    C.ERR.TBL+1(R2),R1
000440  060401                          ADD     R4,R1                           ; RETRIES,*
000442  110162  000001G                 MOVB    R1,C.ERR.TBL+1(R2)
000446  000505                          BR      21$                             ;                               5919
000450  120327  000013        11$:      CMPB    R3,#13                          ; ERROR.CODE,*                  5935
000454  001102                          BNE     21$
000456  020527  000003                  CMP     R5,#3                           ; ERROR.SUB,*
000462  001007                          BNE     13$
000464  005001                  12$:    CLR     R1                              ;                               5937
000466  156001  000052                  BISB    52(R0),R1                       ; *(TALLY.ADDR),*
000472  060401                          ADD     R4,R1                           ; RETRIES,*
000474  110160  000052                  MOVB    R1,52(R0)                       ; *,*(TALLY.ADDR)
000500  000470                          BR      21$                             ;                               5935
000502  005001                  13$:    CLR     R1                              ;                               5939
000504  156001  000054                  BISB    54(R0),R1                       ; *(TALLY.ADDR),*
000510  060401                          ADD     R4,R1                           ; RETRIES,*
000512  110160  000054                  MOVB    R1,54(R0)                       ; *,*(TALLY.ADDR)
000516  000461                          BR      21$                             ;                               5919
000520  005761  000006        14$:      TST     6(R1)                           ; *(ELOG.ADDR)                  5943
000524  001056                          BNE     21$
000526  005761  000010                  TST     10(R1)                          ; *(ELOG.ADDR)                  5944
000532  001053                          BNE     21$
000534  120327  000005                  CMPB    R3,#5                           ; ERROR.CODE,*                  5949
000540  001443                          BEQ     19$
000542  120327  000010                  CMPB    R3,#10                          ; ERROR.CODE,*                  5951
000546  001013                          BNE     16$
000550  012704  000046                  MOV     #46,R4                          ;                               5953
000554  060004                          ADD     R0,R4                           ; TALLY.ADDR,*
000556  020527  000002                  CMP     R5,#2                           ; ERROR.SUB,*                   5951
000562  001002                          BNE     15$
000564  105214                          INCB    (R4)                            ;                               5953
000566  000435                          BR      21$                             ;                               5951
000570  105264  000001        15$:      INCB    1(R4)                           ;                               5955
000574  000432                          BR      21$                             ;                               5946
000576  120327  000011        16$:      CMPB    R3,#11                          ; ERROR.CODE,*                  5957
000602  001003                          BNE     17$
000604  105260  000051                  INCB    51(R0)                          ; *(TALLY.ADDR)
000610  000424                          BR      21$                             ;                               5946
000612  120327  000012        17$:      CMPB    R3,#12                          ; ERROR.CODE,*                  5959
000616  001006                          BNE     18$
000620  013702  000104'                  MOV     ICTLR,R2
000624  006302                          ASL     R2
000626  105262  0000006                 INCB    C.ERR.TBL(R2)
000632  000413                          BR      21$                             ;                               5946
000634  120327  000013        18$:      CMPB    R3,#13                          ; ERROR.CODE,*                  5960
000640  001010                          BNE     21$
000642  020527  000003                  CMP     R5,#3                           ; ERROR.SUB,*
```

```
000646  001003                          BNE     20$
000650  105260  000046         19$:     INCB    46(R0)          ; *(TALLY.ADDR)                 5964
000654  000402                          BR      21$             ;                               5962
000656  105260  000050         20$:     INCB    50(R0)          ; *(TALLY.ADDR)                 5966
000662  005726                 21$:     TST     (SP)+           ;                               5914
000664  000415                          BR      24$             ;                               5912
000666  013700  000104'         22$:     MOV     ICTLR,R0        ;                               5974
000672  006300                          ASL     R0
000674  062700  000000G                 ADD     #C.ERR.TBL,R0
000700  032766  000001  000004          BIT     #1,4(SP)        ; *,SOFT.OCCURED                5972
000706  001403                          BEQ     23$
000710  105260  000001                  INCB    1(R0)           ;                               5974
000714  000401                          BR      24$             ;                               5970
000716  105210                 23$:     INCB    (R0)            ;                               5976
000720  062706  000012         24$:     ADD     #12,SP          ;                               5859
000724  000207                          RTS     PC

; Routine Size:  235 words,     Routine Base:  $CODE$ + 30614
; Maximum stack depth per invocation:  13 words
```

```
;    5979  1      routine ERR_HRD_RTNE (ERRNUM) : novalue =
;    5980  1
;    5981  1      !+
;    5982  1      ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    5983  1      ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    5984  1      !-
;    5985  1
;    5986  2          begin
;    5987  2
;    5988  2          local
;    5989  2              CUR_PRIORITY : word;
;    5990  2
;    5991  2          builtin
;    5992  2              PC;
;    5993  2
;    5994  2          GETPRI (CUR_PRIORITY);
;    5995  2          SETPRI (PRI04);                                 ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    5996  2
;    5997  2          if (.ERRNUM lequ 34) or                         ! FOR NON-BAD BLOCK TYPE ERRORS
;    5998  2             (.ERRNUM gtru 38) or
;    5999  2             (.ERRNUM eql 36) or
;    6000  3             (.ERRNUM eql 37)
;    6001  2          then
;    6002  2
;    6003  3          if BIT_TST (SWP_FLAGS, SWF_HRD)                 ! IF ERRORS TO BE TREATED NORMALLY
;    6004  2          then
;    6005  2
;    6006  2              case .ERRNUM from 31 to 45 of
;    6007  2                  set
;    6008  2
;    6009  2                  [31]:   ERRHRD (31, EGH_30, EMS_30);    ! INVALID COMMAND
;    6010  2
;    6011  2                  [32]:   ERRHRD (32, EGH_30, EMS_30);    ! COMMAND ABORTED
;    6012  2
;    6013  2                  [33]:   ;                               !
;    6014  2
;    6015  2                  [34]:   ;                               !
;    6016  2
;    6017  2                  [35]:   ;                               ! MEDIA FORMAT ERROR
;    6018  2
;    6019  2                  [36]:   ERRHRD (36, EGH_30, EMS_30);    ! WRITE PROTECTED
;    6020  2
;    6021  2                  [37]:   ERRHRD (37, EGH_30, EMS_30);    ! COMPARE ERROR
;    6022  2
;    6023  2                  [38]:   ;                               ! DATA ERROR
;    6024  2
;    6025  2                  [39]:   ERRHRD (39, EGH_30, EMS_30);    ! HOST BUFFER ACCESS ERROR
;    6026  2
;    6027  2                  [40]:   ERRHRD (40, EGH_30, EMS_30);    ! CONTROLLER ERROR
;    6028  2
;    6029  2                  [41]:   ERRHRD (41, EGH_30, EMS_30);    ! DRIVE ERROR
;    6030  2
;    6031  2                  [42]:   ERRHRD (42, EGH_30, 0);         ! HOST WRITE COMPARE ERROR
```

