F 10

```
MMM         MMM     SSSSSSSSSSSS     CCCCCCCCCCC   PPPPPPPPPPPP
MMM         MMM     SSSSSSSSSSSS     CCCCCCCCCCC   PPPPPPPPPPPP
MMM         MMM     SSSSSSSSSSSS     CCCCCCCCCCC   PPPPPPPPPPPP
MMMMMM   MMMMMM  SSS              CCC              PPP         PPP
MMMMMM   MMMMMM  SSS              CCC              PPP         PPP
MMMMMM   MMMMMM  SSS              CCC              PPP         PPP
MMM   MMM   MMM  SSS              CCC              PPP         PPP
MMM   MMM   MMM  SSS              CCC              PPP         PPP
MMM   MMM   MMM  SSS              CCC              PPP         PPP
MMM         MMM     SSSSSSSSS     CCC              PPPPPPPPPPPP
MMM         MMM     SSSSSSSSS     CCC              PPPPPPPPPPPP
MMM         MMM     SSSSSSSSS     CCC              PPPPPPPPPPPP
MMM         MMM              SSS  CCC              PPP
MMM         MMM              SSS  CCC              PPP
MMM         MMM              SSS  CCC              PPP
MMM         MMM              SSS  CCC              PPP
MMM         MMM              SSS  CCC              PPP
MMM         MMM              SSS  CCC              PPP
MMM         MMM  SSSSSSSSSSSS        CCCCCCCCCCC   PPP
MMM         MMM  SSSSSSSSSSSS        CCCCCCCCCCC   PPP
MMM         MMM  SSSSSSSSSSSS        CCCCCCCCCCC   PPP
```

FILEID**MSCP

I 12

```
MM      MM    SSSSSSSS    CCCCCCCC  PPPPPPPP
MM      MM    SSSSSSSS    CCCCCCCC  PPPPPPPP
MMMM  MMMM  SS          CC          PP      PP
MMMM  MMMM  SS          CC          PP      PP
MM  MM  MM  SS          CC          PP      PP
MM  MM  MM  SS          CC          PP      PP
MM      MM    SSSSSS    CC          PPPPPPPP
MM      MM    SSSSSS    CC          PPPPPPPP
MM      MM          SS  CC          PP
MM      MM          SS  CC          PP
MM      MM          SS  CC          PP              ....
MM      MM          SS  CC          PP              ....
MM      MM  SSSSSSSS      CCCCCCCC  PP              ....
MM      MM  SSSSSSSS      CCCCCCCC  PP              ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0000        .TITLE  MSCP - MSCP Server - Emulator
0000        .IDENT  'V04-001'
0000
0000 ;****************************************************************************
0000 ;*                                                                          *
0000 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000 ;*  ALL RIGHTS RESERVED.                                                    *
0000 ;*                                                                          *
0000 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000 ;*  TRANSFERRED.                                                            *
0000 ;*                                                                          *
0000 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000 ;*  CORPORATION.                                                            *
0000 ;*                                                                          *
0000 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000 ;*                                                                          *
0000 ;*                                                                          *
0000 ;****************************************************************************
0000
0000 ;++
0000 ;
0000 ; FACILITY:     MSCP Server
0000 ;
0000 ; ABSTRACT:
0000 ;
0000 ;       This program implements an MSCP(Mass Storage Control Protocol) server.
0000 ;       It accepts as input valid MSCP command packets.  It then accesses the
0000 ;       disk(s) connected to it to perform the indicated operations.  It
0000 ;       returns MSCP status packets and possibly data to the driver.
0000 ;
0000 ; ENVIORNMENT:  User mode
0000 ;
0000 ; AUTHOR:       Kerbey T. Altmann
0000 ;
0000 ; CREATION DATE:06-AUG-1980
0000 ;
0000 ; MODIFIED BY:
0000 ;
0000 ;       V04-001 DWT0240         David W. Thiel          7-Sep-1984
0000 ;               Disable advertisement of volume shadowing capabilities.
0000 ;
0000 ;       V03-015 DWT0232         David W. Thiel/Ellen M. Batbouta        31-Aug-1984
0000 ;               Update storage allocation scheme.  Correct references
0000 ;               to CDRP$L_CDT and CDRP$L_CDTX.  Tie local CDRP
0000 ;               definitions to global symbols better.  Detect and
0000 ;               bugcheck when response sent using stale CDT address.
0000 ;               Deal with CDRP$W_STATE more rigorously.  Add
0000 ;               bug traps to packet deallocation routine.  Correct
0000 ;               packet initialization on invalid commands.  Support
0000 ;               determine access paths.  Include every incoming
```

```
0000  ;                message in OPC_COUNT.  Add reference count to HQB.
0000  ;                Adjust values of PACKETS and SMALL to guarantee that
0000  ;                things will not hang.  Make acess command return
0000  ;                invalid opcode.  Validate I/O function code before
0000  ;                queuing request to driver.  Return correct setting
0000  ;                of MSCP$B_CPYS for UDA disks.  Add many bug traps.
0000  ;
0000  ;        V03-014 EMB0108         Ellen M. Batbouta       19-JUL-1984
0000  ;                Check for mount verification in progress on a local
0000  ;                device in routines, NSEQ and NSEQB.  If MV is in
0000  ;                progress, mark unit as available and return without
0000  ;                processing the request.  This prevents a deadlock
0000  ;                situation.
0000  ;
0000  ;        V03-013 KTA3130         Kerbey T. Altmann       15-May-1984
0000  ;                Fix error in subcode.
0000  ;
0000  ;        V03-012 KTA3127         Kerbey T. Altmann       23-Apr-1984
0000  ;                Fix problem with VC error.
0000  ;
0000  ;        V03-011 KTA3116         Kerbey T. Altmann       02-Apr-1984
0000  ;                Decrement UCB queue length cell, and add entry to
0000  ;                error table for SS$_TIMEOUT.
0000  ;
0000  ;        V03-010 KTA3112         Kerbey T. Altmann       12-Mar-1984
0000  ;                Redo way UQB pointers are handled.  Also redo handling
0000  ;                of blocked commands.
0000  ;
0000  ;        V03-009 KTA3108         Kerbey T. Altmann       06-Mar-1984
0000  ;                Fix two small bugs and allow for new error return from
0000  ;                IO$_PACKACK.
0000  ;
0000  ;        V03-008 KTA3089         Kerbey T. Altmann       12-Nov-1983
0000  ;                Rewrite to add shadowing.
0000  ;
0000  ;        V03-007 ROW0278         Ralph O. Weber          12-JAN-1984
0000  ;                Remove reference to MSCP$M_UF_INACT, the inactive shadow set
0000  ;                unit flag.  This is no longer a unit flag.
0000  ;
0000  ;        V03-006 ROW0246         Ralph O. Weber          26-OCT-1983
0000  ;                Move the "no units at this address" exit label in GET_UNIT_STS
0000  ;                such that a "next unit modifier" get unit status command gets
0000  ;                a zero unit number returned when there are no units to
0000  ;                describe.
0000  ;
0000  ;        V03-005 ROW0236         Ralph O. Weber          10-OCT-1983
0000  ;                Move increment of number of online hosts counter in UQB to
0000  ;                before increment of UCB$B_ONLCNT in online routine.  Also make
0000  ;                increment of UCB$B_ONLCNT conditional on UQB$B_HOST_CNT
0000  ;                indicating that this is the first host to bring the unit
0000  ;                online.  This makes online operations consistant with their
0000  ;                description by Kerbey and with the operations performed when a
0000  ;                available command is processed.
0000  ;
0000  ;        V03-004 KTA3081         Kerbey T. Altmann       13-Sep-1983
0000  ;                Upgrade AVAIL support. Add underpinnings for shadowing.
0000  ;
```

```
0000      ;       V03-003 SSA0001         Stan Amway              12-Sep-1983
0000      ;               Add I/O performance monitoring support.
0000      ;               Redefined CDRP$L_LBN to avoid conflicting use
0000      ;               of CDRP$L_SEQNUM7IRP$L_SEQNUM by PMS routines.
0000      ;
0000      ;       V03-002 KTA3076         Kerbey T. Altmann       12-Aug-1983
0000      ;               Add support for explicit PACK_ACK and AVAIL.
0000      ;               Redo error routines.  Return ALLOCLASS.
0000      ;
0000      ;       V03-001 KTA3070         Kerbey T. Altmann       09-Jul-1983
0000      ;               Fix bug in host que search.
0000      ;
0000      ;--
```

```
0000            .SBTTL  DECLARATIONS
0000
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000
0000            .NOCROSS
0000            $CDRPDEF
0000            $CDTDEF
0000            $DDTDEF
0000            $DEVDEF
0000            $DYNDEF
0000            $IODEF
0000            $IPLDEF
0000            $IRPDEF
0000            $MSCPDEF
0000            $PDTDEF
0000            $PRDEF
0000            $SCSDEF                         ; SCS message header definitions
0000            $SLVDEF
0000            $SSDEF
0000            $UCBDEF
0000            $VADEF
0000
0000            $MSUDEF
0000            $HQBDEF
0000            $UQBDEF
0000
0000    ;
0000    ; MACROS:
0000    ;
0000
0000            .MACRO  BLOCKH
0000             BSBW   BLOCKEDH
0000            .ENDM
0000
0000            .MACRO  BLOCKT
0000             BSBW   BLOCKEDT
0000            .ENDM
0000
0000            .MACRO  WFINT
0000             BSBW   WAIT_FOR_INT
0000            .ENDM
0000
0000            .MACRO  WFINTX
0000             BSBW   WAIT_FOR_INTX
0000            .ENDM
0000    ;
0000            .MACRO  QUE_HEAD LABEL
0000    LABEL:  .LONG   LABEL-MSCP_BEGIN
0000            .LONG   LABEL-MSCP_BEGIN
0000            .ENDM
0000
0000            .MACRO  ACTION,SYSERR,MSCPERR,ROUTINE
0000             .WORD  SYSERR
0000             .WORD  MSCPERR
0000             .IF B ROUTINE
```

```
            0000                .WORD 0
            0000              .IFF
            0000                .WORD ROUTINE-.
            0000              .ENDC
            0000              .ENDM
            0000
            0000   ;
            0000   ; EQUATED SYMBOLS:
            0000   ;
            0000
            0000          HOSTSET           =      <MSCP$M_UF_CMPRD!-
            0000                                    MSCP$M_UF_CMPWR!-
            0000                                    MSCP$M_UF_576!-
            0000                                    MSCP$M_UF_SCCHH!-
            0000                                    MSCP$M_UF_SCCHL!-
            0000                                    MSCP$M_UF_WBKNV!-
00001C47    0000                                    MSCP$M_UF_WRTPS>
            0000
            0000          NONHOSTSET        =      <MSCP$M_UF_REPLC!-
            0000                                    MSCP$M_UF_RMVBL!-
            0000                                    MSCP$M_UF_WRTPH!-
            0000                                    MSCP$M_UF_SSMEM!-
0000E280    0000                                    MSCP$M_UF_SSMST>
            0000
            0000   ;
FFFFFFB0    0000          CDRP$L_MSGBUFX    =      CDRP$L_AST
FFFFFFB4    0000          CDRP$L_BUFLEN     =      CDRP$L_ASTPRM
FFFFFFB8    0000          CDRP$L_BUFADR     =      CDRP$L_WIND
FFFFFFC2    0000          CDRP$B_SHDW_CNT   =      CDRP$B_EFN
FFFFFFC4    0000          CDRP$L_UQB        =      CDRP$L_IOSB
FFFFFFD6    0000          CDRP$W_STATE      =      CDRP$L_BCNT+4
FFFFFFE8    0000          CDRP$L_UQB_FL     =      CDRP$L_SEGVBN
FFFFFFEC    0000          CDRP$L_UQB_BL     =      CDRP$L_DIAGBUF
FFFFFFF4    0000          CDRP$L_CMD_STS    =      CDRP$L_EXTEND
FFFFFFF8    0000          CDRP$W_SLOT       =      CDRP$L_ARB
FFFFFFFA    0000          CDRP$B_FLAGS      =      CDRP$L_ARB+2
FFFFFFFB    0000          CDRP$B_OPCODE     =      CDRP$L_ARB+3
            0000   ;
            0000   ; Define MSCP server private areas beyond normal CDRP fields
            0000   ;
00000048    0000          CDRP$L_LBN        =      CDRP$K_CD_LEN
0000004C    0000          CDRP$L_SHDW_LNK   =      CDRP$L_LBN+4
00000050    0000          CDRP$L_FPCX       =      CDRP$L_SHDW_LNK+4
00000054    0000          CDRP$L_FR3X       =      CDRP$L_FPCX+4
00000058    0000          CDRP$L_FR4X       =      CDRP$L_FR3X+4
0000005C    0000          CDRP$L_SAVD_RTX   =      CDRP$L_FR4X+4
00000060    0000          CDRP$L_CDTX       =      CDRP$L_SAVD_RTX+4
00000064    0000          CDRP$L_XCT_LENX   =      CDRP$L_CDTX+4
00000068    0000          CDRP$B_LBUFF      =      CDRP$L_XCT_LENX+4
00000074    0000          CDRP$L_STRT_UQB   =      CDRP$B_LBUFF+12
00000078    0000          CDRP$L_MST_UQB    =      CDRP$L_STRT_UQB+4
0000007C    0000          CDRP$L_HQB        =      CDRP$L_MST_UQB+4
000000E0    0000   BIG_CDRP =               <CDRP$L_HQB+4>-CDRP$L_IOQFL
            0000
00000022    0000          IRP$B_SHDW_CNT    =      CDRP$B_SHDW_CNT-CDRP$L_IOQFL
00000024    0000          IRP$L_UQB         =      CDRP$L_UQB-CDRP$L_IOQFL
00000036    0000          IRP$W_STATE       =      CDRP$W_STATE-CDRP$L_IOQFL
```

```
00000058  0000          IRP$W_SLOT        =      CDRP$W_SLOT-CDRP$L_IOQFL
0000005A  0000          IRP$B_FLAGS       =      CDRP$B_FLAGS-CDRP$L_IOQFL
000000A8  0000          IRP$L_LBN         =      CDRP$L_LBN-CDRP$L_IOQFL
000000AC  0000          IRP$L_SHDW_LNK    =      CDRP$L_SHDW_LNK-CDRP$L_IOQFL
000000B0  0000          IRP$L_FPCX        =      CDRP$L_FPCX-CDRP$L_IOQFL
000000B4  0000          IRP$L_MAINCDRP    =      CDRP$L_FR3X-CDRP$L_IOQFL
000000B8  0000          IRP$L_FR4X        =      CDRP$L_FR4X-CDRP$L_IOQFL
          0000          ASSUME <IRP$L_FR4X+4> LE IRP$C_LENGTH
          0000
00000000  0000          CDRP$V_SLAVE      =      0          ; This is a slave IRP
00000001  0000          CDRP$V_BUFFER     =      1          ; This CDRP has a buffer
          0000  ;       CDRP$V_           =      2          ;
          0000  ;       CDRP$V_           =      3          ;
00000004  0000          CDRP$V_ABORT      =      4          ; This CDRP has been aborted
00000005  0000          CDRP$V_NOHSTBUF   =      5          ; This CDRP has no host buffer
          0000
00000000  0000          CDRP$K_ST_MEM_WAIT =     0          ; Buffer memory wait
00000001  0000          CDRP$K_ST_MAP_WAIT =     1          ; SCS mapping resource wait
00000002  0000          CDRP$K_ST_DRVRD_WAIT =   2          ; Disk read wait
00000003  0000          CDRP$K_ST_DATA_WAIT =    3          ; SCS data transfer wait
00000004  0000          CDRP$K_ST_DRVWRT_WAIT =  4          ; Disk write wait
00000005  0000          CDRP$K_ST_MSG_WAIT =     5          ; Response message credit wait
00000006  0000          CDRP$K_ST_SHQ_QUE =      6          ; Special holding queue wait
00000007  0000          CDRP$K_ST_BLK_QUE =      7          ; Blocked queue wait
00000008  0000          CDRP$K_ST_ONBLK_QUE =    8          ; Online-copy blocked queue
00000009  0000          CDRP$K_ST_MSG1_WAIT =    9          ; Attention message credit/buffer wait
0000003F  0000          CDRP$K_ST_INVALID =      63         ; Invalid state
          0000
          0000  ;
          0000  ; EXTERNAL SYMBOLS:
          0000  ;
          0000
          0000          .CROSS
          0000          .DEFAULT DISPLACEMENT WORD
          0000
```

```
                            0000            .SBTTL  COMMUNICATIONS AREA
                            0000
                        00000000            .PSECT  MSCP$CODE,PAGE
                            0000
                            0000    MSCP_BEGIN:
                            0000    MSCP_COMM_AREA:
                            0000
               00000001     0000            PRMSW   = 1
                            0000
                            0000            SLVTAB  END      = MSCP_END,-
                            0000                    INITRTN = MSCP_START,-
                            0000                    SUBTYP   = DYN$C_LC_MSCP,-
                            0000                    FACILITY= <MSCP_SERVER>
                            0024
                            0024            LOADVEC TYPE        = SLV$K_SDATA,-
                            0024                    ENTRY       = SCS$GL_MSCP,-
                            0024                    SEC_LABEL = MSCP_BEGIN
               00000000     002D                    DEF_RTN     = 0
                            002D
FFFFFFFF FFFFFFFF FFFFFFFF  002D            .LONG   -1,-1,-1
                            0039
                            0039            .ALIGN  QUAD
                            0040
               00000040     0040            .=MSCP_COMM_AREA+MSU$L_STATE
                            0040    STATE:
               00000000     0040            .LONG   0
                            0044
               00000044     0044            .=MSCP_COMM_AREA+MSU$W_PACKET
                   0000     0044    PACKET: .WORD   0
                            0046
               00000046     0046            .=MSCP_COMM_AREA+MSU$W_INI_PKT
                            0046    INI_PKT:                                ; Number of allocated (free+in-use) packets
                   0000     0046            .WORD   0
                            0048
               00000048     0048            .=MSCP_COMM_AREA+MSU$W_NUM_PKT
                            0048    NUM_PKT:                                ; Number of free packets
                   0000     0048            .WORD   0
                            004A
               0000004A     004A            .=MSCP_COMM_AREA+MSU$W_MAX_PKT
                            004A    MAX_PKT:                                ; Maximum number of packets to cache
                   0000     004A            .WORD   0
                            004C
               0000004C     004C            .=MSCP_COMM_AREA+MSU$W_INI_HOST
                            004C    INI_HOST:
                   0000     004C            .WORD   0
                            004E
               0000004E     004E            .=MSCP_COMM_AREA+MSU$W_NUM_HOST
                            004E    NUM_HOST:
                   0000     004E            .WORD   0
                            0050
               00000050     0050            .=MSCP_COMM_AREA+MSU$L_BUFF_HEAD
                            0050    BUFF_HEAD:
               00000000     0050            .LONG   0
                            0054
               00000054     0054            .=MSCP_COMM_AREA+MSU$L_MAX_BUF
                            0054    MAX_BUF:
               00000000     0054            .LONG   0
```

```
          0058
00000058  0058          .=MSCP_COMM_AREA+MSU$L_SMALL
          0058  SMALL:
00000000  0058          .LONG   0
          005C
0000005C  005C          .=MSCP_COMM_AREA+MSU$L_FRACTION
          005C  FRACTION:
00000000  005C          .LONG   0
          0060
00000060  0060          .=MSCP_COMM_AREA+MSU$W_NUM_QUE
          0060  NUM_QUE:
    0000  0060          .WORD   0
          0062
00000062  0062          .=MSCP_COMM_AREA+MSU$W_MAX_QUE
          0062  MAX_QUE:
    0000  0062          .WORD   0
00000000  0064          .LONG   0                               ; Reserved
          0068
00000068  0068          .=MSCP_COMM_AREA+MSU$L_CDSV_SIZE
          0068  CDSV_SIZE:
00000000  0068          .LONG   0
          006C
0000006C  006C          .=MSCP_COMM_AREA+MSU$L_CDRP_SAVE
          006C  CDRP_SAVE:
00000000  006C          .LONG   0
          0070
00000070  0070          .=MSCP_COMM_AREA+MSU$L_CDRP_LIST
          0070  QUEUES: QUE_HEAD CDRP_LIST
00000078  0078          .=MSCP_COMM_AREA+MSU$L_MEM_WAIT
          0078          QUE_HEAD MEM_WAIT
00000080  0080          .=MSCP_COMM_AREA+MSU$L_UQB_LIST
          0080          QUE_HEAD UQB_LIST
00000003  0088  QUE_NUM = <.-QUEUES>/8
          0088          ASSUME MSU$C_NUMQUE EQ QUE_NUM
          0088
00000088  0088          .=MSCP_COMM_AREA+MSU$L_HOST_DSBL
          0088  HOST_DSBL:
00000000  0088          .LONG   0
          008C
0000008C  008C          .=MSCP_COMM_AREA+MSU$L_UNIT_VEC-4
00000000  008C          .LONG   0                               ; *** ENTRY FOR UNIT # -1 !!!!!