```
;  6032  2
;  6033  2                    [43]:   ERRHRD (43, EGH_30, EMS_30);    ! MESSAGE FROM INTERNAL DIAGNOSTICS
;  6034  2
;  6035  2                    [44]:   ERRHRD (44, EGH_30, EMS_30);    ! DUPLICATE UNIT NUMBER
;  6036  2
;  6037  2                    [45]:   ERRHRD (45, EGH_30, EMS_30);    ! INVALID END CODE
;  6038  2                    tes
;  6039  2                ese
;  6040  3                    begin
;  6041  3                    !****increment error count              ! INCREMENT TOTAL ERROR COUNT
;  6042  3                    PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC); ! PRINT ERROR MESSAGE JUST LIKE DRS
;  6043  3
;  6044  3                    if .ERRNUM neq 42
;  6045  3                    then
;  6046  4                        begin
;  6047  4                        PRINTB (HRD_SUB);                   ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;  6048  4                        EMS_ERR ();                         ! PRINT REST OF THE INFORMATION
;  6049  3                        end;
;  6050  2                    end;
;  6051  2
;  6052  2            if (.ERRNUM eql 35) or                          ! FOR BAD-BLOCK TYPE ERRORS
;  6053  3               (.ERRNUM eql 38)
;  6054  2            then
;  6055  2
;  6056  3                if BIT_TST (SWP_FLAGS, SWF_BLK)             ! IF ERRORS TO BE TREATED NORMALLY
;  6057  2                then
;  6058  2                    selectoneu .ERRNUM of
;  6059  2                        set
;  6060  2
;  6061  2                        [35]:   ERRHRD (35, EGH_30, EMS_30);    ! MEDIA FORMAT ERROR
;  6062  2
;  6063  2                        [38]:   ERRHRD (38, EGH_30, EMS_30);    ! DATA ERROR
;  6064  2                        tes
;  6065  2                ese
;  6066  2                    begin
;  6067  3                    !****increment error count              ! INCREMENT TOTAL ERROR COUNT
;  6068  3                    PRINTB (HRD_MSG, .ERRNUM, .L$LUN, .PC); ! PRINT ERROR LINE JUST LIKE DRS
;  6069  3                    PRINTB (HRD_SUB);                       ! PRINT NEXT LINE TOO
;  6070  3                    EMS_ERR ();                             ! PRINT REST OF THE INFORMATION
;  6071  3                    end;
;  6072  2
;  6073  2            SETPRI (.CUR_PRIORITY);                         ! PRIORITY BACK TO NORMAL
;  6074  2
;  6075  2
;  6076  1            end;


                                .SBTTL  ERR.HRD.RTNE RDRX INTERRUPT SERVICE ROUTINES
000000  004137  000000G    ERR.HRD.RTNE:
                            JSR     R1,$SAVE2                       ;                                   5979
000004  104440              TRAP    40                              ;                                   5994
000006  010002              MOV     R0,R2                           ; %,CUR.PRIORITY
000010  012700  000200      MOV     #200,R0                         ;                                   5995
```

```
000014  104441                          TRAP    41
000016  016601  000010                  MOV     10(SP),R1               ; ERRNUM,*              5997
000022  020127  000042                  CMP     R1,#42
000026  101411                          BLOS    1$
000030  020127  000046                  CMP     R1,#46                  ;                       5998
000034  101006                          BHI     1$
000036  020127  000044                  CMP     R1,#44                  ;                       5999
000042  001403                          BEQ     1$
000044  020127  000045                  CMP     R1,#45                  ;                       6000
000050  001131                          BNE     16$
000052  032737  010000  000000G  1$:    BIT     #10000,SWP.FLAGS        ;                       6003
000060  001475                          BEQ     14$
000062  010100                          MOV     R1,R0                   ;                       6006
000064  162700  000037                  SUB     #37,R0
000070  006300                          ASL     R0
000072  066007  000024'                  ADD     P.AAC(R0),PC            ; Case dispatch
000076  104456                  3$:     TRAP    56                      ;                       6009
000100  000037                          .WORD   37
000102  000000G                         .WORD   EGH.30
000104  000000G                         .WORD   EMS.30
000106  000512                          BR      16$
000110  104456                  4$:     TRAP    56                      ;                       6006
000112  000040                          .WORD   40                      ;                       6011
000114  000000G                         .WORD   EGH.30
000116  000000G                         .WORD   EMS.30
000120  000505                          BR      16$
000122  104456                  5$:     TRAP    56                      ;                       6006
000124  000044                          .WORD   44                      ;                       6019
000126  000000G                         .WORD   EGH.30
000130  000000G                         .WORD   EMS.30
000132  000500                          BR      16$
000134  104456                  6$:     TRAP    56                      ;                       6006
000136  000045                          .WORD   45                      ;                       6021
000140  000000G                         .WORD   EGH.30
000142  000000G                         .WORD   EMS.30
000144  000473                          BR      16$
000146  104456                  7$:     TRAP    56                      ;                       6006
000150  000047                          .WORD   47                      ;                       6025
000152  000000G                         .WORD   EGH.30
000154  000000G                         .WORD   EMS.30
000156  000466                          BR      16$
000160  104456                  8$:     TRAP    56                      ;                       6006
000162  000050                          .WORD   50                      ;                       6027
000164  000000G                         .WORD   EGH.30
000166  000000G                         .WORD   EMS.30
000170  000461                          BR      16$
000172  104456                  9$:     TRAP    56                      ;                       6006
000174  000051                          .WORD   51                      ;                       6029
000176  000000G                         .WORD   EGH.30
000200  000000G                         .WORD   EMS.30
000202  000454                          BR      16$
000204  104456                  10$:    TRAP    56                      ;                       6006
000206  000052                          .WORD   52                      ;                       6031
```

```
000210  000000G                         .WORD   EGH.30
000212  000000                          .WORD   0
000214  000447                          BR      16$                             ;                       6006
000216  104456                  11$:    TRAP    56                              ;                       6033
000220  000053                          .WORD   53
000222  000000G                         .WORD   EGH.30
000224  000000G                         .WORD   EMS.30
000226  000442                          BR      16$                             ;                       6006
000230  104456                  12$:    TRAP    56                              ;                       6035
000232  000054                          .WORD   54
000234  000000G                         .WORD   EGH.30
000236  000000G                         .WORD   EMS.30
000240  000435                          BR      16$                             ;                       6006
000242  104456                  13$:    TRAP    56                              ;                       6037
000244  000055                          .WORD   55
000246  000000G                         .WORD   EGH.30
000250  000000G                         .WORD   EMS.30
000252  000430                          BR      16$                             ;                       6003
000254  010746                  14$:    MOV     PC,-(SP)                        ; PC,*                  6042
000256  013746  000000G                 MOV     L$LUN,-(SP)
000262  010146                          MOV     R1,-(SP)
000264  012746  000000G                 MOV     #HRD.MSG,-(SP)
000270  012746  000004                  MOV     #4,-(SP)
000274  010600                          MOV     SP,R0                           ; SP,*
000276  104414                          TRAP    14
000300  020127  000052                  CMP     R1,#52                          ;                       6044
000304  001411                          BEQ     15$
000306  012716  000000G                 MOV     #HRD.SUB,(SP)                   ;                       6047
000312  012746  000001                  MOV     #1,-(SP)
000316  010600                          MOV     SP,R0                           ; SP,*
000320  104414                          TRAP    14
000322  004737  000000G                 JSR     PC,EMS.ERR                      ;                       6048
000326  005726                          TST     (SP)+                           ;                       6046
000330  062706  000012          15$:    ADD     #12,SP                          ;                       6040
000334  020127  000043          16$:    CMP     R1,#43                          ;                       6052
000340  001403                          BEQ     17$
000342  020127  000046                  CMP     R1,#46                          ;                       6053
000346  001050                          BNE     20$
000350  032737  040000  000000G 17$:    BIT     #40000,SWP,FLAGS                ;                       6056
000356  001420                          BEQ     19$
000360  020127  000043                  CMP     R1,#43                          ;                       6062
000364  001005                          BNE     18$
000366  104456                          TRAP    56
000370  000043                          .WORD   43
000372  000000G                         .WORD   EGH.30
000374  000000G                         .WORD   EMS.30
000376  000434                          BR      20$                             ;                       6059
000400  020127  000046          18$:    CMP     R1,#46                          ;                       6064
000404  001031                          BNE     20$
000406  104456                          TRAP    56
000410  000046                          .WORD   46
000412  000000G                         .WORD   EGH.30
000414  000000G                         .WORD   EMS.30
```

```
000416  000424                          BR      20$                     ;
000420  010746                  19$:    MOV     PC,-(SP)                ; PC,*                  6056
000422  013746  000000G                 MOV     L$LUN, (SP)                                     6069
000426  010146                          MOV     R1,-(SP)
000430  012746  000000G                 MOV     #HRD.MSG,-(SP)
000434  012746  000004                  MOV     #4,-(SP)
000440  010600                          MOV     SP,R0                   ; SP,*
000442  104414                          TRAP    14
000444  012716  000000G                 MOV     #HRD.SUB,(SP)           ;                       6070
000450  012746  000001                  MOV     #1,-(SP)
000454  010600                          MOV     SP,R0                   ; SP,*
000456  104414                          TRAP    14
000460  004737  000000G                 JSR     PC,EMS.ERR              ;                       6071
000464  062706  000014                  ADD     #14,SP                  ;                       6067
000470  010200                  20$:    MOV     R2,R0                   ; CUR.PRIORITY,*        6074
000472  104441                          TRAP    41
000474  000207                          RTS     PC                      ;                       5979

; Routine Size:  159 words,    Routine Base:  $CODE$ + 31542
; Maximum stack depth per invocation:  11 words


000024                                  .PSECT  $PLIT$,  RO ,  D
                        P.AAC:
000024  000000          2$:     .WORD   0                               ; CASE Table for ERR.HRD.RTNE+0072      6006
000026  000012                  .WORD   12                              ; [3$]
000030  000236                  .WORD   236                             ; [4$]
000032  000236                  .WORD   236                             ; [16$]
000034  000236                  .WORD   236                             ; [16$]
000036  000024                  .WORD   24                              ; [16$]
000040  000036                  .WORD   36                              ; [5$]
000042  000236                  .WORD   236                             ; [6$]
000044  000050                  .WORD   50                              ; [16$]
000046  000062                  .WORD   62                              ; [7$]
000050  000074                  .WORD   74                              ; [8$]
000052  000106                  .WORD   106                             ; [9$]
000054  000120                  .WORD   120                             ; [10$]
000056  000132                  .WORD   132                             ; [11$]
000060  000144                  .WORD   144                             ; [12$]
                                                                        ; [13$]
```