          0090  UNIT_VEC:
00000110  0090          .BLKL   2*MSU$K_MAX_UNIT
          0110
00000110  0110          .=MSCP_COMM_AREA+MSU$L_HOST_VEC
          0110  HOST_VEC:
00000190  0110          .BLKL   2*MSU$K_MAX_HOST
          0190
00000190  0190          .=MSCP_COMM_AREA+MSU$Q_CTRL_ID
          0190  CTRL_ID:
00000000  0190          .LONG   0                       ; Controller id
    0000  0194          .WORD   0
      04  0196          .BYTE   MSCP$K_CM_EMULA
      01  0197          .BYTE   MSCP$K_CL_CNTRL
          0198
00000198  0198          .=MSCP_COMM_AREA+MSU$Q_CTRL_INFO
          0198  CTRL_INFO:
```

```
                    0198  VERSION:
              0000  0198          .WORD   0                       ; VERSION
              8004  019A          .WORD   <MSCP$M_CF_REPLC!-      ; controller flags
                    019C  - ; disable advertisement of shadowing   MSCP$M_CF_SHADW!-
                    019C                  MSCP$M_CF_MLTHS>
              FFFF  019C          .WORD   -1                      ; Controller timeout
              0000  019E          .WORD   0                       ; Reserved
                    01A0
          000001A0  01A0          .=MSCP_COMM_AREA+MSU$B_CTRL_DAT
                    01A0  CTRL_DAT:
 00000000 00000000  01A0          .QUAD   0
 00000000 00000000  01A8          .QUAD   0
                    01B0
          000001B0  01B0          .=MSCP_COMM_AREA+MSU$L_OPCOUNT
                    01B0  OPC_COUNT:
          000002B0  01B0          .BLKL   64
                    02B0
          000002B0  02B0          .=MSCP_COMM_AREA+MSU$L_BLKCOUNT
                    02B0  BLK_COUNT:
          000004B4  02B0          .BLKL   129                     ; Statistics for block sizes
                    04B4
          000004B4  04B4          .=MSCP_COMM_AREA+MSU$Q_HQBLIST
                    04B4  HQBLIST:
          000004BC  04B4          .BLKQ   1                       ; List of allocated HQB's
                    04BC
          000004CC  04BC          .=MSCP_COMM_AREA+MSU$K_LENGTH
                    04CC
                    04CC  BUFF_LENGTH:
          000004D0  04CC          .BLKL   1                       ; Length of allocated buffer
                    04D0  BUFF_ADDR:
          000004D4  04D0          .BLKL   1                       ; Address of allocated buffer
                    04D4  BUFF_END:
          000004D8  04D4          .BLKL   1                       ; Length+Address
                    04D8  IO_PRI:
          000004D9  04D8          .BLKB   1
                    04D9
```

```
                      04D9         .SBTTL  MSU ACTION ROUTINES
                      04D9
                      04D9  ;+
                      04D9  ;
                      04D9  ; These routines are called by the MSU program to carry out actions
                      04D9  ; that must be done from high IPL code that lives in non-paged pool.
                      04D9  ;
                      04D9  ; Inputs:
                      04D9  ;
                      04D9  ;      AP -->  #arguments
                      04D9  ;              action code
                      04D9  ;              parameter(s)
                      04D9  ;
                      04D9  ; Outputs:
                      04D9  ;
                      04D9  ;      R0 = status code
                      04D9  ;
                      04D9  ;-
                      04D9
                      04D9         ASSUME  MSU$K_AC_STRT   EQ 1
                      04D9         ASSUME  MSU$K_AC_STOP   EQ 2
                      04D9         ASSUME  MSU$K_AC_ADD    EQ 3
                      04D9         ASSUME  MSU$K_AC_REM    EQ 4
                      04D9         ASSUME  MSU$K_AC_DISC   EQ 5
                      04D9
                      04D9         .ENABLE LSB
                      04D9
                      04D9  MSCP_START:
                      04D9
                      04D9         CASE    4(AP),LIMIT=#1,TYPE=B-
                      04D9         <START,-                            ; Startup
                      04D9          STOP,-                             ; Stop
                      04D9          ADD,-                              ; Add a unit
                      04D9          REMOVE,-                           ; Remove a unit
                      04D9          DISCON>                            ; Disconnect a host
                      04E8
     50    14    3C   04E8         MOVZWL  #SS$_BADPARAM, R0           ; Command out of range
                 05   04EB  10$:   RSB                                 ; Leave
                      04EC
                      04EC  ;
                      04EC  ; START the server
                      04EC  ;
                      04EC
  50   02DC 8F   3C   04EC  START: MOVZWL  #SS$_ILLSEQOP,R0            ; Set an error code
             1F  E2   04F1         BBSS    #MSU$V_START,-              ; Check if already started
    F4 FB4A CF        04F3          STATE,10$
          192E   31   04F7         BRW     START_CONT                  ; Do it
                      04FA
                      04FA         .DISABLE LSB
                      04FA
                      04FA  ;
                      04FA  ; ADD a disk unit
                      04FA  ;
                      04FA         .ENABLE LSB
                      04FA
     50    01    D0   04FA  10$:   MOVL    #1,R0
                 05   04FD         RSB
```

```
             1F     E1  04FE  ADD:     BBC     #MSU$V_START,-               ; Leave if not started yet
     F6 FB3D CF         0500                    STATE,10$
     52    08 AC    D0  0504           MOVL    8(AP),R2                     ; Transfer UQB address
                        0508
                        0508           .DISABLE LSB
                        0508
                        0508  SEND_ATTN:
      7E     56     7D  0508           MOVQ    R6,-(SP)                     ; Unit counter
      56     1F     D0  050B           MOVL    #<MSU$K_MAX_HOST*2>-1,R6     ; Largest host slot index
      57     52     D0  050E           MOVL    R2,R7                        ; UQB address
53    FBFA CF46     D0  0511  10$:     MOVL    HOST_VEC[R6],R3              ; Obtain the HQB pointer
             22     13  0517           BEQL    20$                          ; None, skip on
           171D     30  0519           BSBW    GET_STRUCT                   ; Allocate a packet
          1F 50     E9  051C           BLBC    R0,30$                       ; Branch on failure
     7C A5   53     D0  051F           MOVL    R3,CDRP$L_HQB(R5)            ; Store HQB address
         2C A3      B6  0523           INCW    HQB$W_REFC(R3)               ; Up HQB reference count
     C4 A5   57     D0  0526           MOVL    R7,CDRP$L_UQB(R5)            ; Address of UQB
         20 A5      D4  052A           CLRL    CDRP$L_RSPID(R5)             ; Never send w/RSPID
     54  14 A3      D0  052D           MOVL    HQB$L_CDT(R3),R4             ; Pick up CDT
     24 A5   54     D0  0531           MOVL    R4,CDRP$L_CDT(R5)            ; Set into CDRP
     54  10 A4      D0  0535           MOVL    CDT$L_PDT(R4),R4             ; Now grab the PDT address
             07     10  0539           BSBB    40$                          ; Set up to return synchronously
          D3 56     F4  053B  20$:     SOBGEQ  R6,10$                       ; Iterate over all units
      56     8E     7D  053E  30$:     MOVQ    (SP)+,R6                     ; Restore registers
                    05  0541           RSB
                        0542
     D6 A5   09     B0  0542  40$:     MOVW    #CDRP$K_ST_MSG1_WAIT, -      ; Tag this state
                        0546                   CDRP$W_STATE(R5)
                        0546           ALLOC_MSG_BUF                        ; Allocate a message buffer
             82     D4  0549           CLRL    (R2)+                        ; No command ref number on attn msg
     53  C4 A5      D0  054B           MOVL    CDRP$L_UQB(R5),R3            ; Pick up unit block
     82  28 A3      3C  054F           MOVZWL  UQB$W_UNIT(R3),(R2)+         ; Copy unit number
     82  40 8F      9A  0553           MOVZBL  #64,(R2)+                    ; Set the attn msg code
     82  10 A3      D0  0557           MOVL    UQB$W_MULT_UNT(R3),(R2)+     ; Set mulit_unit & unit_flags
             82     D4  055B           CLRL    (R2)+                        ; Reserved field
     82  18 A3      7D  055D           MOVQ    UQB$Q_UNIT_ID(R3),(R2)+      ; Copy unit identifier
     62  20 A3      D0  0561           MOVL    UQB$L_MEDIA_ID(R3),(R2)      ; Copy media identifier
     51      20     D0  0565           MOVL    #32,R1                       ; Size
                        0568           SEND_CNT_MSG_BUF                     ; Send it out
                        056B  ATTN_CLEANUP:
     53  7C A5      D0  056B           MOVL    CDRP$L_HQB(R5),R3            ; HQB address
         2C A3      B7  056F           DECW    HQB$W_REFC(R3)
           1723     31  0572           BRW     FREE_STRUCT                  ; Release data structure
                        0575  ;
                        0575  ; REMOVE a disk unit
                        0575  ;
                        0575
                    05  0575  REMOVE: RSB
                        0576
                        0576  ;
                        0576  ; DISCONNECT a host
                        0576  ;
                        0576
     53  08 AC      D0  0576  DISCON: MOVL    8(AP),R3                     ; Pick up CDT
     54  10 A3      D0  057A          MOVL    CDT$L_PDT(R3),R4             ; Pick up PDT
           00E9     30  057E          BSBW    VC_ERR                       ; Do the disconnect
     50      01     D0  0581          MOVL    #1,R0                        ; Always success
```

```
05 0584   544          RSB
```

```
                         0585          .SBTTL  LISTEN ACTION ROUTINE
                         0585     ;
                         0585     ; When someone wants to connect with us, control will come here
                         0585     ; as a fork process initiated by the port driver at IPL 8.  If the
                         0585     ; connector is the class driver, we will accept it, otherwise we will
                         0585     ; reject the connection.  Upon a successful acceptance, allocate a
                         0585     ; host control block and wait for a message.
                         0585     ;
                         0585     ;        R2  --> Connect message
                         0585     ;        R3  --> CDT of listen
                         0585     ;        R4  --> PDT
                         0585     ;
                         0585
                         0585     LIS_FORK:
       04    00      EF  0585          EXTZV   #0,#4,-                     ; Pick up low 4 bits of node
    51    20 A3          0588           CDT$B_RSTATION(R3),R1
 07 FAF8 CF    51    E1  058B          BBC     R1,HOST_DSBL,20$            ; Have we disabled this host?
                         0591     10$: REJECT  #1                          ; Yes, don't connect
                     05  0597          RSB                                 ; Return to Port Driver
                         0598
       FAB2 CF    B5     0598     20$: TSTW    NUM_HOST                    ; Any host space left?
             F3    13    059C          BEQL    10$                         ; No, reject this connect
       10    51    91    059E          CMPB    R1,#MSU$K_MAX_HOST          ; Is it a funny one?
             07    1E    05A1          BGEQU   30$                         ; Yes, bring it in line
    FB68 CF41    D5      05A3          TSTL    HOST_VEC[R1]                ; Any one there now?
             10    13    05A8          BEQL    50$                         ; No, okay to use
       51    10    D0    05AA     30$: MOVL    #MSU$K_MAX_HOST,R1          ; Starting place for search
    FB5E CF41    D5      05AD     40$: TSTL    HOST_VEC[R1]                ; Any one there now?
             06    13    05B2          BEQL    50$                         ; Yes
    F5 51    20    F2    05B4          AOBLSS  #MSU$K_MAX_HOST*2,R1,40$;   Try again
             D7    11    05B8          BRB     10$                         ; No room, reject the connect
                         05BA
             51    DD    05BA     50$: PUSHL   R1                          ; Save vector
             53    DD    05BC          PUSHL   R3
 51    0048 8F    3C     05BE          MOVZWL  #HQB$C_LEN, R1              ; Size needed
       008E    30        05C3          BSBW    ALLOMEM                     ; Grab some pool
          53 8ED0        05C6          POPL    R3
       04 50    E8       05C9          BLBS    R0,60$                      ; All set
             8E    D5    05CC          TSTL    (SP)+                       ; Clean stack
             C1    11    05CE          BRB     10$                         ; Reject the connect
                         05D0
                         05D0          ASSUME  HQB$B_FLAG EQ HQB$B_VEC+1
    1E A2    8E    F7    05D0     60$: CVTLW   (SP)+,HQB$B_VEC(R2)         ; Set the vector number & clear flag
       55    52    D0    05D4          MOVL    R2,R5                       ; Transfer address
             82    7C    05D7          CLRQ    (R2)+                       ; Clean out trash
       82    51    B0    05D9          MOVW    R1,(R2)+                    ; Set length
 82    0B60 8F    B0     05DC          MOVW    #<DYN$C_SCS_HQB@8>!-        ; Set type
                         05E1                  DYN$C_SCS,(R2)+
                         05E1     ;
                         05E1     ; ACCEPT the connection and que up the HQB
                         05E1     ;
                         05E1          ACCEPT  MSGADR=MSG_IN,-             ; Message input addr
                         05E1                  DGADR =B^DG_ERR,-           ; DG input
                         05E1                  ERRADR=VC_ERR,-             ; Error address
                         05E1                  INITCR=PACKET,-             ; Initial credit
                         05E1                  MINSCR=#1,-                 ; Minimum send credit
                         05E1                  CONDAT=CTRL_DAT,-           ; CONNECT/ACCEPT data
```

```
                         05E1                    AUXSTR=(R5)             ; Auxiliary structure
            3D 50   E9   060B            BLBC    R0,DEALR0               ; ERROR
       FEA1 CF 65   0E   060E            INSQUE  HQB$L_FLINK(R5),HQBLIST ; Link to list of allocated HQB's
         52  0C A5  DE   0613            MOVAL   12(R5),R2               ; Bump to next
               82   7C   0617            CLRQ    (R2)+
                         0619            ASSUME  HQB$L_CDT EQ 20
            82  53  D0   0619            MOVL    R3,(R2)+                ; Save the CDT address
                82  D4   061C            CLRL    (R2)+                   ; Clear more flags
                         061E            ASSUME  HQB$W_USE_FRAC EQ 28
            04      91   061E            CMPB    #PDT$C_PS,-
            07 A4        0620                      PDT$B_PDT_TYPE(R4)    ; Is it the passthru driver?
               04   12   0622            BNEQ    70$                     ; No
         1F A5  01  90   0624            MOVB    #1,HQB$B_FLAG(R5)       ; Yes, set flag
            28 A5   B4   0628    70$:    CLRW    HQB$W_NUM_QUE(R5)
            2A A5   B4   062B            CLRW    HQB$W_MAX_QUE(R5)
         2C A5  01  B0   062E            MOVW    #1,HQB$W_REFC(R5)       ; Reference count = 1 for CDT
       50  30 A5    9E   0632            MOVAB   HQB$Q_PENDING(R5),R0    ; Cleanup pending queue
            60  50  D0   0636            MOVL    R0,(R0)                 ; Initialize forward pointer
            60  80  DE   0639            MOVAL   (R0)+,(R0)              ; Initialize back pointer
       51  1E A5    9A   063C            MOVZBL  HQB$B_VEC(R5),R1
    FACA CF41  55   D0   0640            MOVL    R5,HOST_VEC[R1]         ; Save the host block address
         FA04 CF    B7   0646            DECW    NUM_HOST                ; Drop number of hosts left
                    05   064A            RSB
                         064B
            50  55  D0   064B    DEALR0: MOVL    R5,R0                   ; Set up pointer
                         064E
      00000000'9F   17   064E    DEALMEM:JMP     @#EXE$DEANONPAGED       ; Get rid of HQB and return
                         0654
      00000000'9F   17   0654    ALLOMEM:JMP     @#EXE$ALONONPAGED       ; Get memory
                         065A    ;
                         065A    ; Error
                         065A    ;
                         065A    DG_ERR: BUG_CHECK MSCPSERV,FATAL        ; ERROR
                         065E
       53  1CE8'CF  D0   065E    STOP:   MOVL    LIS_CDT, R3             ; Pick up the listen cdt
                         0663
                         0663    LIS_ERR:DISCONNECT                      ; Do a disconnect
                    05   0669    RSBX:   RSB                             ; Return
                         066A
                         066A            .DISABLE lsb
```

```
                          066A          .SBTTL  VIRTUAL CIRCUIT ERROR ROUTINE
                          066A
                          066A  ;*****************************************************************************
                          066A  ;                                                                           *
                          066A  ;                                                                           *
                          066A  ;           D I S C O N N E C T   A   V I R T U A L   C I R C U I T         *
                          066A  ;                                                                           *
                          066A  ;                                                                           *
                          066A  ;*****************************************************************************
                          066A  ;
                          066A  ; Return all data structures for this host and break link
                          066A  ;
                          066A  ;         R3 = CDT addr
                          066A  ;         R4 = PDT addr
                          066A  ;
                          066A
                          066A          .ENABLE LSB
                          066A
                          066A  VC_ERR:
            0FD8 8F   BB  066A          PUSHR   #^M<R3,R4,R6,R7,R8,R9,R10,R11> ; Save registers
       58     5C A3   D0  066E          MOVL    CDT$L_AUXSTRUC(R3),R8   ; Pick up HQB
              28 A8   B5  0672          TSTW    HQB$W_NUM_QUE(R8)       ; Are there any CDRP's outstanding?
                 19   13  0675          BEQL    30$                     ; No, leave now
       59        01   D0  0677          MOVL    #1,R9                   ; Init iteration counter
       5A        01   CE  067A  10$:    MNEGL   #1,R10                  ; Yes, Start at unit 0
                 5A   D6  067D  20$:    INCL    R10                     ; Next bit position
       51        1F   9A  067F          MOVZBL  #<2*MSU$K_MAX_UNIT>-1,R1; Max bit size
       51        5A   C2  0682          SUBL    R10,R1                  ; Number left
 5A   10 A8  51  5A   EA  0685          FFS     R10,R1,-
                          068B                  HQB$L_UNIT_ONLN(R8),R10; Find any units online
                 06   12  068B          BNEQ    40$                     ; Got one
              EA 59   F4  068D          SOBGEQ  R9,10$                  ; Finished?
               010C   31  0690  30$:    BRW     FIN_HQB                 ; Yes, leave now
                          0693  ;
                          0693  ; Now run thru the queues on the UQB to find all CDRP's.  NOTE: The order
                          0693  ; in which queues are accessed is important and should not be changed.
                          0693  ;
    5B   F9F8 CF4A    D0  0693  40$:    MOVL    UNIT_VEC[R10],R11       ; Get unit que block
                 E2   13  0699          BEQL    20$                     ; None
       56     70 AB   9E  069B          MOVAB   UQB$Q_SHQ(R11),R6       ; Point to special holding quehead
                 13   10  069F          BSBB    50$                     ; Do the work
       56     68 AB   9E  06A1          MOVAB   UQB$Q_BLOCKQ(R11),R6    ; Point to blocked quehead
                 0D   10  06A5          BSBB    50$                     ; Do the work
              50 AB   B5  06A7          TSTW    UQB$W_NUM_QUE(R11)      ; Any CDRP's queued?
                 D1   13  06AA          BEQL    20$                     ; No
       56     60 AB   9E  06AC          MOVAB   UQB$L_CDRP_FL(R11),R6   ; Point to quehead
                 02   10  06B0          BSBB    50$                     ; Do the work
                 C9   11  06B2          BRB     20$                     ; Try for another unit
                          06B4  ;
                          06B4  ; Subroutine to scan queue and dispatch to appropriate code.
                          06B4  ;
       57        66   D0  06B4  50$:    MOVL    (R6),R7                 ; Get first entry
       56        57   D1  06B7  60$:    CMPL    R7,R6                   ; Is there any more?
                 AD   13  06BA          BEQL    RSBX                    ; No
       55     18 A7   DE  06BC          MOVAL   -CDRP$L_UQB_FL(R7),R5   ; Cover the CDRP portion
       57        67   D0  06C0          MOVL    (R7),R7                 ; Get next entry (later REMQUE zeros flink)
              14 A8   D1  06C3          CMPL    HQB$L_CDT(R8),-         ; Belong to this host?
```

```
         60 A5     06C6                    CDRP$L_CDTX(R5)
            ED  12 06C8            BNEQ    60$                      ; No
                   06CA    ;
                   06CA    ; Dispatch on the state value.  This is changed each time the CDRP goes
                   06CA    ; thru a particular routine.  Each state may need different treatment.
                   06CA    ;
            10  88 06CA            BISB    #1@CDRP$V_ABORT,-        ; Set 'ABORTED' bit
         FA A5     06CC                    CDRP$B_FLAGS(R5)
   53    14 A8  D0 06CE            MOVL    HQB$L_CDT(R8),R3         ; Pick up the cdt address in r3
   50    D6 A5  9A 06D2            MOVZBL  CDRP$W_STATE(R5),R0      ; Pick up state
                   06D6            CASE    R0,-
                   06D6                    <ST_MEM_WAIT,-           ; 0-Memory wait
                   06D6                     ST_MAP_WAIT,-           ; 1-MAP resource wait
                   06D6                     ST_DRV_WAIT,-           ; 2-Driver que wait
                   06D6                     ST_DRV_WAIT,-           ; 3-DATA command wait
                   06D6                     ST_DRV_WAIT,-           ; 4-Driver que wait
                   06D6                     ST_MSG_WAIT,-           ; 5-MSG send wait
                   06D6                     ST_SHQ_QUE,-            ; 6-Special holding que
                   06D6                     ST_BLK_QUE,-            ; 7-Blocked que
                   06D6                     ST_ONBLK_QUE>           ; 8-Online-copy Blocked que
                   06EC
                   06EC            BUG_CHECK MSCPSERV,FATAL         ; ERROR
                   06F0
                   06F0    65$:    BUG_CHECK MSCPSERV,FATAL         ; ERROR -- connection / CDT mismatch
                   06F4
                   06F4    ;
                   06F4    ; Driver wait:  The CDRP has been delivered to the disk driver and is
                   06F4    ; awaiting completion.  It has a mapping resource, a buffer, and a
                   06F4    ; message buffer.
                   06F4    ;
                   06F4    ST_DRV_WAIT:
         C0 59  E8 06F4            BLBS    R9,60$                   ; Skip first time
   52    B0 A5  D0 06F7            MOVL    CDRP$L_MSGBUFX(R5),R2    ; Pick up message buffer
F8 A2    18 A3  D1 06FB            CMPL    CDT$L_LCONID(R3), -      ; Check validity of remote conid
                   0700                    SCS$L_DST_CONID(R2)
            EE  12 0700            BNEQ    65$                      ; Trying to respond to stale connection
         B0 A5  D4 0702            CLRL    CDRP$L_MSGBUFX(R5)       ; Show it gone
                   0705            DEALLOC_MSG_BUF_REG              ; Deallocate it
   D6 A5    03  B1 0708            CMPW    #3,CDRP$W_STATE(R5)      ; Is it data wait?
            A9  12 070C            BNEQ    60$                      ; No, allow it to finish
                   070E
                   070E    ;
                   070E    ; Send/Request data wait:  The CDRP is somewhere in the process of doing
                   070E    ; a send/request data operation.  It has a mapping resource, a buffer, and
                   070E    ; possibly a RSPID and a message buffer.
                   070E    ;
                   070E    ST_DAT_WAIT:
         20 A5  D5 070E            TSTL    CDRP$L_RSPID(R5)         ; No, if there is a RSPID
            06  13 0711            BEQL    70$                      ; .. then
                   0713            DEALLOC_RSPID                    ; Deallocate it
         1C A5  D5 0719    70$:    TSTL    CDRP$L_MSG_BUF(R5)       ; If there is something
            03  13 071C            BEQL    80$                      ; .. then
                   071E            DEALLOC_MSG_BUF                  ; Deallocate the message buffer
                   0721    80$:
                   0721    ; ??? Crock that assumes that this field is initialized!
      50    65  D0 0721            MOVL    CDRP$L_FQFL(R5),R0       ; Get possible next element
                   0724            ASSUME  CDRP$L_FQFL EQ 0
```

```
55    04 A0   D1  0724          CMPL    4(R0),R5                ; Is this block queued to anything?
         03   12  0728          BNEQ    90$                     ; No
   50    65   0F  072A          REMQUE  CDRP$L_FQFL(R5),R0      ; Remove from wait que
                  072D  ; ??? End crock
         50   D4  072D  90$:    CLRL    R0                      ; Set for failure
   7E    53   7D  072F          MOVQ    R3,-(SP)                ; Save CDT,PDT
      0C B5   16  0732          JSB     @CDRP$L_FPC(R5)         ; Finish it off
   53    8E   7D  0735          MOVQ    (SP)+,R3                ; Restore
       FF7C   31  0738          BRW     60$                     ; Loop for next CDRP
                  073B
                  073B  ;
                  073B  ; Memory wait:  The CDRP is waiting for a data buffer.  It has no resources
                  073B  ; and can be immediately cleaned up after being dequeued from memory wait.
                  073B  ;
                  073B  ST_MEM_WAIT:
   F921 CF    B7  073B          DECW    NUM_QUE
                  073F
                  073F  ;
                  073F  ; Message wait:  The CDRP is on a credit wait queue from a RECYCL_MSG_BUF
                  073F  ; operation.  It has a buffer and a message buffer.  It can be dequeued
                  073F  ; from the credt wait queue.
                  073F  ;
                  073F  ST_MSG_WAIT:
                  073F
                  073F  ;
                  073F  ; Map wait:  The CDRP is waiting for mapping resources.  It has a data
                  073F  ; buffer resource only.  It can be immediately cleaned up after being dequeued
                  073F  ; from the map wait queue.
                  073F  ;
                  073F  ST_MAP_WAIT:
   50    65   0F  073F          REMQUE  CDRP$L_FQFL(R5),R0      ; Remove from wait que
         2D   11  0742          BRB     150$                    ;  and finish off
                  0744
                  0744  ;
                  0744  ; Special holding que:  The CDRP is waiting for CIP.  It has no resources
                  0744  ; and can be immediately cleaned up after being dequeued.
                  0744  ;
                  0744  ST_SHQ_QUE:
50    E8 A5   0F  0744          REMQUE  CDRP$L_UQB_FL(R5),R0    ; Remove from special holding que
         04   12  0748          BNEQ    100$                    ; Not the last entry
         08   8A  074A          BICB    #UQB$M_CMDPEND,-        ; Clear the command pending status
      2B AB       074C                  UQB$B_FLAGS(R11)
         60   D4  074E  100$:   CLRL    (R0)                    ; Show not queued
         28   11  0750          BRB     155$                    ; Done
                  0752
                  0752  ;
                  0752  ; Blocked que:  The CDRP is blocked.  It has a message buffer and possibly
                  0752  ; a buffer. It can be immediately cleaned up after being dequeued.
                  0752  ;
                  0752  ST_BLK_QUE:
      71'AF   9F  0752          PUSHAB  B^150$                  ; Set return address
      7C AB   B7  0755  120$:   DECW    UQB$W_NUM_BLK(R11)      ; Drop the blocked count
50    E8 A5   0F  0758          REMQUE  CDRP$L_UQB_FL(R5),R0    ; Remove from blocked queue
         04   12  075C          BNEQ    130$                    ; Not the last entry
         02   8A  075E          BICB    #UQB$M_BLOCKED,-        ; None left, clear flag
      2B AB       0760                  UQB$B_FLAGS(R11)
         60   D4  0762  130$:   CLRL    (R0)                    ; Clear link
```

```
                   05  0764         RSB                             ; Leave now
                       0765
                       0765  ;
                       0765  ; Online-copy blocked que:  The CDRP is blocked.  It has no resources
                       0765  ; and can be immediately cleaned up after being dequeued.
                       0765  ;
                       0765  ST_ONBLK_QUE:
          30 59    E8  0765         BLBS    R9,160$                 ; If second time, then
             EB    10  0768         BSBB    120$                    ;  clean up
    53    10 A5    7D  076A         MOVQ    CDRP$L_FR3(R5),R3       ; Pick up registers
          0B1B     30  076E         BSBW    CLEAN_UP_ONLCPY         ; Do the special clean up
                       0771
                       0771  ;
                       0771  ; Clean up the CDRP and deallocate resources.
                       0771  ;
    09    FB A5    91  0771  150$:  CMPB    CDRP$B_OPCODE(R5), -    ; If non-sequential command
                       0775                 #MSCP$K_OP_ONLIN
          03       1B  0775         BLEQU   155$                    ;  then
                       0777         UNMAP                           ; Get rid of possible map resource
    52    B0 A5    D0  077A  155$:  MOVL    CDRP$L_MSGBUFX(R5),R2   ; Pick up message buffer
             0D    13  077E         BEQL    157$                    ; Just in case!
 F8 A2    18 A3    D1  0780         CMPL    CDT$L_LCONID(R3), -     ; Check validity of remote conid
                       0785                 SCS$L_DST_CONID(R2)
             14    12  0785         BNEQ    165$                    ; Trying to respond to stale connection
          B0 A5    D4  0787         CLRL    CDRP$L_MSGBUFX(R5)      ; Show it gone
                       078A         DEALLOC_MSG_BUF_REG             ; Deallocate it
       53    58    D0  078D  157$:  MOVL    R8,R3                   ; Set HQB address
             54    DD  0790         PUSHL   R4                      ; Save PDT
          033F     30  0792         BSBW    SEND_PKT                ; Clean up the packet
          54 8ED0      0795         POPL    R4                      ; Restore PDT
          FF1C     31  0798  160$:  BRW     60$                     ; Continue
                       079B
                       079B  165$:  BUG_CHECK MSCPSERV,FATAL        ; ERROR
                       079F
                       079F         .DISABLE LSB
                       079F  ;
                       079F  ; Clean up of CDRP's all done. Now finish up UNIT and HOST blocks.