```
;    6077  1     routine ERR_SOFT_RTNE (ERRNUM) : novalue =
;    6078  1
;    6079  1     !+
;    6080  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6081  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6082  1     !-
;    6083  1
;    6084  2         begin
;    6085  2
;    6086  2         builtin
;    6087  2             PC;
;    6088  2
;    6089  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                    ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    6090  2         then
;    6091  2
;    6092  2             case .ERRNUM from 50 to 54 of
;    6093  2                 set
;    6094  2
;    6095  2                 [50]:       ERRSOFT (50, 0, 0);            ! CONTROLLER ERROR
;    6096  2
;    6097  2                 [51]:       ERRSOFT (51, 0, 0);            ! HOST MEMORY ACCESS ERROR
;    6098  2
;    6099  2                 [52]:       ERRSOFT (52, 0, 0);            ! DISK TRANSFER ERROR
;    6100  2
;    6101  2                 [53]:       ERRSOFT (53, 0, 0);            ! SDI ERROR
;    6102  2
;    6103  2                 [54]:       ERRSOFT (54, 0, 0);            ! SMALL DISK ERROR
;    6104  2                 tes
;    6105  2         else
;    6106  3             begin
;    6107  3             !****increment error count                     ! INCREMENT TOTAL ERROR COUNT
;    6108  3             PRINTB (SFT_MSG, .ERRNUM, .L$LUN, .PC);        ! PRINT ERROR LINE JUST LIKE DRS
;    6109  2             end;
;    6110  2
;    6111  1         end;


032240                                          .SBTTL  ERR.SOFT.RTNE RDRX INTERRUPT SERVICE ROUTINES
                                                .PSECT  $CODE$,  RO

000000  032737  020000  000006  ERR.SOFT.RTNE:
                                                BIT     #20000,SWP.FLAGS        ;                       6089
000006  001440                                  BEQ     7$
000010  016600  000002                          MOV     2(SP),RO                ; ERRNUM,*              6092
000014  162700  000062                          SUB     #62,RO
000020  006300                                  ASL     RO
000022  066007  000062'                        ADD     P.AAD(RO),PC            ; Case dispatch
000026  104457                          2$:     TRAP    57                      ;                       6095
000030  000062                                  .WORD   62
000032  000000                                  .WORD   0
000034  000000                                  .WORD   0
000036  000207                                  RTS     PC                      ;                       6092
000040  104457                          3$:     TRAP    57                      ;                       6097
```

```
000042  000063                          .WORD   63
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  000207                          RTS     PC                              ;                       6092
000052  104457                  4$:     TRAP    57                              ;                       6099
000054  000064                          .WORD   64
000056  000000                          .WORD   0
000060  000000                          .WORD   0
000062  000207                          RTS     PC                              ;                       6092
000064  104457                  5$:     TRAP    57                              ;                       6101
000066  000065                          .WORD   65
000070  000000                          .WORD   0
000072  000000                          .WORD   0
000074  000207                          RTS     PC                              ;                       6092
000076  104457                  6$:     TRAP    57                              ;                       6103
000100  000066                          .WORD   66
000102  000000                          .WORD   0
000104  000000                          .WORD   0
000106  000207                          RTS     PC                              ;                       6089
000110  010746                  7$:     MOV     PC,-(SP)                        ; PC,*                  6108
000114  013746  000000G                 MOV     L$LUN,-(SP)
000120  016646  000006                  MOV     6(SP),-(SP)                     ; ERRNUM,*
000124  012746  000000G                 MOV     #SFT.MSG,-(SP)
000130  012746  000004                  MOV     #4,-(SP)
000134  010600                          MOV     SP,R0                           ; SP,*
000136  104414                          TRAP    14
000140  062706  000012                  ADD     #12,SP                          ;                       6106
000144  000207                          RTS     PC                              ;                       6077

; Routine Size:  50 words,    Routine Base:  $CODE$ + 32240
; Maximum stack depth per invocation:  7 words

000062                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAD:
000062  000000                  1$:     .WORD   0                               ; CASE Table for ERR.SOFT.RTNE+0022     6092
000064  000012                          .WORD   12                              ; [2$]
000066  000024                          .WORD   24                              ; [3$]
000070  000036                          .WORD   36                              ; [4$]
000072  000050                          .WORD   50                              ; [5$]
```

```
;    6112  1      routine ERR_HRD_RTNE_APT (ERRNUM) : novalue =
;    6113  1
;    6114  1
;    6115  1      !+
;    6116  1      ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRHRD' MACRO CALL TO DRS OR TO FAKE
;    6117  1      ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6118  1      !-
;    6119  1
;    6120  1
;    6121  2          begin
;    6122  2
;    6123  2
;    6124  2          local
;    6125  2              CUR_PRIORITY;
;    6126  2
;    6127  2          builtin
;    6128  2              PC;
;    6129  2
;    6130  2          GETPRI (CUR_PRIORITY);
;    6131  2          SETPRI (PRI04);                          ! DON'T ALLOW SOFT_ERROR MESSAGES TO COME IN NOW
;    6132  2
;    6133  2
;    6134  2          if .APT_MODE
;    6135  2          then
;    6136  2
;    6137  3              begin
;    6138  3              .MAIL_BOX_TESTNUM = .RP_ADDR [LBN_LO];   ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    6139  3              .MAIL_BOX_SUBTST = .RP_ADDR [DISK];      ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER UNDER APT ONLY
;    6140  3              end;
;    6141  2
;    6142  2
;    6143  2          if (.ERRNUM lequ 34) or                  ! FOR NON-BAD BLOCK TYPE ERRORS
;    6144  2             (.ERRNUM gtru 38) or
;    6145  2             (.ERRNUM eql 36) or
;    6146  3             (.ERRNUM eql 37)
;    6147  3
;    6148  2          then
;    6149  2
;    6150  3              if BIT_TST (SWP_FLAGS, SWF_HRD)      ! IF ERRORS TO BE TREATED NORMALLY
;    6151  2              then
;    6152  2
;    6153  2                  case .ERRNUM from 31 to 45 of
;    6154  2                      set
;    6155  2
;    6156  2                      [31]:   ERRDF (31, EGH_30, EMS_30);   ! INVALID COMMAND
;    6157  2
;    6158  2                      [32]:   ERRDF (32, EGH_30, EMS_30);   ! COMMAND ABORTED
;    6159  2
;    6160  2                      [33]:   ;                             !
;    6161  2
;    6162  2                      [34]:   ;                             !
;    6163  2
;    6164  2                      [35]:   ;                             ! MEDIA FORMAT ERROR
```

```
;    6165  2
;    6166  2                [36]:   ERRDF (36, EGH_30, EMS_30);      ! WRITE PROTECTED
;    6167  2
;    6168  2                [37]:   ERRDF (37, EGH_30, EMS_30);      ! COMPARE ERROR
;    6169  2
;    6170  2                [38]:   ;                                ! DATA ERROR
;    6171  2
;    6172  2                [39]:   ERRDF (39, EGH_30, EMS_30);      ! HOST BUFFER ACCESS ERROR
;    6173  2
;    6174  2                [40]:   ERRDF (40, EGH_30, EMS_30);      ! CONTROLLER ERROR
;    6175  2
;    6176  2                [41]:   ERRDF (41, EGH_30, EMS_30);      ! DRIVE ERROR
;    6177  2
;    6178  2                [42]:   ERRDF (42, EGH_30, 0);           ! HOST WRITE COMPARE ERROR
;    6179  2
;    6180  2                [43]:   ERRDF (43, EGH_30, EMS_30);      ! MESSAGE FROM INTERNAL DIAGNOSTICS
;    6181  2
;    6182  2                [44]:   ERRDF (44, EGH_30, EMS_30);      ! DUPLICATE UNIT NUMBER
;    6183  2
;    6184  2                [45]:   ERRDF (45, EGH_30, EMS_30);      ! INVALID END CODE
;    6185  2                tes
;    6186  2
;    6187  2            else
;    6188  2
;    6189  3                begin
;    6190  3                !****increment error count                ! INCREMENT TOTAL ERROR COUNT
;    6191  3                PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);   ! PRINT ERROR MESSAGE JUST LIKE DRS
;    6192  3
;    6193  3
;    6194  3                if .ERRNUM neq 42
;    6195  3
;    6196  3                then
;    6197  4                    begin
;    6198  4                    PRINTB (HRD_SUB);                    ! NEXT LINE FOR NON-HOST COMPARE ERRORS
;    6199  4                    EMS_ERR ();                          ! PRINT REST OF THE INFORMATION
;    6200  3                    end;
;    6201  2                end;
;    6202  2
;    6203  2            if (.ERRNUM eql 35) or                       ! FOR BAD-BLOCK TYPE ERRORS
;    6204  3               (.ERRNUM eql 38)
;    6205  3
;    6206  2            then
;    6207  2
;    6208  3                if BIT_TST (SWP_FLAGS, SWF_BLK)          ! IF ERRORS TO BE TREATED NORMALLY
;    6209  2                then
;    6210  2
;    6211  2                    selectoneu .ERRNUM of
;    6212  2                        set
;    6213  2
;    6214  2                        [35]:   ERRDF (35, EGH_30, EMS_30);      ! MEDIA FORMAT ERROR
;    6215  2
;    6216  2                        [38]:   ERRDF (38, EGH_30, EMS_30);      ! DATA ERROR
;    6217  2                        tes
```

```
;    6218  2
;    6219  2              else
;    6220  2
;    6221  3                  begin
;    6222  3                  !****increment error count                    ! INCREMENT TOTAL ERROR COUNT
;    6223  3                  PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);        ! PRINT ERROR LINE JUST LIKE DRS
;    6224  3                  PRINTB (HRD_SUB);                             ! PRINT NEXT LINE TOO
;    6225  3                  EMS_ERR ();                                   ! PRINT REST OF THE INFORMATION
;    6226  2                  end;
;    6227  2
;    6228  2
;    6229  2          SETPRI (.CUR_PRIORITY);                               ! PRIORITY BACK TO NORMAL
;    6230  2
;    6231  2
;    6232  1          end;


032404                                          .SBTTL  ERR,HRD,RTNE,APT RDRX INTERRUPT SERVICE ROUTINES
                                                .PSECT  $CODE$,  RO

000000  004137  000000G                 ERR,HRD,RTNE,APT:
000004  104440                         JSR     R1,$SAVR2                   ;                                   6112
000006  010002                         TRAP    40                          ;                                   6130
                                        MOV     R0,R2                       ; *,CUR,PRIORITY
000010  012700  000200                 MOV     #200,R0                     ;                                   6131
000014  104441                         TRAP    41
000016  032737  000001  001162'        BIT     #1,APT,MODE                 ;                                   6134
000024  001412                         BEQ     1$
000026  013700  000000G                MOV     RP,ADDR,R0                  ;                                   6138
000032  016077  000050  001164'        MOV     50(R0),@MAIL,BOX,TESTNUM
000040  013700  000000G                MOV     RP,ADDR,R0                  ;                                   6139
000044  016077  000010  001166'        MOV     10(R0),@MAIL,BOX,SUBTST
000052  016601  000010          1$:     MOV     10(SP),R1                   ; ERRNUM,*                          6143
000056  020127  000042                 CMP     R1,#42
000062  101411                         BLOS    2$
000064  020127  000046                 CMP     R1,#46                      ;                                   6144
000070  101006                         BHI     2$
000072  020127  000044                 CMP     R1,#44                      ;                                   6145
000076  001405                         BEQ     2$
000100  020127  000045                 CMP     R1,#45                      ;                                   6146
000104  001131                         BNE     17$
000106  032737  010000  000000G  2$:    BIT     #10000,SWP,FLAGS            ;                                   6150
000114  001475                         BEQ     15$
000116  010100                         MOV     R1,R0                       ;                                   6153
000120  162700  000037                 SUB     #37,R0
000124  006300                         ASL     R0
000126  066007  000074'                ADD     P,AAE(R0),PC                ; Case dispatch
000132  104455                  4$:    TRAP    55                          ;                                   6156
000134  000037                         .WORD   37
000136  000000G                        .WORD   ECH,30
000140  000000G                        .WORD   EMS,30
000142  000512                         BR      17$
000144  104455                  5$:    TRAP    55                          ;                                   6158
```

F 3