                       079F  ;
                       079F  FIN_HQB:
       5A    01    CE  079F         MNEGL   #1,R10                  ; Yes, Start at unit 0
       53    58    D0  07A2  10$:   MOVL    R8,R3                   ; Transfer HQB address
             5A    D6  07A5         INCL    R10                     ; Next bit position
       51    1F    9A  07A7         MOVZBL  #<2*MSU$K_MAX_UNIT>-1,R1; Max bit size
       51    5A    C2  07AA         SUBL    R10,R1                  ; Number left
 5A   10 A8  51    5A  EA  07AD     FFS     R10,R1,-
                       07B3                 HQB$L_UNIT_ONLN(R8),R10; Find any units online
             32    13  07B3         BEQL    40$                     ; No units alive
    54   F8D6 CF4A D0  07B5         MOVL    UNIT_VEC[R10],R4        ; Get unit que block
             05    12  07BB         BNEQ    20$                     ; Got one
  E0 10 A3   5A    E4  07BD         BBSC    R10,-                   ; Clear the bogus bit
                       07C2                 HQB$L_UNIT_ONLN(R3),10$
          1474     30  07C2  20$:   BSBW    GET_STRUCT              ; Get data structure
          04 50    E8  07C5         BLBS    R0,30$                  ; Branch on success
                       07C8         BUG_CHECK MSCPSERV,FATAL        ; It failed
                       07CC
    7C A5    53    D0  07CC  30$:   MOVL    R3,CDRP$L_HQB(R5)       ; Address of HQB
          2C A3    B6  07D0         INCW    HQB$W_REFC(R3)          ; Upreference count
```

```
        B0 A5   D4 07D3         CLRL    CDRP$L_MSGBUFX(R5)      ; Prevent spurious unloads
     F8 A5 5A   D0 07D6         MOVL    R10,CDRP$W_SLOT(R5)     ; Clean out flags and others
           08   90 07DA         MOVB    #MSCP$K_OP_AVAIL,-      ; Set the command
        FB A5      07DC                  CDRP$B_OPCODE(R5)
     78 A5 54   D0 07DE         MOVL    R4,CDRP$L_MST_UQB(R5)   ; Needed by routine
         03BB   30 07E2         BSBW    SEQ                     ; Go do the AVAIL
           BB   11 07E5         BRB     10$                     ; Loop until done
                   07E7
                   07E7 40$:
                   07E7 ; ??? Insert bug trapping code here to find live references to the now stale CDT
     53   14 A8 D0 07E7         MOVL    HQB$L_CDT(R8),R3        ; CDT to look up
     54   10 A3 D0 07EB         MOVL    CDT$L_PDT(R3),R4        ; PDT address
     50   0F'AF 9E 07EF         MOVAB   B^100$,R0               ; Action routine
  00000000'GF   16 07F3         JSB     G^SCS$LKP_RDTCDRP       ; Look for matching CDRPs on RDT
     50   0F'AF 9E 07F9         MOVAB   B^100$,R0               ; Action routine
  00000000'GF   16 07FD         JSB     G^SCS$LKP_RDTWAIT       ; Look for matching CDRPs on RSPID wait queu
     50   0F'AF 9E 0803         MOVAB   B^100$,R0               ; Action routine
  00000000'GF   16 0807         JSB     G^SCS$LKP_MSGWAIT       ; Look for matching CDRPs on message send qu
           04   11 080D         BRB     110$
                   080F
                   080F 100$:   BUG_CHECK       MSCPSERV,FATAL  ; Found dangling CDT reference
                   0813
                   0813 110$:
                   0813 ; ??? End of bug traps
        55 58   D0 0813         MOVL    R8,R5                   ; Hold HQB address
      0FD8 8F   BA 0816         POPR    #^M<R3,R4,R6,R7,R8,R9,R10,R11>
                   081A         DISCONNECT
                   0820 ; ??? more bug traps
     14 A5 01   D0 0820         MOVL    #1,HQB$L_CDT(R5)        ; zap CDT address
                   0824 ; ??? end of more bug traps
     51 1E A5   9A 0824         MOVZBL  HQB$B_VEC(R5),R1        ; Pick up vect number
    F8E3 CF41   D4 0828         CLRL    HOST_VEC[R1]            ; Show us gone
      F81D CF   B6 082D         INCW    NUM_HOST                ; Up number of host slot left
        53 55   D0 0831         MOVL    R5,R3                   ; HQB address
        2C A3   B7 0834         DECW    HQB$W_REFC(R3)          ; Decrement reference count of HQB
           06   12 0837         BNEQ    120$                    ; Not yet 0 -- we never deallocate in this c
        50 63   0F 0839         REMQUE  HQB$L_FLINK(R3),R0      ; Remove from list headed by MSU$Q_HQBLIST
         FE0F   30 083C         BSBW    DEALMEM                 ; Get rid of HQB
                05 083F 120$:   RSB
                   0840
```

```
                        0840           .SBTTL  WAIT FOR INT
                        0840
                        0840     ;+
                        0840     ;
                        0840     ; The resuming address and R3/R4 are saved in the CDRP.  Control
                        0840     ; is returned to caller's caller.
                        0840     ;
                        0840     ; Input:
                        0840     ;         R3 -->  HQB
                        0840     ;         R4 -->  UQB
                        0840     ;         R5 -->  CDRP
                        0840     ;-
                        0840
                        0840           .ENABLE LSB
                        0840
                        0840     BLOCKEDH:
 68 A4   E8 A5     0E   0840           INSQUE  CDRP$L_UQB_FL(R5), -      ; Insert this packet on blocked que
                        0845                   UQB$Q_BLOCKQ(R4)          ;  at head
           05      11   0845           BRB     10$
                        0847
                        0847     BLOCKEDT:
 6C B4   E8 A5     0E   0847           INSQUE  CDRP$L_UQB_FL(R5), -      ; Insert this packet on blocked que
                        084C                   @UQB$Q_BLOCKQ+4(R4)       ;  at tail
   D6 A5   07      B0   084C     10$:  MOVW    #7,CDRP$W_STATE(R5)       ; Ditto
           02      88   0850           BISB    #UQB$M_BLOCKED,-          ; Set blocked flag
        2B A4           0852                   UQB$B_FLAGS(R4)
        7C A4      B6   0854           INCW    UQB$W_NUM_BLK(R4)         ; Bump number of packet queue to unit
        7C A4      B1   0857           CMPW    UQB$W_NUM_BLK(R4),-       ; Is this number <= max?
        7E A4           085A                   UQB$Q_MAX_BLK(R4)
           05      1B   085C           BLEQU   WAIT_FOR_INT              ; Yes, skip
        7C A4      B0   085E           MOVW    UQB$Q_NUM_BLK(R4),-       ; No, set new max
        7E A4           0861                   UQB$Q_MAX_BLK(R4)
                        0863                                             ; And fall thru to wait
                        0863     WAIT_FOR_INT:
   10 A5   53      7D   0863           MOVQ    R3,CDRP$L_FR3(R5)         ; Save registers
      0C A5 8ED0        0867           POPL    CDRP$L_FPC(R5)            ; Store resume point
                   05   086B           RSB                               ; Return to caller's caller
                        086C
                        086C           .DISABLE LSB
```

```
                            086C          .SBTTL  MAIN LINE ROUTINE
                            086C
                            086C   ;+
                            086C   ;
                            086C   ; This is the main line routine that does all the work.
                            086C   ;
                            086C   ; Inputs:
                            086C   ;
                            086C   ;          R1  =   Length of message
                            086C   ;          R2  --> Message
                            086C   ;          R3  --> CDT
                            086C   ;          R4  --> PDT
                            086C   ;
                            086C   ;-
                            086C
                            086C          .ENABLE LSB
                            086C
                 18    BB   086C   10$:   PUSHR   #^M<R3,R4>               ; Save vital registers
                            086E          DEALLOC_MSG_BUF_REG              ; Deallocate message buffer
                 18    BA   0871          POPR    #^M<R3,R4>               ; Restore registers
               FDF4    31   0873          BRW     VC_ERR                   ; Break connection and return
                            0876
                            0876   MSG_IN:
               13C0    30   0876          BSBW    GET_STRUCT               ; Get data structure
             F0  50    E9   0879          BLBC    R0,10$                   ; Failed to get one
             F930  CF  D6   087C          INCL    OPC_COUNT                ; Grand total of requests
         B0 A5   52    D0   0880          MOVL    R2,CDRP$L_MSGBUFX(R5)    ; Store the message pointer
              1C A5    D4   0884          CLRL    CDRP$L_MSG_BUF(R5)       ; Mark it not in use
         24 A5   53    D0   0887          MOVL    R3,CDRP$L_CDT(R5)        ;  and the CDT
         60 A5   53    D0   088B          MOVL    R3,CDRP$L_CDTX(R5)       ; Store CDT in non-volatile cell
         53   5C A3    D0   088F          MOVL    CDT$L_AUXSTRUC(R3),R3    ; Pick up the HQB address
         7C A5   53    D0   0893          MOVL    R3,CDRP$L_HQB(R5)        ; Store HQB address
              2C A3    B6   0897          INCW    HQB$W_REFC(R3)           ; Up reference count
              28 A3    B6   089A          INCW    HQB$W_NUM_QUE(R3)        ; Bump number of packet queue to host
              28 A3    B1   089D          CMPW    HQB$W_NUM_QUE(R3),-      ; Is this number <= max?
              2A A3         08A0                   HQB$Q_MAX_QUE(R3)
                 05    1B   08A2          BLEQU   30$                      ; Yes, skip
              28 A3    B0   08A4          MOVW    HQB$W_NUM_QUE(R3),-      ; No, set new max
              2A A3         08A7                   HQB$Q_MAX_QUE(R3)
                            08A9   ;
                            08A9   ; Check the MSCP command packet for protocol errors
                            08A9   ;
           0C    51    B1   08A9   30$:   CMPW    R1,#MSCP$K_MIN_SIZE      ; Is it the min length?
                 17    19   08AC          BLSS    40$                      ; No, error
         50   08 A2    9A   08AE          MOVZBL  MSCP$B_OPCODE(R2),R0     ; Pick up the opcode
                 75    13   08B2          BEQL    ERR_1                    ; None is error
         40 8F   50    91   08B4          CMPB    R0,#MSU$K_MAX_OPC        ; Out of valid range?
                 6F    14   08B8          BGTR    ERR_1                    ; Yes, error
      54   1CFC'CF40   9A   08BA          MOVZBL  COM_PKT_LEN[R0],R4       ; Get expected length
                 67    13   08C0          BEQL    ERR_1                    ; Invalid command
           54    51    B1   08C2          CMPW    R1,R4                    ; Big enough?
                 68    19   08C5   40$:   BLSS    ERR_2                    ; No, error
   0A A2   1D4C'CF40   B3   08C7          BITW    MOD_TBL[R0], -
                            08CE                   MSCP$W_MODIFIER(R2)     ; Check allowable
                 63    12   08CE          BNEQ    ERR_3                    ; Non-allow bit set, error
         F4 A5   02    CE   08D0          MNEGL   #2,CDRP$L_CMD_STS(R5)    ; Set up initial cmd status
                            08D4   ;
```

```
                              08D4  ; Grab the unit number and try to find a UQB for it
                              08D4  ;
              51   52    D0   08D4          MOVL    R2,R1                   ; Transfer to a better reg
          52    04 A1    3C   08D7          MOVZWL  MSCP$W_UNIT(R1),R2      ; Pick up unit number from command
              50   52    D0   08DB          MOVL    R2,R0                   ; Copy
   54    52  FFFFFFF0 8F CB   08DE          BICL3   #^C<<1@MSU$K_UNIT_SIZ>-1>,R2,R4 ; Slot
      52    F7A5 CF44    D0   08E6          MOVL    UNIT_VEC[R4],R2         ; Look for unit block in slot
                   07    13   08EC          BEQL    50$                     ; Empty
                04 A1    B1   08EE          CMPW    MSCP$W_UNIT(R1),-
                28 A2         08F1                   UQB$W_UNIT(R2)         ; Hit, is it a match?
                   04    13   08F3          BEQL    60$                     ; Yes
                   65    10   08F5  50$:    BSBB    FIND_SLOT               ; No, try more probing
                   54    19   08F7          BLSS    NOT_FOUND               ; Still did not find it
          78 A5    52    D0   08F9  60$:    MOVL    R2,CDRP$L_MST_UQB(R5)   ; Save UQB pointer
          F8 A5    54    3C   08FD  65$:    MOVZWL  R4,CDRP$W_SLOT(R5)      ; Save slot in CDRP and zero FLAGS
       50    08 A1       9A   0901          MOVZBL  MSCP$B_OPCODE(R1),R0    ; Pick up the opcode
          FB A5    50    90   0905          MOVB    R0,CDRP$B_OPCODE(R5)    ; Save opcode in CDRP
             F8A2 CF40   D6   0909          INCL    OPC_COUNT[R0]           ; Count the opcode
          04 1F    A3    E9   090E          BLBC    HQB$B_FLAG(R3),70$      ; Is this the passthru driver?
                   20    88   0912          BISB    #1@CDRP$V_NOHSTBUF,-    ; Yes, set flag
                FA A5         0914                   CDRP$B_FLAGS(R5)
                              0916  ;
                              0916  ; Split up command into class then dispatch on it.
                              0916  ;
     50    50    FD 8F   78   0916  70$:    ASHL    #-3,R0,R0               ; Pick up class field
                              091B          CASE    R0,-                    ; Dispatch on class
                              091B                  <IMMED,-                ;  0 - IMMEDIATE
                              091B                   SEQ,-                  ;  8 - SEQUENTIAL
                              091B                   NSEQ,-                 ; 16 - NON-SEQUENTIAL
                              091B                   MAINT,-                ; 24 - MAINTENANCE
                              091B                   NSEQB>                 ; 32 - NON-SEQUENTIAL/BUFFER
                              0929  ;
                              0929  ; Error paths
                              0929  ;
          50    08   18  9C   0929  ERR_1:  ROTL    #24,#MSCP$B_OPCODE,R0   ; Opcode error
                     08  11   092D          BRB     ERR_X
                              092F  ;
                     50  D4   092F  ERR_2:  CLRL    R0                      ; Illegal length
                     04  11   0931          BRB     ERR_X
                              0933  ;
          50    0A   18  9C   0933  ERR_3:  ROTL    #24,#MSCP$W_MODIFIER,R0 ; Modifier error
             52    B0 A5 D0   0937  ERR_X:  MOVL    CDRP$L_MSGBUFX(R5),R2   ; Recover message buffer
                         C9   093B          BISL3   #<MSCP$K_ST_ICMD@16-
                              093C                   !MSCP$K_OP_END>,-      ; Set invalid
  08 A2    50   00010080 8F   093C                   R0,MSCP$B_OPCODE(R2)
                   78 A5 D4   0944          CLRL    CDRP$L_MST_UQB(R5)      ; No master UQB
               51     0C D0   0947          MOVL    #MSCP$K_MIN_SIZE,R1     ; Length of message
                   01A0  31   094A          BRW     SEND_END
                              094D  ;
                              094D  NOT_FOUND:
                   78 A5 D4   094D          CLRL    CDRP$L_MST_UQB(R5)      ; Zero out UQB pointer
                   08 A1 91   0950          CMPB    MSCP$B_OPCODE(R1),-     ; Command need unit?
                      08      0953                   #MSCP$K_OP_AVAIL
                      A7 1F   0954          BLSSU   65$                     ; No, allow to continue
               50     03 D0   0956          MOVL    #MSCP$K_ST_OFFLN,R0     ; Yes, set failure status
                   0178  31   0959          BRW     SEND_PKT                ; And leave
                              095C
```

```
                          095C        .DISABLE LSB
                          095C   ;
                          095C   ; Routine to find slot number from unit number
                          095C   ;
                          095C   ;       R0 = Unit number to match against
                          095C   ;
                          095C   FIND_SLOT:
            54    10  C0  095C          ADDL    #MSU$K_MAX_UNIT,R4        ; Try for second probe
  52   F72C CF44      D0  095F          MOVL    UNIT_VEC[R4],R2           ; Look for unit block in slot
                  06  13  0965          BEQL    60$                       ; Empty
         28 A2    50  B1  0967          CMPW    R0,UQB$W_UNIT(R2)         ; Hit, is it a match?
                  1B  13  096B          BEQL    100$                      ; Yes
            54    1F  D0  096D   60$:   MOVL    #<2*MSU$K_MAX_UNIT>-1,R4; Sequential search
  52   F71B CF44      D0  0970   70$:   MOVL    UNIT_VEC[R4],R2           ; Pick up unit
                  06  13  0976          BEQL    80$                       ; Empty
         28 A2    50  B1  0978          CMPW    R0,UQB$W_UNIT(R2)         ; Hit, is it a match?
                  0A  13  097C          BEQL    100$                      ; Yes
                  54  D7  097E   80$:   DECL    R4                        ; Loop thru
            10    54  D1  0980          CMPL    R4,#MSU$K_MAX_UNIT        ; Reached end?
                  EB  18  0983          BGEQ    70$                       ; No, try again
            54    01  CE  0985   90$:   MNEGL   #1,R4                     ; Set a failure indicator
                      05  0988   100$:  RSB
```

```
                0989          .SBTTL  IMMEDIATE class commands
                0989  ;
                0989  ;       R0    scratch
                0989  ;       R1 = Ptr to message buffer
                0989  ;       R2 = low 3 bits of command
                0989  ;       R3 = Ptr to HQB
                0989  ;       R4 = Unit number
                0989  ;       R5 = Ptr to CDRP
                0989  ;
                0989  IMMED:  CASE    CDRP$B_OPCODE(R5),-      ; Dispatch on command subtype
                0989                    TYPE=B,LIMIT=#MSCP$K_OP_ABORT-
                0989                  <ABORT,-                  ; 1-ABORT
                0989                   GET_CMD_STS,-            ; 2-GET COMMAND STATUS
                0989                   GET_UNIT_STS,-           ; 3-GET UNIT STATUS
                0989                   SET_CTRL_CHR>            ; 4-SET CONTROLLER CHARACTERISTICS
      91   11   0996          BRB     ERR_T                     ; Illegal command
```

```
                  0998          .SBTTL  -          1 - ABORT
                  0998  ;*****************************************************
                  0998  ;                                                    *
                  0998  ;        #1  -  A B O R T                            *
                  0998  ;                                                    *
                  0998  ;*****************************************************
                  0998  ;
                  0998  ABORT:
       121D  30   0998          BSBW    SEARCH_QUE              ; Find the entry
      73 50  E9   099B          BLBC    R0,SS                   ; Failed to find it
                  099E          CASE    CDRP$W_STATE-CDRP$L_UQB_FL(R1),TYPE=W-
                  099E                  <10$,-                  ; 0 - Mem wait
                  099E                   20$,-                  ; 1 - Map wait
                  099E                   40$,-                  ; 2 - Driver que wait
                  099E                   40$,-                  ; 3 - Data command wait
                  099E                   40$,-                  ; 4 - Driver que wait
                  099E                   20$,-                  ; 5 - Msg send wait
                  099E                   50$,-                  ; 6 - Special holding que
                  099E                   70$,-                  ; 7 - Blocked que
                  099E                   70$>                   ; 8 - Online-copy blocked que
                  09B5
                  09B5          BUG_CHECK  MSCPSERV,FATAL       ; ERROR
                  09B9  ;
                  09B9  ; CDRP was on the memory wait queue.  Remove it and post it.
                  09B9  ;
   F6A3 CF   B7   09B9  10$:    DECW    NUM_QUE
                  09BD  ;
                  09BD  ; CDRP was on the map resource wait queue.  Remove it and post it.