```
000146  000040                          .WORD   40
000150  000000G                         .WORD   EGH.30
000152  000000G                         .WORD   EMS.30
000154  000505                          BR      17$
000156  104455                  6$:     TRAP    55                      ;                       6153
000160  000044                          .WORD   44                      ;                       6166
000162  000000G                         .WORD   EGH.30
000164  000000G                         .WORD   EMS.30
000166  000500                          BR      17$
000170  104455                  7$:     TRAP    55                      ;                       6153
000172  000045                          .WORD   45                      ;                       6168
000174  000000G                         .WORD   EGH.30
000176  000000G                         .WORD   EMS.30
000200  000473                          BR      17$
000202  104455                  8$:     TRAP    55                      ;                       6153
000204  000047                          .WORD   47                      ;                       6172
000206  000000G                         .WORD   EGH.30
000210  000000G                         .WORD   EMS.30
000212  000466                          BR      17$
000214  104455                  9$:     TRAP    55                      ;                       6153
000216  000050                          .WORD   50                      ;                       6174
000220  000000G                         .WORD   EGH.30
000222  000000G                         .WORD   EMS.30
000224  000461                          BR      17$
000226  104455                  10$:    TRAP    55                      ;                       6153
000230  000051                          .WORD   51                      ;                       6176
000232  000000G                         .WORD   EGH.30
000234  000000G                         .WORD   EMS.30
000236  000454                          BR      17$
000240  104455                  11$:    TRAP    55                      ;                       6153
000242  000052                          .WORD   52                      ;                       6178
000244  000000G                         .WORD   EGH.30
000246  000000                          .WORD   0
000250  000447                          BR      17$
000252  104455                  12$:    TRAP    55                      ;                       6153
000254  000053                          .WORD   53                      ;                       6180
000256  000000G                         .WORD   EGH.30
000260  000000G                         .WORD   EMS.30
000262  000442                          BR      17$
000264  104455                  13$:    TRAP    55                      ;                       6153
000266  000054                          .WORD   54                      ;                       6182
000270  000000G                         .WORD   EGH.30
000272  000000G                         .WORD   EMS.30
000274  000435                          BR      17$
000276  104455                  14$:    TRAP    55                      ;                       6153
000300  000055                          .WORD   55                      ;                       6184
000302  000000G                         .WORD   EGH.30
000304  000000G                         .WORD   EMS.30
000306  000430                          BR      17$
000310  010746                  15$:    MOV     PC,-(SP)                ; PC,*                  6190
000312  013746  000000G                 MOV     L$LUN,-(SP)                                     6191
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                  ;                               6194
000340  001411                                  BEQ     16$
000342  012716  000000G                         MOV     #HRD.SUB,(SP)           ;                               6198
000346  012746  000001                          MOV     #1,-(SP)
000352  010600                                  MOV     SP,R0                   ; SP,*
000354  104414                                  TRAP    14
000356  004737  000000G                         JSR     PC,EMS.ERR              ;                               6199
000362  005726                                  TST     (SP)+                   ;                               6197
000364  062706  000012              16$:        ADD     #12,SP                  ;                               6189
000370  020127  000043              17$:        CMP     R1,#43                  ;                               6203
000374  001403                                  BEQ     18$
000376  020127  000046                          CMP     R1,#46                  ;                               6204
000402  001050                                  BNE     21$
000404  032737  040000  000000G     18$:        BIT     #40000,SWP,FLAGS        ;                               6208
000412  001420                                  BEQ     20$
000414  020127  000043                          CMP     R1,#43                  ;                               6214
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$                     ;                               6211
000434  020127  000046              19$:        CMP     R1,#46                  ;                               6216
000440  001031                                  BNE     21$
000442  104455                                  TRAP    55
000444  000046                                  .WORD   46
000446  000000G                                 .WORD   EGH.30
000450  000000G                                 .WORD   EMS.30
000452  000424                                  BR      21$                     ;                               6208
000454  010746              20$:                MOV     PC,-(SP)                ; PC,*                          6223
000460  013746  000000G                         MOV     L$LUN,(SP)
000464  010146                                  MOV     R1,-(SP)
000466  012746  000000G                         MOV     #DF.MSG,-(SP)
000472  012746  000004                          MOV     #4,-(SP)
000476  010600                                  MOV     SP,R0                   ; SP,*
000500  104414                                  TRAP    14
000502  012716  000000G                         MOV     #HRD.SUB,(SP)           ;                               6224
000506  012746  000001                          MOV     #1,-(SP)
000512  010600                                  MOV     SP,R0                   ; SP,*
000514  104414                                  TRAP    14
000516  004737  000000G                         JSR     PC,EMS.ERR              ;                               6225
000522  062706  000014                          ADD     #14,SP                  ;                               6221
000526  010200              21$:                MOV     R2,R0                   ; CUR.PRIORITY,*                6229
000530  104441                                  TRAP    41
000532  000207                                  RTS     PC                      ;                               6112

; Routine Size:  173 words,    Routine Base:  $CODE$ + 32404
; Maximum stack depth per invocation:  11 words
```

```
000074                                          .PSECT  $PLIT$,  RO ,  D

                                        P.AAE:
000074  000000                         3$:    .WORD   0                       ; CASE Table for ERR.HRD.RTNE.AP+0126   6153
000076  000012                                .WORD   12                      ; [4$]
000100  000236                                .WORD   236                     ; [5$]
000102  000236                                .WORD   236                     ; [17$]
000104  000236                                .WORD   236                     ; [17$]
000106  000024                                .WORD   24                      ; [17$]
000110  000036                                .WORD   36                      ; [6$]
000112  000236                                .WORD   236                     ; [7$]
000114  000050                                .WORD   50                      ; [17$]
000116  000062                                .WORD   62                      ; [8$]
000120  000074                                .WORD   74                      ; [9$]
000122  000106                                .WORD   106                     ; [10$]
000124  000120                                .WORD   120                     ; [11$]
000126  000132                                .WORD   132                     ; [12$]
000130  000144                                .WORD   144                     ; [13$]
                                                                            ; [14$]


;    6233  1
;    6234  1
```

```
;    6235  1     routine ERR_SOFT_RTNE_APT (ERRNUM, index) : novalue =
;    6236  1
;    6237  1     !+
;    6238  1     ! THIS ROUTINE DECIDES WHETHER TO ISSUE AN 'ERRSOFT' MACRO CALL TO DRS OR TO FAKE
;    6239  1     ! THE SAME EFFECT WITHOUT ISSUING THE CALL
;    6240  1     !-
;    6241  1
;    6242  2         begin
;    6243  2
;    6244  2         local
;    6245  2             ELOG_ADDR : ref block [EP_LEN, word] field (EP_FIELDS);
;    6246  2
;    6247  2         builtin
;    6248  2             PC;
;    6249  2
;    6250  2         ELOG_ADDR = ELOG_PKT + (.index * EP_LEN * 2);        ! ADDRESS OF THE SAVED ERROR-LOG INFORMATION
;    6251  2
;    6252  2         if .APT_MODE
;    6253  2         then
;    6254  3             begin
;    6255  3             .MAIL_BOX_TESTNUM = .ELOG_ADDR [EL_BLOCK];       ! CHANGE TEST NUMBER TO SHOW LBN UNDER APT ONLY
;    6256  3             .MAIL_BOX_SUBTST = .ELOG_ADDR [EL_DK_NUM];       ! CHANGE SUB-TEST NUMBER TO SHOW DISK NUMBER IN APT ONLY
;    6257  2             end;
;    6258  2
;    6259  3         if BIT_TST (SWP_FLAGS, SWF_SFT)                      ! IF SOFT ERRORS TO BE TREATED LIKE OTHER ERRORS
;    6260  2         then
;    6261  2
;    6262  2             case .ERRNUM from 50 to 54 of
;    6263  2                 set
;    6264  2
;    6265  2                 [50]:       ERRDF (50, 0, 0);                ! CONTROLLER ERROR
;    6266  2
;    6267  2                 [51]:       ERRDF (51, 0, 0);                ! HOST MEMORY ACCESS ERROR
;    6268  2
;    6269  2                 [52]:       ERRDF (52, 0, 0);                ! DISK TRANSFER ERROR
;    6270  2
;    6271  2                 [53]:       ERRDF (53, 0, 0);                ! SDI ERROR
;    6272  2
;    6273  2                 [54]:       ERRDF (54, 0, 0);                ! SMALL DISK ERROR
;    6274  2                 tes
;    6275  2         else
;    6276  3             begin
;    6277  3             !****increment error count                       ! INCREMENT TOTAL ERROR COUNT
;    6278  3             PRINTB (DF_MSG, .ERRNUM, .L$LUN, .PC);           ! PRINT ERROR LINE JUST LIKE DRS
;    6279  2             end;
;    6280  2
;    6281  1         end;


033136                                          .SBTTL  ERR.SOFT.RTNE.APT RDRX INTERRUPT SERVICE ROUTINES
                                                .PSECT  $CODE$,  RO

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

```
                                        MOV     2(SP),-(SP)             ; INDEX,*                       6250
000004  012746  000102                  MOV     #102,-(SP)
000010  004737  000000G                 JSR     PC,BL$MUL
000014  062700  000000G                 ADD     #ELOG.PKT,R0
000020  032737  000001  001162'         BIT     #1,APT.MODE             ;                               6252
000026  001406                          BEQ     1$
000030  016077  000056  001164'         MOV     56(R0),@MAIL.BOX.TESTNUM        ; *(ELOG.ADDR),*        6255
000036  016077  000012  001166'         MOV     12(R0),@MAIL.BOX.SUBTST         ; *(ELOG.ADDR),*        6256
000044  032737  020000  000000G 1$:     BIT     #20000,SWP.FLAGS        ;                               6259
000052  001410                          BEQ     8$
000054  016600  000010                  MOV     10(SP),R0               ; ERRNUM,*                      6262
000060  162700  000062                  SUB     #62,R0
000064  006300                          ASL     R0
000066  066007  000132'                 ADD     P.AAF(R0),PC            ; Case dispatch
000072  104455              3$:         TRAP    55                      ;                               6265
000074  000062                          .WORD   62
000076  000000                          .WORD   0
000100  000000                          .WORD   0
000102  000441                          BR      9$                      ;                               6262
000104  104455              4$:         TRAP    55                      ;                               6267
000106  000063                          .WORD   63
000110  000000                          .WORD   0
000112  000000                          .WORD   0
000114  000434                          BR      9$                      ;                               6262
000116  104455              5$:         TRAP    55                      ;                               6269
000120  000064                          .WORD   64
000122  000000                          .WORD   0
000124  000000                          .WORD   0
000126  000427                          BR      9$                      ;                               6262
000130  104455              6$:         TRAP    55                      ;                               6271
000132  000065                          .WORD   65
000134  000000                          .WORD   0
000136  000000                          .WORD   0
000140  000422                          BR      9$                      ;                               6262
000142  104455              7$:         TRAP    55                      ;                               6273
000144  000066                          .WORD   66
000146  000000                          .WORD   0
000150  000000                          .WORD   0
000152  000415                          BR      9$                      ;                               6259
000154  010716              8$:         MOV     PC,(SP)                 ; PC,*                          6278
000156  013746  000000G                 MOV     L$LUN,-(SP)
000162  016646  000012                  MOV     12(SP),-(SP)            ; ERRNUM,*
000166  012746  000000G                 MOV     #DF.MSG,-(SP)
000172  012746  000004                  MOV     #4, (SP)
000176  010600                          MOV     SP,R0                   ; SP,*
000200  104414                          TRAP    14
000202  062706  000010                  ADD     #10,SP                  ;                               6233
000206  022626              9$:         CMP     (SP)+,(SP)+             ;                               6234
000210  000207                          RTS     PC                      ;                               6235

; Routine Size:  69 words,    Routine Base:  $CODE$ + 33136
; Maximum stack depth per invocation:  8 words
```

```
000132                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAF:
000132  000000                2$:     .WORD   0               ; CASE Table for ERR.SOFT.RTNE.A+0066   6262
000134  000012                        .WORD   12              ; [3$]
000136  000024                        .WORD   24              ; [4$]
000140  000036                        .WORD   36              ; [5$]
000142  000050                        .WORD   50              ; [6$]


;    6282  1
;    6283  1
;    6284  1    end
;    6285  1
;    6286  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
;         $GGG$                           324       RO ,  I  ,  LCL,  REL,  CON
;         $CODE$                         7028       RO ,  I  ,  LCL,  REL,  CON
;         $PLIT$                           50       RO ,  D  ,  LCL,  REL,  CON


;                                   Library Statistics
;
;                                                  Symbols
;       File                                Total   Loaded   Percent      Pages Mapped   Processing Time
;
;  DISK$USER2:[POWERS]ZRQAD0.L16;3            404      333        82             21          00:00.1


;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAD0.BL2/LIST=ZRQAD0.LS2/OBJECT=ZRQAD0.OB2/SOURCE=PAGE:53
```