                  09BD  ;
 50   18 A1  0F   09BD  20$:    REMQUE  CDRP$L_FQFL- -
                  09C1                  CDRP$L_UQB_FL(R1),R0    ; Remove from wait queue
      18 A1  9F   09C1  30$:    PUSHAB  -CDRP$L_UQB_FL(R1)      ; Push addr of CDRP to be aborted
         4B  10   09C4          BSBB    SS                      ; Return the ABORT end packet
         55 8ED0  09C6          POPL    R5                      ; Recover aborted CDRP pointer
 09   FB A5  91   09C9          CMPB    CDRP$B_OPCODE(R5), -    ; If non-sequential command
                  09CD                  #MSCP$K_OP_ONLIN
         03  1B   09CD          BLEQU   35$                     ;  then
                  09CF          UNMAP                           ; Get rid of possible map resource
   50    02  D0   09D2  35$:    MOVL    #MSCP$K_ST_ABRTD, R0    ; Set status
 52   B0 A5  D0   09D5          MOVL    CDRP$L_MSGBUFX(R5),R2   ; Recover message pointer
      E0 A5  D0   09D9          MOVL    CDRP$L_ABCNT(R5),-
      0C A2       09DC                  MSCP$L_BYTE_CNT(R2)     ; Show acumulated byte count
       00F3  31   09DE          BRW     SEND_PKT                ; Return the command end packet
                  09E1  ;
                  09E1  ; CDRP was a activity pending queue.  We cannot arbitrarily remove it.
                  09E1  ; Therefore, just set the "ABORT" bit and let the action routine deal with
                  09E1  ; it when it returns.
                  09E1  ;
         10  88   09E1  40$:    BISB    #1@CDRP$V_ABORT,-       ; Set 'ABORTED' bit
      12 A1       09E3                  CDRP$B_FLAGS-CDRP$L_UQB_FL(R1)
         2A  11   09E5          BRB     SS                      ; Leave with success
                  09E7  ;
                  09E7  ; CDRP was on the special holding queue.  Remove it and post it.
                  09E7  ;
   50    61  0F   09E7  50$:    REMQUE  (R1),R0                 ; Remove from special holding que
         04  12   09EA          BNEQ    60$                     ; Not the last entry
         08  8A   09EC          BICB    #UQB$M_CMDPEND,-        ; Clear the command pending status
```

```
      2B A4       09EE  1164                  UQB$B_FLAGS(R4)
         61  D4   09F0  1165  60$:   CLRL     (R1)                  ; Show not queued
         CD  11   09F2  1166         BRB      30$                   ; Finish up
                  09F4  1167  ;
                  09F4  1168  ; CDRP was on the blocked queue.  Remove it and post it.
                  09F4  1169  ;
      7C A4  B7   09F4  1170  70$:   DECW     UQB$W_NUM_BLK(R4)     ; Drop the blocked count
   50    61  0F   09F7  1171         REMQUE   (R1),R0               ; Remove from blocked queue
         F4  12   09FA  1172         BNEQ     60$                   ; Not the last entry
         02  8A   09FC  1173         BICB     #UQB$M_BLOCKED,-      ; Clear blocked flag
      2B A4       09FE  1174                  UQB$B_FLAGS(R4)
         EE  11   0A00  1175         BRB      60$                   ; And finish up
```

```
                        0A02          .SBTTL  -       2 - GET COMMAND STATUS
                        0A02    ;*****************************************************
                        0A02    ;                                                     *
                        0A02    ;       # 2  -  G E T   C M D   S T S                 *
                        0A02    ;                                                     *
                        0A02    ;*****************************************************
                        0A02    ;
                        0A02    GET_CMD_STS:
          11B3    30    0A02            BSBW    SEARCH_QUE                 ; Find entry
         09 50    E9    0A05            BLBC    R0, SS                     ; None, always return success
    54   B0 A5    D0    0A08            MOVL    CDRP$L_MSGBUFX(R5), R4     ; Pick up buffer address
         0C A1    D0    0A0C            MOVL    CDRP$L_CMD_STS-CDRP$L_UQB_FL(R1),- ; Found it,
         10 A4          0A0F             MSCP$L_CMD_STS(R4)        ;  copy the status code
          00BE    31    0A11    SS:     BRW     SEND_SUCCESS
```

```
                          0A14        .SBTTL  -       3 - GET UNIT STATUS
                          0A14  ;*******************************************************
                          0A14  ;                                                      *
                          0A14  ;          #3  -  G E T   U N I T   S T S             *
                          0A14  ;                                                      *
                          0A14  ;*******************************************************
                          0A14  ;
                          0A14  GET_UNIT_STS:
                          0A14          ASSUME  MSCP$V_MD_NXUNT EQ 0
        2E 0A A1    E9    0A14          BLBC    MSCP$W_MODIFIER(R1),50$ ; If not NXTUNT, do staight processing
     52    04 A1    3C    0A18          MOVZWL  MSCP$W_UNIT(R1),R2      ; Pick up orginal unit number
  54     F660 CF    DE    0A1C          MOVAL   UQB_LIST,R4             ; Point to head of list
        54    64    D1    0A21          CMPL    (R4),R4                 ; Queue empty?
              0B    13    0A24          BEQL    20$                     ; Yes, return OFFLINE failure, unit=0
        50    54    D0    0A26          MOVL    R4,R0                   ; No, copy starting point
        54    64    D0    0A29  10$:    MOVL    (R4),R4                 ; Get next entry
        50    54    D1    0A2C          CMPL    R4,R0                   ; End of line?
              09    12    0A2F          BNEQ    40$                     ; No, continue
           04 A1    B4    0A31  20$:    CLRW    MSCP$W_UNIT(R1)         ; Yes, give special code (Unit 0)
        50    03    D0    0A34  30$:    MOVL    #MSCP$K_ST_OFFLN,R0     ; Set for OFFLINE
            009A    31    0A37          BRW     SEND_PKT                ; Return end packet
                          0A3A
     A8 A4    52    B1    0A3A  40$:    CMPW    R2,UQB$W_UNIT- -
                          0A3E                   UQB$Q_UNITQ(R4)        ; Compare unit numbers
              E9    1A    0A3E          BGTRU   10$                     ; Not far enough, keep looping
     54    80 A4    DE    0A40          MOVAL   -UQB$Q_UNITQ(R4),R4     ; Set R4 to head of UQB
              08    11    0A44          BRB     60$                     ; Done
                          0A46
  54   F645 CF44    D0    0A46  50$:    MOVL    UNIT_VEC[R4], R4        ; Get pointer to UQB
              E6    13    0A4C          BEQL    30$                     ; None, return status
                          0A4E  ;
                          0A4E  ; Found UQB for unit, transfer information
                          0A4E  ;
           28 A4    B0    0A4E  60$:    MOVW    UQB$W_UNIT(R4),-
           04 A1          0A51                   MSCP$W_UNIT(R1)        ; Return the unit
           2F A1    94    0A53          CLRB    MSCP$B_RCT_CPYS(R1)     ; Assume no RCT
     50    30 A4    D0    0A56          MOVL    UQB$L_UCB(R4),R0        ; Get device UCB
              13    13    0A5A          BEQL    80$                     ; None
           44 A0    9B    0A5C          MOVZBW  UCB$B_SECTORS(R0),-     ; Transfer number of sectors
           24 A1          0A5F                   MSCP$W_TRACK(R1)       ;  per track.
           45 A0    9B    0A61          MOVZBW  UCB$B_TRACKS(R0),-      ; Number of tracks per group.
           26 A1          0A64                   MSCP$W_GROUP(R1)
  04 38 A0    08    E1    0A66          BBC     #DEV$V_RCT, -           ; Branch if no RCT
                          0A6B                   UCB$L_DEVCHAR(R0),80$
     2F A1    01    90    0A6B          MOVB    #1,MSCP$B_RCT_CPYS(R1)  ; Mark RCT present
     28 A1    01    D0    0A6F  80$:    MOVL    #1,MSCP$W_CYLINDER(R1)  ; Number of groups per cylinder.
           2C A1    B4    0A73          CLRW    MSCP$W_RCT_SIZE(R1)     ; No disk has these
           2E A1    94    0A76          CLRB    MSCP$B_RBNS(R1)         ;
     50    0E A4    3C    0A79          MOVZWL  UQB$W_STATUS(R4),R0     ; Pick up unit status
              0C    12    0A7D          BNEQ    FIN_UNIT_STS            ; Either OFFLINE or AVAIL
     52    2A A4    9A    0A7F          MOVZBL  UQB$B_SLOT(R4),R2       ; Pick up slot number
     10 A3    52    E0    0A83          BBS     R2,HQB$L_UNIT_ONLN(R3),-
              03          0A87                   FIN_UNIT_STS           ; If not already ONLINE to this host
        50    04    3C    0A88          MOVZWL  #MSCP$K_ST_AVLBL,R0     ;  then it is AVAIL to it
                          0A8B
                          0A8B  FIN_UNIT_STS:
  0C A1    10 A4    D0    0A8B          MOVL    UQB$W_MULT_UNT(R4), -
```

```
                        0A90  1249             MSCP$W_MULT_UNT(R1)      ; Copy mult_unit & unit_flags
14 A1    18 A4    7D    0A90  1250     MOVQ    UQB$Q_UNIT_ID(R4), -
                        0A95  1251             MSCP$Q_UNIT_ID(R1)       ; Copy unit identifier
         20 A4    7D    0A95  1252     MOVQ    UQB$L_MEDIA_ID(R4),-
         1C A1          0A98  1253             MSCP$L_MEDIA_ID(R1)      ; Copy media identifier & shdw unit/flgs
            38    11    0A9A  1254     BRB     SEND_PKT
```

```
                           0A9C        .SBTTL  -       4 - SET CONTROLLER CHARACTERISTICS
                           0A9C ;*********************************************************
                           0A9C ;                                                        *
                           0A9C ;          # 4  -  S E T   C T R L   C H R               *
                           0A9C ;                                                        *
                           0A9C ;*********************************************************
                           0A9C ;
                           0A9C SEND_MISMATCH:
                           0A9C        BUG_CHECK       MSCPSERV,FATAL  ; Responding using stale CDT address
                           0AA0
                           0AA0 SET_CTRL_CHR:
  F6F2 CF    0C A1     B1  0AA0        CMPW    MSCP$W_VERSION(R1), -
                           0AA6         VERSION                        ; Does the version match?
             07        13  0AA6        BEQL    10$                     ; Yes, okay
    50    0C   18      9C  0AA8        ROTL    #24,#MSCP$W_VERSION,R0  ; No, set sub status
           FE88        31  0AAC        BRW     ERR_X                   ; Send invalid command packet
                           0AAF
    18 A3    0E A1     90  0AAF 10$:   MOVB    MSCP$W_CNT_FLGS(R1), -
                           0AB4         HQB$W_CNT_FLGS(R3)             ; Transfer Low byte
          10 A1        B0  0AB4        MOVW    MSCP$W_HST_TMO(R1),-
          1A A3            0AB7         HQB$W_HST_TMO(R3)
    20 A3    14 A1     7D  0AB9        MOVQ    MSCP$Q_TIME(R1), -
                           0ABE         HQB$Q_TIME(R3)                 ; ... and time
                           0ABE ;
                           0ABE ; Return controller information to host
                           0ABE ;
     00000000'9F       90  0ABE        MOVB    @#CLU$GL_ALLOCLS,-      ; Transfer the allocation class
          04 A1            0AC4         MSCP$B_CNT_ALCS(R1)
  0C A1    F6CE CF     7D  0AC6        MOVQ    CTRL_INFO, -
                           0ACC         MSCP$W_VERSION(R1)             ; Send back info
  14 A1    F6C0 CF     7D  0ACC        MOVQ    CTRL_ID, -
                           0AD2         MSCP$Q_CNT_ID(R1)              ; ... and ID
                           0AD2
                           0AD2 SEND_SUCCESS:
             50        D4  0AD2        CLRL    R0                      ; Set success
                           0AD4
                           0AD4 SEND_PKT:
    52    B0 A5        D0  0AD4        MOVL    CDRP$L_MSGBUFX(R5),R2   ; Pick up message buffer ptr
             47        13  0AD8        BEQL    CLEAN_UP                ; No message buffer
    0A A2    50        B0  0ADA        MOVW    R0,MSCP$W_STATUS(R2)    ; Set the status
                           0ADE
                           0ADE SEND_PKT_STS:
    51    08 A2        9A  0ADE        MOVZBL  MSCP$B_OPCODE(R2),R1    ; Pick up opcode
  08 A2    80 8F       88  0AE2        BISB2   #MSCP$K_OP_END, -
                           0AE7         MSCP$B_OPCODE(R2)              ; Make an end packet
  51    1D24'CF41      9A  0AE7        MOVZBL  END_PKT_LEN[R1],R1      ; Pick up length
                           0AED
                           0AED SEND_END:
    1C A5    52        D0  0AED        MOVL    R2,CDRP$L_MSG_BUF(R5)   ; Stuff the message buffer address
    54    60 A5        D0  0AF1        MOVL    CDRP$L_CDTX(R5),R4      ; Pick up the CDT address
    24 A5    54        D0  0AF5        MOVL    R4,CDRP$L_CDT(R5)       ; Store into standard location
  F8 A2    18 A4       D1  0AF9        CMPL    CDT$L_LCONID(R4), -     ; Check validity of remote conid
                           0AFE         SCS$L_DST_CONID(R2)
             9C        12  0AFE        BNEQ    SEND_MISMATCH           ; Trying to respond to stale connection
    D6 A5    05        B0  0B00        MOVW    #5,CDRP$W_STATE(R5)     ; Message send wait state
    53    5C A4        D0  0B04        MOVL    CDT$L_AUXSTRUC(R4),R3   ; Pick up HQB address
    54    10 A4        D0  0B08        MOVL    CDT$L_PDT(R4),R4        ; Set up PDT address
```

```
   AC A5    51    D0  0B0C          MOVL    R1,CDRP$L_PID(R5)        ; Save the length
                      0B10          RECYCL_MSG_BUF                   ; Recycle the message
   51    AC A5    D0  0B13          MOVL    CDRP$L_PID(R5),R1        ; Restore the length
                      0B17          SEND_CNT_MSG_BUF                 ; Send the message
   53    7C A5    D0  0B1A          MOVL    CDRP$L_HQB(R5),R3        ; Get HQB address
         28 A3    B7  0B1E          DECW    HQB$W_NUM_QUE(R3)        ; Drop count on host
                      0B21
                      0B21          .ENABLE LSB
                      0B21  CLEAN_UP:
   08    FB A5    91  0B21          CMPB    CDRP$B_OPCODE(R5), -     ; For immediate commands
                      0B25                  #MSCP$K_OP_AVAIL
            07    1F  0B25          BLSSU   5$
            55    DD  0B27          PUSHL   R5                       ; Save CDRP address
            0E    10  0B29          BSBB    20$                      ; Finish cleaning up
            55  8ED0  0B2B          POPL    R5                       ; Restore CDRP address
   53    7C A5    D0  0B2E  5$:     MOVL    CDRP$L_HQB(R5),R3        ; Get HQB address
         2C A3    B7  0B32          DECW    HQB$W_REFC(R3)           ; Decrement HQB reference count
          1160    30  0B35          BSBW    FREE_STRUCT              ; Free data structure
                  05  0B38  10$:    RSB                              ; Just leave
                      0B39
   54    78 A5    D0  0B39  20$:    MOVL    CDRP$L_MST_UQB(R5),R4    ; Pick up UQB pointer
            F9    13  0B3D          BEQL    10$                      ; There is none
         E8 A5    D5  0B3F          TSTL    CDRP$L_UQB_FL(R5)        ; Is this CDRP queued on a UQB?
            0A    13  0B42          BEQL    30$                      ; No
   50    E8 A5    0F  0B44          REMQUE  CDRP$L_UQB_FL(R5),R0     ; Remove from unit que
         E8 A5    D4  0B48          CLRL    CDRP$L_UQB_FL(R5)        ; Prevent double remque - if this code
                      0B4B                                           ;  is moved, always get rid of link!!!
         50 A4    B7  0B4B          DECW    UQB$W_NUM_QUE(R4)        ; Drop count on unit
                      0B4E  ;
                      0B4E  ; Check for any buffer resource and return it.  If possible start up any
                      0B4E  ; thread waiting for buffer.
                      0B4E  ;
   FA A5    02    93  0B4E  30$:    BITB    #1@CDRP$V_BUFFER, -      ; Check for buffer
                      0B52                  CDRP$B_FLAGS(R5)
            1C    13  0B52          BEQL    40$                      ; None, skip next
          0777    30  0B54          BSBW    DEALLOC_BUFF             ; Yes, get rid of it
       F505 CF    B5  0B57          TSTW    NUM_QUE                  ; Anyone waiting?
            13    13  0B5B          BEQL    40$                      ; No, leave now
   55  F517 DF    0F  0B5D          REMQUE  @MEM_WAIT,R5             ; Grab first entry
            0C    1D  0B62          BVS     40$                      ; None
            54    DD  0B64          PUSHL   R4                       ; Save context
   53    10 A5    7D  0B66          MOVQ    CDRP$L_FR3(R5),R3        ; Restore old context
         0C B5    16  0B6A          JSB     @CDRP$L_FPC(R5)          ; Finish off the pkt
            54  8ED0  0B6D          POPL    R4                       ; Restore old, old context
                      0B70  ;
                      0B70  ; If this is last command on execute queue, then check for any blocked
                      0B70  ; commands.  If there are some, start up the first one.  If the SEQ bit
                      0B70  ; is still set, we are in single command mode, so do not fetch any more
                      0B70  ; commands, otherwise drain the blocked queue.
                      0B70  ;
         50 A4    B5  0B70  40$:    TSTW    UQB$W_NUM_QUE(R4)        ; Any entries queued?
            C3    12  0B73          BNEQ    10$                      ; Yes, leave
                      0B75
                      0B75          .DISABLE        LSB
                      0B75  DRAIN_BLOCKQ:
         7C A4    B5  0B75  10$:    TSTW    UQB$W_NUM_BLK(R4)        ; Any entries queued?