K3


```
;    6287  0     module ZRQAM4 (
;    6288  0
;    6289  0     %title 'RD/RX EXERCISER'
;    6290  0                  ident = 'V01.6',
;    6291  0                  addressing_mode (absolute),
;    6292  0                  environment (noeis)
;    6293  0                  ) %
;    6294  0
;    6295  1     begin
;    6296  1
;    6297  1     %sbttl 'LASTAD AND SETUP'
;    6298  1
;    6299  1     library 'ZRQADO.L16';
;    6300  1
;    6301  1     require 'BLSMAC.REQ';                          ! DIAGNOSTIC SUPERVISOR LIBRARY
;    7792  1
;    7793  2     LASTAD
;    7794  2
;    7795  2     BGNSETUP (4)                                                        !ZZZ
;    7796  2
; P  7797  2             BGNPTAB
; P  7798  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'100060', 0, 0, RD52_MAX_LBN, 0  !ZZZ
; P  7799  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
; P  7800  2
;    7801  2             ENDPTAB
;    7802  2
; P  7803  2             BGNPTAB
; P  7804  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'100001', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  7805  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    7806  2             ENDPTAB
;    7807  2
; P  7808  2             BGNPTAB
; P  7809  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'100002', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  7810  2                             ! IP, VECTOR, BR, DISK ADDR, START BLOCK, END BLOCK
;    7811  2             ENDPTAB
; P  7812  2             BGNPTAB
; P  7813  2                     INIT_IP_ADDR, INIT_INTR_VECT, INIT_BR_LEVEL, %o'100003', 0, 0, RX50_MAX_LBN, 0  !ZZZ
; P  7814  2                     !HERE'S ONE FOR THE 4TH DRIVE                                                   !ZZZ
;    7815  2             ENDPTAB                                                                                 !ZZZ
;    7816  2
;    7817  1     ENDSETUP


                                        .TITLE  ZRQAM4 RD/RX EXERCISER
                                        .IDENT  /V01.6/
                                        .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  100060                          .WORD   -77720
000020  000000                          .WORD   0
000022  000000                          .WORD   0
000024  150477                          .WORD   -27301
000026  000000                          .WORD   0
000030  000060'                 P.AAC:  .WORD   L$LAST+54
000032  000010                          .WORD   10                      ; Plit count word
000034  172150                  P.AAD:  .WORD   -5630
000036  000154                          .WORD   154
000040  000004                          .WORD   4
000042  100001                          .WORD   -77777
000044  000000                          .WORD   0
000046  000000                          .WORD   0
000050  001437                          .WORD   1437
000052  000000                          .WORD   0
000054  000104'                 P.AAE:  .WORD   L$LAST+100
000056  000010                          .WORD   10                      ; Plit count word
000060  172150                  P.AAF:  .WORD   -5630
000062  000154                          .WORD   154
000064  000004                          .WORD   4
000066  100002                          .WORD   -77776
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  100003                          .WORD   -77775
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
```

```
                                        .SBTTL  $END.LINK LASTAD AND SETUP
000000  000207                  $END.LINK::
                                        RTS     PC                                      ;                                       7791

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


;    7818  1      end
;    7819  1
;    7820  0      eludom


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;         $XYZ$                           44       RO ,  I  ,  LCL,  REL,  CON


;                               Library Statistics
;
;                                       -------- Symbols --------        Pages       Processing
;       File                            Total    Loaded   Percent       Mapped      Time
;
;  DISK$USER2:[POWERS]ZRQAD0.L16;3       404         7         1          21        00:00.2


;                                       COMMAND QUALIFIERS

;       BLISS/PDP11 ZRQAD0.BL2/LIST=ZRQAD0.LS2/OBJECT=ZRQAD0.OB2/SOURCE=PAGE:53

; Size:           7029 code + 417 data words
; Run Time:         03:40.4
; Elapsed Time:     05:47.2
; Lines/CPU Min:     2129
; Lexemes/CPU-Min: 20098
; Memory Used:  477 pages
; Compilation Complete
```

```
ZRQAM3          RD/R....B1
ZRQAM3          RD/R....C1
ZRQAM3          RD/R....D1
ZRQAM3          RD/R....E1
ZRQAM3          RD/R....F1
ZRQAM3          RD/R....G1
ZRQAM3          RD/R....H1
ZRQAM3          RD/R....I1
ZRQAM3          RD/R....J1
ZRQAM3          RD/R....K1
ZRQAM3          RD/R....L1
ZRQAM3          RD/R....M1
ZRQAM3          RD/R....N1

ZRQAM3          RD/R....B2
ZRQAM3          RD/R....C2
ZRQAM3          RD/R....D2
ZRQAM3          RD/R....E2
ZRQAM3          RD/R....F2
ZRQAM3          RD/R....G2
ZRQAM3          RD/R....H2
ZRQAM3          RD/R....I2
ZRQAM3          RD/R....J2
ZRQAM3          RD/R....K2
ZRQAM3          RD/R....L2
ZRQAM3          RD/R....M2
ZRQAM3          RD/R....N2

ZRQAM3          RD/R....B3
ZRQAM3          RD/R....C3
ZRQAM3          RD/R....D3
ZRQAM3          RD/R....E3
ZRQAM3          RD/R....F3
ZRQAM3          RD/R....G3
ZRQAM3          RD/R....H3
ZRQAM3          RD/R....I3
ZRQAM3          RD/R....J3
ZRQAM4          RD/R....K3
ZRQAM4          RD/R....L3
ZRQAM4          RD/R....M3
```