            25    13  0B78          BEQL    30$                      ; No, leave
```

```
55   68 B4    0F  0B7A        REMQUE  @UQB$Q_BLOCKQ(R4),R5       ; Remove from blocked que
        1F    1D  0B7E        BVS     30$                        ; None, leave
     7C A4    B7  0B80        DECW    UQB$W_NUM_BLK(R4)          ; Drop count
        04    12  0B83        BNEQ    20$                        ; Is it the last?
        02    8A  0B85        BICB    #UQB$M_BLOCKED,-           ; Yes, clear the flag
     2B A4        0B87                 UQB$B_FLAGS(R4)
55   18 A5    DE  0B89   20$: MOVAL   CDRP$L_FQFL- -
                  0B8D                 CDRP$L_UQB_FL(R5),R5      ; Set new CDRP ptr
        54    DD  0B8D        PUSHL   R4                         ; Save old UQB
53   10 A5    7D  0B8F        MOVQ    CDRP$L_FR3(R5),R3          ; Restore old context
     0C B5    16  0B93        JSB     @CDRP$L_FPC(R5)            ; Finish off the pkt
        54  8ED0  0B96        POPL    R4                         ; Restore old UQB
        30    93  0B99        BITB    #<UQB$M_SEQ!UQB$M_RIP>,-   ; Check if in SEQ mode
     2B A4        0B9B                 UQB$B_FLAGS(R4)
        D6    13  0B9D        BEQL    10$                        ; No, get next command
              05  0B9F   30$: RSB
```

```
                    0BA0        .SBTTL  SEQUENTIAL class commands
                    0BA0  ;
                    0BA0  ; This section of code sets up for sequential command processing.  First
                    0BA0  ; we make checks for sequentiality.
                    0BA0  ;         a) If commands are already blocked, then this command must be blocked
                    0BA0  ;                 also to preserve sequentiality.  Code at label CLEAN_UP will
                    0BA0  ;                 unravel all of this
                    0BA0  ;         b) If there are any non-sequential commands on the processing queue,
                    0BA0  ;                 then this command must be blocked and the blocked flag set
                    0BA0  ;                 in the UQB.
                    0BA0  ;         c) EXCEPT that an ONLINE or AVAILABLE to a shadow master in the
                    0BA0  ;                 copy-in-progress phase will be let thru, since that phase
                    0BA0  ;                 is not strictly sequential.
                    0BA0  ;
                    0BA0  ;         R0    scratch
                    0BA0  ;         R1 = Ptr to message buffer
                    0BA0  ;         R2 = low 3 bits of command
                    0BA0  ;         R3 = Ptr to HQB
                    0BA0  ;         R4 = Unit slot number
                    0BA0  ;         R5 = Ptr to CDRP
                    0BA0  ;
      FB A5    91   0BA0  SEQ:    CMPB    CDRP$B_OPCODE(R5),-       ; Check opcode now
         0B         0BA3                    #MSCP$K_OP_DTACP
         60    1A   0BA4          BGTRU   70$                       ; Out of range - error
         03    1F   0BA6          BLSSU   10$                       ; In range, continue
       0794    31   0BA8          BRW     DET_ACC_PATH              ; DAP is a nop
                    0BAB
54    78 A5    D0   0BAB  10$:    MOVL    CDRP$L_MST_UQB(R5),R4     ; Get UQB pointer
               93   0BAF          BITB    #<UQB$M_BLOCKED!-         ; If already blocked,
                    0BB0                      UQB$M_RIP!-           ;  or in a repair
                    0BB0                      UQB$M_SEQ>,-          ;  or in sequential mode
   2B A4    32      0BB0                      UQB$B_FLAGS(R4)       ; block this cmd
         0F    12   0BB3          BNEQ    30$
         09    E1   0BB5          BBC     #MSCP$V_UF_SSMST,-        ; If not shadow master, skip test
   05 12 A4         0BB7                    UQB$W_UNT_FLGS(R4),20$
         00    E0   0BBA          BBS     #UQB$V_CIP,-              ; If CIP, then allow thru
   1A 2B A4         0BBC                    UQB$B_FLAGS(R4),50$
      50 A4    B5   0BBF  20$:    TSTW    UQB$W_NUM_QUE(R4)         ; Any entries queued?
         15    13   0BC2          BEQL    50$                       ; No, leave now
         10    88   0BC4  30$:    BISB    #UQB$M_SEQ,-              ; Yes, set for sequential mode
      2B A4         0BC6                    UQB$B_FLAGS(R4)         ;     (rundown the IO)
                    0BC8          BLOCKT                            ;  and block the command until done
                    0BCB  ;
                    0BCB  ; Just in case we get restarted and there are entries left in progress, we
                    0BCB  ; have to wait, but we also have to block any new entries.
                    0BCB  ;
      50 A4    B5   0BCB  40$:    TSTW    UQB$W_NUM_QUE(R4)         ; Any entries queued?
         09    13   0BCE          BEQL    50$                       ; No, leave now
         10    88   0BD0          BISB    #UQB$M_SEQ,-              ; Yes, set sequential mode while waiting
      2B A4         0BD2                    UQB$B_FLAGS(R4)
                    0BD4          BLOCKH                            ; Block the command
         F2    11   0BD7          BRB     40$                       ; Command is resumed, check again
                    0BD9
      E8 A5    D4   0BD9  50$:    CLRL    CDRP$L_UQB_FL(R5)         ; So no spurious remques
         10    8A   0BDC          BICB    #UQB$M_SEQ,-              ; Clear the flag in event of an
      2B A4         0BDE                    UQB$B_FLAGS(R4)         ;     error exit
50       03    D0   0BE0          MOVL    #MSCP$K_ST_OFFLN,R0       ; Assume offline status
```

```
         50    0E A4   3C  0BE3          MOVZWL  UQB$W_STATUS(R4),R0         ; Pick up status from unit
                  0C   12  0BE7          BNEQ    60$                         ; Either OFFLINE or AVAIL
         51    2A A4   9A  0BE9          MOVZBL  UQB$B_SLOT(R4),R1           ; Recover slot number
   03 10 A3       51   E0  0BED          BBS     R1,HQB$L_UNIT_ONLN(R3),60$ ; If not alread ONLINE this host
            50    04   D0  0BF2          MOVL    #MSCP$K_ST_AVLBL, R0        ;  then it is AVAIL
         51    B0 A5   D0  0BF5    60$:  MOVL    CDRP$L_MSGBUFX(R5),R1       ; Pick up message buffer
                           0BF9          CASE    CDRP$B_OPCODE(R5),-         ; Dispatch on command subtype
                           0BF9                   TYPE=B,LIMIT=#MSCP$K_OP_AVAIL-
                           0BF9                  <AVAIL,-                    ;  8 - AVAILABLE
                           0BF9                   ONLINE,-                   ;  9 - ONLINE
                           0BF9                   SET_UNIT_CHR,-             ; 10 - SET UNIT CHARACTERISTICS
                           0BF9                   DET_ACC_PATH>              ; 11 - DETERMINCE ACCESS PATH
             FD20      31  0C06    70$:  BRW     ERR_T
```

```
                 0C09         .SBTTL  -       8 - AVAILABLE
                 0C09   ;*****************************************************
                 0C09   ;                                                    *
                 0C09   ;           # 8  -  A V A I L A B L E                *
                 0C09   ;                                                    *
                 0C09   ;*****************************************************
                 0C09   ;
      50   B5    0C09   AVAIL:  TSTW    R0                      ; Check unit status
      1F   12    0C0B           BNEQ    30$                     ; Either offline or available
51    F8 A5   3C 0C0D   10$:    MOVZWL  CDRP$W_SLOT(R5),R1      ; Pick up unit number
      10   88    0C11           BISB    #UQB$M_SEQ,-            ; Set for sequential mode
   2B A4         0C13                    UQB$B_FLAGS(R4)        ;    no new IO until done
   E8 A5   0E    0C15           INSQUE  CDRP$L_UQB_FL(R5),-     ; Insert this CDRP on unit queue
   64 B4         0C18                    @UQB$L_CDRP_BL(R4)
   50 A4   B6    0C1A           INCW    UQB$W_NUM_QUE(R4)       ; Account for it
      1C   10    0C1D           BSBB    MAKE_AVAIL              ; Make available
54    78 A5   D0 0C1F           MOVL    CDRP$L_MST_UQB(R5),R4   ; Recover UQB pointer
      10   8A    0C23           BICB    #UQB$M_SEQ,-            ; Clear the sequential flag -
   2B A4         0C25                    UQB$B_FLAGS(R4)        ;   allow any IO now
      50   D4    0C27           CLRL    R0                      ; Send success
    FEA8   31    0C29   20$:    BRW     SEND_PKT                ; Do it
                 0C2C   ;
                 0C2C   ; If unit was OFFLINE, command is illegal in this state, send OFFLINE status
                 0C2C   ;
                 0C2C           ASSUME  <MSCP$K_ST_OFFLN&1> NE 0
   FA 50   E8    0C2C   30$:    BLBS    R0,20$                  ; OFFLINE
      54   D5    0C2F           TSTL    R4                      ; Any UQB?
      05   13    0C31           BEQL    40$                     ; No, skip next
      09   E0    0C33           BBS     #MSCP$V_UF_SSMST,-      ; If this is a shadow master,
D5 12 A4         0C35                    UQB$W_UNT_FLGS(R4),10$ ;  then allow the command else ...
    FE97   31    0C38   40$:    BRW     SEND_SUCCESS            ; AVAILABLE, is NOP
                 0C3B   ;
                 0C3B   ; Subroutine to make unit AVAILABLE.  It can be called recursively.
                 0C3B   ;
                 0C3B   ;       R1  =   Unit number
                 0C3B   ;       R3 -->  HQB
                 0C3B   ;       R4 -->  UQB
                 0C3B   ;       SP -->  Caller of MAKE_AVAIL
                 0C3B   ;       4(SP)>  Caller's caller
                 0C3B   ;
                 0C3B   MAKE_AVAIL:
00 10 A3   51  E5 0C3B          BBCC    R1,HQB$L_UNIT_ONLN(R3),10$ ; Show this unit not online this host
51    1E A3   9A 0C40   10$:    MOVZBL  HQB$B_VEC(R3),R1        ; Pick up host slot number
      54   D5    0C44           TSTL    R4                      ; Any UQB?
      2A   13    0C46           BEQL    30$                     ; No, skip next
20 34 A4   51  E5 0C48          BBCC    R1,UQB$L_HOST_ONLN(R4),20$ ; Show this host not online this unit
   0C A4   97    0C4D           DECB    UQB$B_HOST_CNT(R4)      ; Drop host count on unit
      20   18    0C50           BGEQ    30$                     ; If any host left, leave now
      09   E0    0C52           BBS     #MSCP$V_UF_SSMST,-      ; If this is a shadow master,
   12 A4         0C54                    UQB$W_UNT_FLGS(R4),-   ;  then allow the command else ...
      66         0C56                    AV_CASE3
      04   B0    0C57           MOVW    #MSCP$K_ST_AVLBL,-      ;  else set status for unit
   0E A4         0C59                    UQB$W_STATUS(R4)       ; ONLINE --> AVAILABLE
52    30 A4   D0 0C5B           MOVL    UQB$L_UCB(R4),R2        ; Pick up UCB pointer
 00AE C2   97    0C5F           DECB    UCB$B_ONLCNT(R2)        ; Decrement count
      0E   15    0C63           BLEQ    40$                     ; If zero, do IO
                 0C65   ;
```

```
                       0C65   ; This is the last host online to this unit, but there is a local path
                       0C65   ; to the unit.  Thus there is no actual IO needed to the unit.  However,
                       0C65   ; check if any post processing is called for (shadow member only).
                       0C65   ;
            0E    E1   0C65          BBC     #MSCP$V_UF_SSMEM,-          ; If this is a shadow member,
      08 12 A4         0C67                  UQB$W_ONT_FLGS(R4),30$      ;  then
          0098    31   0C6A          BRW     AV_CASE2X                   ; Continue in special routine
                       0C6D   ;
                       0C6D   ; Host was never online to this unit.  However make special check for
                       0C6D   ; a shadow master with no units and in a transistion state.
                       0C6D   ;
            09    E0   0C6D   20$:   BBS     #MSCP$V_UF_SSMST,-          ; If this is a shadow master,
         12 A4         0C6F                  UQB$W_ONT_FLGS(R4),-        ;  then allow the command else ...
            4B         0C71                  AV_CASE3
            05         0C72   30$:   RSB                                 ; Leave without further processing
                       0C73   ;
                       0C73   ; This is the last host online to this unit, and there are no local paths
                       0C73   ; to it either.  Thus decide if SPINDOWN is needed, and issue the AVAIL.
                       0C73   ;
            55    D5   0C73   40$:   TSTL    R5                          ; Is there a CDRP?
            FB    13   0C75          BEQL    30$                         ; No, skip tests
      51    B0 A5 D0   0C77          MOVL    CDRP$L_MSGBUFX(R5),R1       ; Recover MSCP packet pointer
            0D    13   0C7B          BEQL    50$                         ; None
            01    E1   0C7D          BBC     #MSCP$V_MD_SPNDW,-          ; If SPINDOWN modifier...
      08 0A A1         0C7F                  MSCP$W_MODIFIER(R1),50$
                  B0   0C82          MOVW    #IO$_UNLOAD!-               ; Set function to UNLOAD
                       0C83                  IO$M_INHERLOG,-
 C0 A5    0801 8F      0C83                  CDRP$W_FUNC(R5)
            06    11   0C88          BRB     60$
                       0C8A
                  B0   0C8A   50$:   MOVW    #IO$_AVAILABLE!-            ; Set function to AVAILABLE
                       0C8B                  IO$M_INHERLOG,-
 C0 A5    0811 8F      0C8B                  CDRP$W_FUNC(R5)
            0E    E1   0C90   60$:   BBC     #MSCP$V_UF_SSMEM,-          ; If this is a shadow member,
      08 12 A4         0C92                  UQB$W_ONT_FLGS(R4),70$      ;  then
      CC A5 8ED0       0C95          POPL    CDRP$L_SVAPTE(R5)           ; Save normal return address
     0CFE'CF      9F   0C99          PUSHAB  AV_CASE2                    ;  and substitute special routine
                       0C9D   70$::  BSBB    SETX                        ; Do the IO
                       0C9D   ;
                       0C9D   ; Subroutine to setup for disk IO
                       0C9D   ;
  D6 A5     02    B0   0C9D   SETX:  MOVW    #2,CDRP$W_STATE(R5)         ; Show it queued to driver
  54 A5     53    D0   0CA1          MOVL    R3,CDRP$L_FR3X(R5)          ; Save the HQB pointer
  C4 A5     54    D0   0CA5          MOVL    R4,CDRP$L_UQB(R5)           ; Save the UQB pointer
         30 A4    D0   0CA9          MOVL    UQB$L_UCB(R4),-             ; Transfer UCB address
         BC A5         0CAC                  CDRP$L_UCB(R5)              ;   to UCB addr in IRP
         48 A5    D4   0CAE          CLRL    CDRP$L_LBN(R5)              ; Just in case
  53     A0 A5    DE   0CB1          MOVAL   CDRP$L_IOQFL(R5),R3         ; Cover IRP w/R3
            08    E2   0CB5          BBSS    #IRP$V_PHYSIO,-             ; Set PHYSICAL IO bypass
      00 CA A5         0CB7                  CDRP$W_STS(R5),10$
          0DB7    31   0CBA   10$:   BRW     DO_DISKX                    ; Send it off
                       0CBD   ;
                       0CBD   ; AVAILABLE to a shadow set master: Send AVAILABLE to each of the physical
                       0CBD   ; units in the shadow set.  If the shadow set is disolved, then get rid of
                       0CBD   ; the shadow master unit.
                       0CBD   ;
                       0CBD   AV_CASE3:
```

```
       26 A4    33  0CBD         CVTWB   UQB$W_SHDW_STS(R4),-
       C2 A5        0CC0                  CDRP$B_SHDW_CNT(R5)     ; If no members ...
          25    13  0CC2         BEQL    30$                     ;  then just deallocate shadow UQB
       5C A5  8ED0  0CC4         POPL    CDRP$L_SAVD_RTX(R5)     ; Save return address
          01    8A  0CC8         BICB    #1@CDRP$V_SLAVE,-       ; Clear out unneeded flag
       FA A5        0CCA                  CDRP$B_FLAGS(R5)
    54    64    D0  0CCC 10$:    MOVL    UQB$L_SHDW_FL(R4),R4    ; Pick up next shadow member
          0C    13  0CCF         BEQL    20$                     ; None left
 51    2A A4    9A  0CD1         MOVZBL  UQB$B_SLOT(R4),R1       ; Pick up unit slot
        FF63    30  0CD5         BSBW    MAKE_AVAIL              ; Do the available
       C2 A5    97  0CD8         DECB    CDRP$B_SHDW_CNT(R5)     ; Drop the count
          EF    1A  0CDB         BGTRU   10$                     ; Loop if not done
       5C A5    DD  0CDD 20$:    PUSHL   CDRP$L_SAVD_RTX(R5)     ; Return address
 54    78 A5    D0  0CE0         MOVL    CDRP$L_MST_UQB(R5),R4   ; Pick up shadow master UQB
       26 A4    B5  0CE4         TSTW    UQB$W_SHDW_STS(R4)      ; Is this the last physical unit?
          14    12  0CE7         BNEQ    40$                     ; No, just return
 51    2A A4    9A  0CE9 30$:    MOVZBL  UQB$B_SLOT(R4),R1       ; Pick up unit slot
  F39E CF41     D4  0CED         CLRL    UNIT_VEC[R1]            ; Clear shadow unit out of vector
 50    0080 C4  0F  0CF2         REMQUE  UQB$Q_UNITQ(R4),R0      ; Unlink from chain
    50    54    D0  0CF7         MOVL    R4,R0                   ; Set R0 to point to head
        F951    31  0CFA         BRW     DEALMEM                 ; Get rid of shadow master UQB
                    0CFD                                         ;  control returns to MAKE_AVAIL caller
                05  0CFD 40$:    RSB                             ; Return to caller's caller
                    0CFE ;
                    0CFE ; AVAILABLE to a shadow set member: Clear out the shadowing fields, remove
                    0CFE ; this unit from the shadow set ring, and if last unit, set shadow set
                    0CFE ; master unit to available.
                    0CFE ;
                    0CFE AV_CASE2:
 53    54 A5    7D  0CFE         MOVQ    CDRP$L_FR3X(R5),R3      ; Recover HQB & UQB
       CC A5    DD  0D02         PUSHL   CDRP$L_SVAPTE(R5)       ; Return address
                    0D05
                    0D05 AV_CASE2X:
          0E    E5  0D05         BBCC    #MSCP$V_UF_SSMEM,-      ; Clear the shadow set member flag
    00 12 A4        0D07                  UQB$W_UNT_FLGS(R4),10$
       24 A4    B4  0D0A 10$:    CLRW    UQB$W_SHDW_UNT(R4)      ; Clear out old shadow master
 50    78 A4    D0  0D0D         MOVL    UQB$L_SHDW_MST(R4),R0   ; Point to shadow master
       78 A4    D4  0D11         CLRL    UQB$L_SHDW_MST(R4)      ; Remove the pointer
       26 A0    B7  0D14         DECW    UQB$W_SHDW_STS(R0)      ; Decre shadow member count
          0B    12  0D17         BNEQ    20$                     ; At least one left
          60    D4  0D19         CLRL    UQB$L_SHDW_FL(R0)       ; Clear master pointer
       78 A0    D4  0D1B         CLRL    UQB$L_SHDW_LOW(R0)      ;  and low pointer
          04    B0  0D1E         MOVW    #MSCP$K_ST_AVLBL,-      ; Set status to AVAIL
       0E A0        0D20                  UQB$W_STATUS(R0)
          22    11  0D22         BRB     40$                     ; Skip all else
                    0D24
    52    64    D0  0D24 20$:    MOVL    UQB$L_SHDW_FL(R4),R2    ; Get our successor
    51    64    0F  0D27         REMQUE  UQB$L_SHDW_FL(R4),R1    ; Remove us from the shadow ring
 54    78 A0    D1  0D2A         CMPL    UQB$L_SHDW_LOW(R0),R4   ; Were we the lowest unit?
          0E    12  0D2E         BNEQ    30$                     ; No, leave
       28 A2    B0  0D30         MOVW    UQB$W_UNIT(R2),-        ; Transfer its unit
       24 A0        0D33                  UQB$W_SHDW_UNT(R0)      ;  to shadow master's unit field
       30 A2    D0  0D35         MOVL    UQB$L_UCB(R2),-         ; Transfer its UCB pointer
       30 A0        0D38                  UQB$L_UCB(R0)           ;  to shadow master
 78 A0    52    D0  0D3A         MOVL    R2,UQB$L_SHDW_LOW(R0)   ; Update the new lowest pointer
    54    60    D1  0D3E 30$:    CMPL    UQB$L_SHDW_FL(R0),R4    ; Were we the first in the ring?
          03    12  0D41         BNEQ    40$                     ; No, skip on
```

```
         60   52   D0  0D43       MOVL    R2,UQB$L_SHDW_FL(R0)     ; Yes, update master with new link
                   05  0D46  40$: RSB                              ; Return to MAKE_AVAIL caller
                       0D47  ;
                       0D47  ; AVAIL_ALL: Make the unit AVAILABLE to all hosts to which it is currently
                       0D47  ;        ONLINE.  Also make it physically AVAILABLE.  Cannot be called for
                       0D47  ;        shadow masters.
                       0D47  ;
                       0D47  ;        R4 -->  UQB of unit to be made AVAIL
                       0D47  ;
                       0D47  AVAIL_ALL:
            5C A5 8ED0 0D47       POPL    CDRP$L_SAVD_RTX(R5)      ; Save return address
      C4 A5   54   D0  0D4B       MOVL    R4,CDRP$L_UQB(R5)        ; Save UQB pointer
            2A A4  9B  0D4F       MOVZBW  UQB$B_SLOT(R4),-         ; Pick up the slot index
            F8 A5      0D52               CDRP$W_SLOT(R5)          ;  and save in CDRP
               01  8A  0D54       BICB    #1@CDRP$V_SLAVE,-        ; Clear unneeded flag
            FA A5      0D56               CDRP$B_FLAGS(R5)
               1F  B0  0D58       MOVW    #<2*MSU$K_MAX_HOST>-1,-  ; Stuff max index into convenient
            D0 A5      0D5A               CDRP$W_BOFF(R5)          ;  place
         53 D0 A5  3C  0D5C  10$: MOVZWL  CDRP$W_BOFF(R5),R3       ; Pick up index
   53   F3AB CF43  D0  0D60       MOVL    HOST_VEC[R3],R3          ; Pick up possible HQB address
               07  13  0D66       BEQL    20$                      ; Non, loop for more
         51 F8 A5  3C  0D68       MOVZWL  CDRP$W_SLOT(R5),R1       ; Pick up the slot index
             FECC  30  0D6C       BSBW    MAKE_AVAIL               ; AVAILABLE the unit
            D0 A5  B7  0D6F  20$: DECW    CDRP$W_BOFF(R5)          ; Count down the index
               E8  18  0D72       BGEQ    10$                      ; Loop for more hosts
            5C B5  17  0D74       JMP     @CDRP$L_SAVD_RTX(R5)     ; Leave
```

```
                          0D77        .SBTTL  -       9 - ONLINE
                          0D77  ;*****************************************************
                          0D77  ;                                                    *
                          0D77  ;        # 9 - O N L I N E                           *
                          0D77  ;                                                    *
                          0D77  ;*****************************************************
                          0D77  ;
                          0D77  ONLINE:
          40 50      E8   0D77          BLBS    R0,60$                    ; If OFFLINE, error out
        E3B8 8F      AA   0D7A          BICW    #^C<HOSTSET>,-            ; Clear non-host flags in the packet
          0E A1           0D7E                  MSCP$W_UNT_FLGS(R1)
             04      E0   0D80          BBS     #MSCP$V_MD_SHDSP,-
       13 0A A1           0D82                  MSCP$W_MODIFIER(R1),10$; Is this a shadowing operation?
                          0D85  ;
                          0D85  ; CASE 1: ONLINE to a single physical unit, no shadowing at all
                          0D85  ;
                          0D85
                          0D85  ;;***                                                          ;***
                          0D85  ;; TEMP UNTIL VMS GETS SHADOWING IN MOUNT                       ;***
                          0D85  ;;                                                             ;***
   13 12 A4     09   E0   0D85          BBS     #MSCP$V_UF_SSMST,UQB$W_UNT_FLGS(R4),20$ ;***
                          0D8A  ;;***                                                          ;***
                          0D8A
                     B3   0D8A          BITW    #<MSCP$M_UF_SSMST!-       ; No, check consistency
                          0D8B                   MSCP$M_UF_SSMEM>,-
   12 A4      4200 8F     0D8B                  UQB$W_UNT_FLGS(R4)        ; Is the unit involved in shadowing?
                63   13   0D90          BEQL    CHECK_ONLINE              ; No, go directly to ONLINE test
   50    0A     18   9C   0D92          ROTL    #24,#MSCP$W_MODIFIER,R0   ; Yes, set illegal modifier
                15   11   0D96          BRB     40$                       ; ... send invalid end message
                          0D98
                09   E1   0D98  10$:    BBC     #MSCP$V_UF_SSMST,-        ; Is this unit a shadow master?
       20 12 A4           0D9A                  UQB$W_UNT_FLGS(R4),70$
                          0D9D  ;
                          0D9D  ; CASE 3: ONLINE to a shadow master unit
                          0D9D  ;
                          0D9D
                          0D9D  ;;***                                                          ;***
                          0D9D  ;; TEMP                                                        ;***
                          0D9D  ;;                                                             ;***
   20 A1      04 A1  B0   0D9D  20$:    MOVW    MSCP$W_UNIT(R1),MSCP$W_SHDW_UNT(R1)           ;***
                          0DA2  ;;***                                                          ;***
                          0DA2
          20 A1      B1   0DA2          CMPW    MSCP$W_SHDW_UNT(R1),-     ; Yes, now does the virtual
          28 A4           0DA5                  UQB$W_UNIT(R4)            ;  unit match the physical?
             07      13   0DA7          BEQL    50$                       ; Yes
   50    20     18   9C   0DA9  30$:    ROTL    #24,#MSCP$W_SHDW_UNT,R0   ; No, set illegal shadow unit
            FB87     31   0DAD  40$:    BRW     ERR_X
                          0DB0
          26 A4      B0   0DB0  50$:    MOVW    UQB$W_SHDW_STS(R4),-      ; Pick up the number of members
          22 A1           0DB3                  MSCP$W_SHDW_STS(R1)
             3E      12   0DB5          BNEQ    CHECK_ONLINE              ; Non-zero, continue processing
       50    24      B0   0DB7          MOVW    #<1*MSCP$K_ST_SBCOD>!-
                          0DBA                  MSCP$K_ST_AVLBL,R0        ; AVAILABLE, NO MEMBERS
            FD17     31   0DBA  60$:    BRW     SEND_PKT
                          0DBD  ;
                          0DBD  ; CASE 2: ONLINE to a physical unit to bring it into a shadow set
                          0DBD  ;
```

```
     50    20 A1   B0  0DBD  1713 70$:    MOVW    MSCP$W_SHDW_UNT(R1),R0  ; Pick up shadow unit
              E6   18  0DC1  1714         BGEQ    30$                     ; Error if sign bit not set
        28 A4 50   B1  0DC3  1715         CMPW    R0,UQB$W_UNIT(R4)       ; Does virtual unit match the physical?
              E0   13  0DC7  1716         BEQL    30$                     ; Yes, error (only true for master)
              0E   E1  0DC9  1717         BBC     #MSCP$V_UF_SSMEM,-      ; No, now is this an existing shadow
        19 12 A4       0DCB  1718                 UQB$W_UNT_FLGS(R4),100$; set member
        24 A4 50   B1  0DCE  1719         CMPW    R0,UQB$W_SHDW_UNT(R4)   ; Yes, does its shadow master match?
              D5   12  0DD2  1720         BNEQ    30$                     ; No, error
           14 A1   C1  0DD4  1721 80$:    ADDL3   MSCP$L_EXCL_LBA(R1),-   ; Now pick up the excluded area
        52 18 A1       0DD7  1722                 MSCP$W_EXCL_LBC(R1),R2
        48 A4 52   D1  0DDA  1723         CMPL    R2,UQB$L_MAXBLOCK(R4)   ; Is it beyond the legal limits
              15   1B  0DDE  1724         BLEQU   CHECK_ONLINE            ; No, all is okay
     50    14 18   9C  0DE0  1725         ROTL    #24,#MSCP$L_EXCL_LBA,R0 ; Yes, set status
            FB50   31  0DE4  1726 90$:    BRW     ERR_X                   ;  ... send invalid end message
                       0DE7  1727 ;
                       0DE7  1728 ; Physical unit is not already a shadow set member.  Now test if it is
                       0DE7  1729 ; already ONLINE as a non-shadow unit to anyone.  If so, error out.
                       0DE7  1730 ;
           0C A4   95  0DE7  1731 100$:   TSTB    UQB$B_HOST_CNT(R4)      ; Is it already online?
              E8   19  0DEA  1732         BLSS    80$                     ; No, allow it thru
 50   FF000000 8F  D0  0DEC  1733         MOVL    #^XFF000000,R0          ; Yes, this is illegal
              EF   11  0DF3  1734         BRB     90$                     ; Tell host
                       0DF5  1735 ;
                       0DF5  1736 ; Now check status of unit.
                       0DF5  1737 ;
                       0DF5  1738 CHECK_ONLINE:
              50   B5  0DF5  1739         TSTW    R0                      ; Check status
              24   12  0DF7  1740         BNEQ    30$                     ; AVAIL
                       0DF9  1741 ;
                       0DF9  1742 ; Was ONLINE to this host already
                       0DF9  1743 ;
     50    0100 8F A8  0DF9  1744         BISW2   #<MSCP$K_SC_ALONL*-     ; Set subcode "Already online"
                       0DFE  1745                 MSCP$K_ST_SBCOD>,R0
         1314'CF   9F  0DFE  1746         PUSHAB  COPY_CHAR               ; Where to go next
              52   DD  0E02  1747 10$:    PUSHL   R2                      ; Save register
         E280 8F   AB  0E04  1748         BICW3   #NONHOSTSET,-           ; Pick up host settable flags from
        52 12 A4       0E08  1749                 UQB$W_UNT_FLGS(R4),R2   ;  the unit
        52 0E A1   B1  0E0B  1750         CMPW    MSCP$W_UNT_FLGS(R1),R2  ; Check againts host settable flags
              04   BA  0E0F  1751         POPR    #^M<R2>                 ; Restore, but leave CC set
              09   13  0E11  1752         BEQL    20$                     ; All okay, fill in info
              8E   D5  0E13  1753         TSTL    (SP)+                   ; Remove return address
     50    0E 18   9C  0E15  1754         ROTL    #24,#MSCP$W_UNT_FLGS,R0 ; Set secondary status
            FB1B   31  0E19  1755         BRW     ERR_X                   ;  ... send invalid end packed
                       0E1C  1756
                   05  0E1C  1757 20$:    RSB
                       0E1D  1758 ;
                       0E1D  1759 ; Was AVAILABLE, now turn ONLINE
                       0E1D  1760 ;
           0E A4   B5  0E1D  1761 30$:    TSTW    UQB$W_STATUS(R4)        ; Check the actual unit
              02   12  0E20  1762         BNEQ    35$                     ; No other host has it online, skip
              DE   10  0E22  1763         BSBB    10$                     ; Test unit flags
              09   E0  0E24  1764 35$:    BBS     #MSCP$V_UF_SSMST,-      ; If shadow master unit,
        6F 12 A4       0E26  1765                 UQB$W_UNT_FLGS(R4),50$ ;  skip the physical online stuff
        52 1E A3   9A  0E29  1766         MOVZBL  HQB$B_VEC(R3),R2        ; Pick up host slot number
     14 34 A4 52   E0  0E2D  1767         BBS     R2,UQB$L_HOST_ONLN(R4),40$ ; If host online bit set, MOUNTVER
           0C A4   95  0E32  1768         TSTB    UQB$B_HOST_CNT(R4)      ; Test count of online hosts
              61   18  0E35  1769         BGEQ    50$                     ; Skip UCB & packack, if already done
```

```
       52   30 A4   D0 0E37          MOVL    UQB$L_UCB(R4),R2           ; Pick up UCB address
          00AE C2   96 0E3B          INCB    UCB$B_ONLCNT(R2)           ; Increment the online count
    01    00AE C2   91 0E3F          CMPB    UCB$B_ONLCNT(R2),#1        ; Is it already ONLINE?
               52   1A 0E44          BGTRU   50$                        ; Yes, do not issue another PACK_ACK
                    B0 0E46 40$:     MOVW    #IO$_PACKACK!-             ; Set function to PACK_ACK
                       0E47                   IO$M_INHERLOG,-
  C0 A5   0808 8F      0E47                   CDRP$W_FUNC(R5)
               10   88 0E4C          BISB    #UQB$M_SEQ,-               ; Set the sequential flag
            2B A4      0E4E                   UQB$B_FLAGS(R4)
            E8 A5   0E 0E50          INSQUE  CDRP$L_UQB_FL(R5),-        ; Insert on unit queue
            64 B4      0E53                   @UQB$L_CDRP_BL(R4)
            50 A4   B6 0E55          INCW    UQB$W_NUM_QUE(R4)          ; Account for it
             FE42   30 0E58          BSBW    SETX                       ; Do the IO
       53   54 A5   7D 0E5B          MOVQ    CDRP$L_FR3X(R5),R3         ; Restore HQB & UQB address
               10   8A 0E5F          BICB    #UQB$M_SEQ,-               ; Clear the sequential flag
            2B A4      0E61                   UQB$B_FLAGS(R4)
       51   B0 A5   D0 0E63          MOVL    CDRP$L_MSGBUFX(R5),R1      ; Restore message buffer address
                       0E67 ; ??? bug trap
               2B   13 0E67          BEQL    47$                        ; Bug trap
                       0E69 ; ??? end bug trap
                       0E69          ASSUME  MSCP$K_ST_SUCC EQ 0
            2C 50   E8 0E69          BLBS    R0,50$                     ; Success
  00F4 8F      50   B1 0E6C          CMPW    R0,#SS$_ILLIOFUNC          ; Error, was it illegal function
               25   13 0E71          BEQL    50$                        ; Yes, okay because drive not support
       52   30 A4   D0 0E73          MOVL    UQB$L_UCB(R4),R2           ; Pick up UCB address
          00AE C2   97 0E77          DECB    UCB$B_ONLCNT(R2)           ; Drop the online count
  0254 8F      50   B1 0E7B          CMPW    R0,#SS$_VOLINV             ; Is it volume invalid?
               0C   13 0E80          BEQL    45$                        ; Yes, change error to mscp error
  01A4 8F      50   B1 0E82          CMPW    R0,#SS$_MEDOFL             ; Is it medium offline?
               05   13 0E87          BEQL    45$                        ; Yes, change error to mscp error
       50      0B   D0 0E89          MOVL    #MSCP$K_ST_DRIVE,R0        ; Make the error fatal drive error
               03   11 0E8C          BRB     46$
                       0E8E
       50      03   D0 0E8E 45$:     MOVL    #MSCP$K_ST_OFFLN,R0        ; Substitute MSCP error status
             FC40   31 0E91 46$:     BRW     SEND_PKT                   ;  and leave now
                       0E94
                       0E94 47$:     BUG_CHECK       MSCPSERV,FATAL     ; HQB invalid!!!
                       0E98
               50   D4 0E98 50$:     CLRL    R0                         ; Set success status (ONLINE)
       52   1E A3   9A 0E9A          MOVZBL  HQB$B_VEC(R3),R2           ; Pick up host slot number
    03 34 A4   52   E2 0E9E          BBSS    R2,UQB$L_HOST_ONLN(R4),60$ ; Set host online bit in unit
            0C A4   96 0EA3          INCB    UQB$B_HOST_CNT(R4)         ; Bump the host online count
       52   F8 A5   3C 0EA6 60$:     MOVZWL  CDRP$W_SLOT(R5),R2         ; Pick up unit slot number
 00 10 A3      52   E2 0EAA          BBSS    R2,HQB$L_UNIT_ONLN(R3),70$ ; Set unit-online bit in host
            0E A4   B4 0EAF 70$:     CLRW    UQB$W_STATUS(R4)           ; Set ONLINE status in unit
                       0EB2 ;
                       0EB2 ; Physical IO to unit, if any, done.  Now check if any post processing
                       0EB2 ; needed - shadow physical units only.
                       0EB2 ;
               09   E0 0EB2          BBS     #MSCP$V_UF_SSMST,-         ; Leave now if shadow master unit
      05 12 A4         0EB4                   UQB$W_UNT_FLGS(R4),CCHAR
               04   E0 0EB7          BBS     #MSCP$V_MD_SHDSP,-         ; Leave now if not shadowing
      03 0A A1         0EB9                   MSCP$W_MODIFIER(R1),CASE2A
             0455   31 0EBC CCHAR:   BRW     COPY_CHAR
```

```
                                  0EBF          .SBTTL  -                       Shadow units
                                  0EBF  ;
                                  0EBF  ; CASE 2: Deal with the shadow unit
                                  0EBF  ;
                                  0EBF  CASE2A:
           01    0C A4      91    0EBF          CMPB    UQB$B_HOST_CNT(R4),#1   ; If this unit is already ONLINE
                    F7      1E    0EC3          BGEQU   CCHAR                   ;  then leave now
                    38      BB    0EC5          PUSHR   #^M<R3,R4,R5>           ; Save some registers
                                  0EC7  ;
                                  0EC7  ; Look up the shadow unit.  If it exists we will do some checks, if it
                                  0EC7  ; does not.  We will create a new UQB for it.  If there is no space we
                                  0EC7  ; will return an error code.
                                  0EC7  ;
           52    20 A1      3C    0EC7          MOVZWL  MSCP$W_SHDW_UNT(R1),R2  ; Pick up shadow unit from command
              50    52      D0    0ECB          MOVL    R2,R0                   ; Copy
  54    52  FFFFFFF0 8F     CB    0ECE          BICL3   #^C<<1@MSU$K_UNIT_SIZ>-1>,R2,R4 ; Slot
        52    F1B5 CF44     D0    0ED6          MOVL    UNIT_VEC[R4],R2         ; Look for unit block in slot
                    07      13    0EDC          BEQL    10$                     ; Empty
                 20 A1      B1    0EDE          CMPW    MSCP$W_SHDW_UNT(R1),-
                 28 A2            0EE1                   UQB$W_UNIT(R2)         ; Hit, is it a match?
                    40      13    0EE3          BEQL    OLD_UQB                 ; Yes
                  FA74      30    0EE5  10$:    BSBW    FIND_SLOT               ; No, try more probing
                    3B      18    0EE8          BGEQ    OLD_UQB                 ; Found unit
                                  0EEA  ;
                                  0EEA  ; No match in table.  Find an empty slot.
                                  0EEA  ;
           52    20 A1      3C    0EEA          MOVZWL  MSCP$W_SHDW_UNT(R1),R2  ; Pick up shadow unit from command
  53    52  FFFFFFF0 8F     CB    0EEE          BICL3   #^C<<1@MSU$K_UNIT_SIZ>-1>,R2,R3 ; Slot
        55    F195 CF43     D0    0EF6          MOVL    UNIT_VEC[R3],R5         ; Look for unit block in slot
                    14      13    0EFC          BEQL    30$                     ; Empty
              53    10      C0    0EFE          ADDL    #MSU$K_MAX_UNIT,R3      ; Try for second probe
        55    F18A CF43     D0    0F01          MOVL    UNIT_VEC[R3],R5         ; Look for unit block in slot
                    77      13    0F07          BEQL    NEW_UQB                 ; Empty
              53    1F      D0    0F09          MOVL    #<2*MSU$K_MAX_UNIT>-1,R3; Sequential search
        55    F17F CF43     D0    0F0C  20$:    MOVL    UNIT_VEC[R3],R5         ; Pick up unit
                    6C      13    0F12  30$:    BEQL    NEW_UQB                 ; Empty
                    53      D7    0F14          DECL    R3                      ; Loop thru
              10    53      D1    0F16          CMPL    R3,#MSU$K_MAX_UNIT      ; Reached end?
                    F1      18    0F19          BGEQ    20$                     ; No, try again
                                  0F1B  ;
                                  0F1B  ; No slots left in the unit table.  Return an error
                                  0F1B  ;
                                  0F1B  CASE2A_ERR:
        50    00AA 8F       B0    0F1B          MOVW    #<5*MSCP$K_ST_SBCOD>!-
                                  0F20                   MSCP$K_ST_CNTLR,R0     ; Insufficient resources
                                  0F20
                                  0F20  CASE2A_ERR_OUT:
                    38      BA    0F20          POPR    #^M<R3,R4,R5>           ; Restore registers
                  FBAF      31    0F22          BRW     SEND_PKT
                                  0F25  ;
                                  0F25  ; Existing UQB, check characteristics
                                  0F25  ;
                                  0F25  OLD_UQB:
           54    04 AE      D0    0F25          MOVL    4(SP),R4                ; Recover physical unit UQB
                 48 A4      D1    0F29          CMPL    UQB$L_MAXBLOCK(R4),-    ; Check maxblock
                 48 A2            0F2C                   UQB$L_MAXBLOCK(R2)
                    07      12    0F2E          BNEQ    10$                     ; Failed
```

```
        20 A4   D1  0F30          CMPL    UQB$L_MEDIA_ID(R4),-       ; Check media_id
        20 A2       0F33                   UQB$L_MEDIA_ID(R2)
           07   13  0F35          BEQL    20$                        ; Okay, continue
 50  0105 8F    B0  0F37     10$: MOVW    #<8*MSCP$K_ST_SBCOD>!-
                    0F3C                   MSCP$K_ST_MFMTE,R0        ; MEDIA FORMAT ERROR
           E2   11  0F3C          BRB     CASE2A_ERR_OUT
                    0F3E     ;
                    0F3E     ; Update fields for shadowing
                    0F3E     ;
        26 A2   B6  0F3E     20$: INCW    UQB$W_SHDW_STS(R2)         ; Bump the count of units
        28 A2   B0  0F41          MOVW    UQB$W_UNIT(R2),-           ; Copy virtual unit number
        24 A4       0F44                   UQB$W_SHDW_UNT(R4)        ;  to physical unit's shadow unit field
           0E   E2  0F46          BBSS    #MSCP$V_UF_SSMEM,-         ; Set shadow set member flag
     00 12 A4       0F48                   UQB$W_UNT_FLGS(R4),30$
        28 A4   B1  0F4B     30$: CMPW    UQB$W_UNIT(R4),-           ; Is this unit number smaller than
        24 A2       0F4E                   UQB$W_SHDW_UNT(R2)        ;  any other in the shadow chain?
           0E   1E  0F50          BGEQU   40$                        ; No, skip on
        28 A4   B0  0F52          MOVW    UQB$W_UNIT(R4),-           ; Yes, update the field in shadow
        24 A2       0F55                   UQB$W_SHDW_UNT(R2)        ;  master unit field
        30 A4   D0  0F57          MOVL    UQB$L_UCB(R4),-            ; Update the field in shadow
        30 A2       0F5A                   UQB$L_UCB(R2)             ;  master UCB field
     78 A2  54  D0  0F5C          MOVL    R4,UQB$L_SHDW_LOW(R2)      ; Update the lowest unit pointer
     78 A4  52  D0  0F60     40$: MOVL    R2,UQB$L_SHDW_MST(R4)      ; Set pointer to shadow master
                    0F64     ;
                    0F64     ; Link this physical UQB into shadowing ring
                    0F64     ;
        50  62  D0  0F64          MOVL    UQB$L_SHDW_FL(R2),R0       ; Start with R0-->first entry
        52  50  D0  0F67          MOVL    R0,R2                      ; Copy
 52     04 A2   D0  0F6A     50$: MOVL    UQB$L_SHDW_BL(R2),R2       ; Get next entry
        52  50  D1  0F6E          CMPL    R0,R2                      ; End of the line?
           07   13  0F71          BEQL    60$                        ; Yes
        28 A4   B1  0F73          CMPW    UQB$W_UNIT(R4),-           ; Compare unit numbers
        28 A2       0F76                   UQB$W_UNIT-UQB$L_SHDW_FL(R2)
           F0   1F  0F78          BLSSU   50$                        ; Current is still higher than target
        62  64  0E  0F7A     60$: INSQUE  UQB$L_SHDW_FL(R4),(R2)     ; Insert ahead of this unit
         00D9   31  0F7D          BRW     CASE2A_DONE                ;  and leave now
                    0F80     ;
                    0F80     ; No existing UQB, create one
                    0F80     ;
                    0F80     ;        R1 -->  MSCP packet
                    0F80     ;        R2  =   Shadow unit number
                    0F80     ;        R3  =   Shadow slot
                    0F80     ;        R4 -->  Physical unit UQB
                    0F80     ;        (SP)--> HQB
                    0F80     ;
                    0F80     NEW_UQB:
           0E   BB  0F80          PUSHR   #^M<R1,R2,R3>              ; Save registers
        22 A1   B4  0F82          CLRW    MSCP$W_COPY_SPD(R1)        ; Ignore copy speed field
 51     9C 8F   9A  0F85          MOVZBL  #UQB$C_LEN,R1              ; Size of unit block
         F6C8   30  0F89          BSBW    ALLOMEM                    ; Go get it
        06 50   E8  0F8C          BLBS    R0,110$                    ; Okay
     5E    0C   C0  0F8F          ADDL    #3*4,SP                    ; Clear off registers
         FF86   31  0F92          BRW     CASE2A_ERR                 ; Error out
                    0F95
        55  52  D0  0F95    110$: MOVL    R2,R5                      ; New UQB pointer
 50     08 AE   D0  0F98          MOVL    8(SP),R0                   ; Restore slot number
 53     0C AE   7D  0F9C          MOVQ    12(SP),R3                  ; Pick up HQB & UQB
```

```
  00 10 A3    50   E2  0FA0  1939         BBSS    R0,HQB$L_UNIT_ONLN(R3),111$ ; Set a 'online' bit
 F0E5 CF40    55   D0  0FA5  1940 111$:   MOVL    R5,UNIT_VEC[R0]             ; Stuff it where it belongs
        82    54   D0  0FAB  1941         MOVL    R4,(R2)+                    ; Set link to first UQB
              82   D4  0FAE  1942         CLRL    (R2)+                       ; Clear out unused
        82    51   B0  0FB0  1943         MOVW    R1,(R2)+                    ; Set size field
   82   0A60  8F   B0  0FB3  1944         MOVW    #<DYN$C_SCS_UQB@8>!-        ; Set type
                       0FB8  1945                  DYN$C_SCS,(R2)+
                       0FB8  1946         ASSUME  UQB$B_HOST_CNT EQ 12
        82    01   AE  0FB8  1947         MNEGW   #1,(R2)+                    ; Set initial HOST_CNT, AVL
        82    04   B0  0FBB  1948         MOVW    #MSCP$K_ST_AVLBL,(R2)+      ; Set STATUS to AVAILABLE
                       0FBE  1949         ASSUME  UQB$W_MULT_UNT EQ UQB$B_HOST_CNT+4
              82   B4  0FBE  1950         CLRW    (R2)+                       ; Set unit
                   B0  0FC0  1951         MOVW    #<MSCP$M_UF_REPLC!-         ; Set flags - Controller Bad Block
                       0FC1  1952                  MSCP$M_UF_SSMST>,-         ;             - Shadow set master
   82   8200  8F       0FC1  1953                 (R2)+                       ; ... in the unit block
              82   D4  0FC5  1954         CLRL    (R2)+                       ; Clear out reserved
                       0FC7  1955         ASSUME  UQB$Q_UNIT_ID EQ UQB$W_MULT_UNT+8
              82   7C  0FC7  1956         CLRQ    (R2)+                       ; Clear low 32 bits of unit id
     FF A2    04   90  0FC9  1957         MOVB    #4,-1(R2)                   ; Set MSCP class code
                       0FCD  1958         ASSUME  UQB$L_MEDIA_ID EQ UQB$Q_UNIT_ID+8
     82    20 A4   D0  0FCD  1959         MOVL    UQB$L_MEDIA_ID(R4),(R2)+; Media ID
                       0FD1  1960         ASSUME  UQB$W_SHDW_UNT EQ UQB$L_MEDIA_ID+4
     82    28 A4   B0  0FD1  1961         MOVW    UQB$W_UNIT(R4),(R2)+        ; Shadow unit (lowest phys unit)
                       0FD5  1962         ASSUME  UQB$W_SHDW_STS EQ UQB$W_SHDW_UNT+2
        82    01   B0  0FD5  1963         MOVW    #1,(R2)+                    ; Shadow status (number of phys units)
                       0FD8  1964         ASSUME  UQB$W_UNIT EQ UQB$W_SHDW_STS+2
     82    04 AE   B0  0FD8  1965         MOVW    4(SP),(R2)+                 ; Set in unit number
                       0FDC  1966         ASSUME  UQB$B_SLOT EQ UQB$W_UNIT+2
     82    08 AE   90  0FDC  1967         MOVB    8(SP),(R2)+                 ; Set in slot number
              82   94  0FE0  1968         CLRB    (R2)+                       ; Shadow count
                       0FE2  1969         ASSUME  UQB$L_VOL_SER EQ UQB$B_SLOT+2
              82   D4  0FE2  1970         CLRL    (R2)+                       ; Clear volume serial
                       0FE4  1971         ASSUME  UQB$L_UCB EQ UQB$L_VOL_SER+4
     82    30 A4   D0  0FE4  1972         MOVL    UQB$L_UCB(R4),(R2)+         ; Set UCB pointer
              82   D4  0FE8  1973         CLRL    (R2)+                       ; Hosts online
   82   5306  8F   B0  0FEA  1974         MOVW    #^X5306,(R2)+               ; Length
82   4F444148 8F   D0  0FEF  1975         MOVL    #^A/HADO/,(R2)+
82   20202057 8F   D0  0FF6  1976         MOVL    #^A/W   /,(R2)+
              82   B4  0FFD  1977         CLRW    (R2)+
              82   D4  0FFF  1978         CLRL    (R2)+
                       1001  1979         ASSUME  UQB$L_MAXBLOCK EQ UQB$T_NAME+16
     82    48 A4   D0  1001  1980         MOVL    UQB$L_MAXBLOCK(R4),(R2)+; Maxblocks
                       1005  1981         ASSUME  UQB$L_START_LBN EQ UQB$L_MAXBLOCK+4
              82   D4  1005  1982         CLRL    (R2)+                       ; Start LBN
                       1007  1983         ASSUME  UQB$W_NUM_QUE EQ UQB$L_START_LBN+4
              82   7C  1007  1984         CLRQ    (R2)+                       ; Counts
                       1009  1985         ASSUME  UQB$L_FENCEL EQ UQB$W_NUM_QUE+8
              82   7C  1009  1986         CLRQ    (R2)+                       ; Fence low and high
                       100B  1987         ASSUME  UQB$L_CDRP_FL EQ UQB$L_FENCEL+8
        62    62   DE  100B  1988         MOVAL   (R2),(R2)                   ; Executing CDRP que head
        82    82   DE  100E  1989         MOVAL   (R2)+,(R2)+
                       1011  1990         ASSUME  UQB$Q_BLOCKQ EQ UQB$L_CDRP_FL+8
        62    62   DE  1011  1991         MOVAL   (R2),(R2)                   ; Blocked CDRP que head
        82    82   DE  1014  1992         MOVAL   (R2)+,(R2)+
                       1017  1993         ASSUME  UQB$Q_SHQ EQ UQB$Q_BLOCKQ+8
        62    62   DE  1017  1994         MOVAL   (R2),(R2)                   ; Special holding que head
        82    82   DE  101A  1995         MOVAL   (R2)+,(R2)+
```

```
                      101D         ASSUME  UQB$L_SHDW_LOW EQ UQB$Q_SHQ+8
          82  54  D0  101D         MOVL    R4,(R2)+                 ; Set lowest phys unit
                      1020         ASSUME  UQB$W_NUM_BLK EQ UQB$L_SHDW_LOW+4
              82  D4  1020         CLRL    (R2)+                    ; Blocked counts
                      1022         ASSUME  UQB$Q_UNITQ EQ UQB$W_NUM_BLK+4
              82  7C  1022         CLRQ    (R2)+                    ; Unit que head
              82  7C  1024         CLRQ    (R2)+
   24 A4   04 AE  B0  1026         MOVW    4(SP),UQB$W_SHDW_UNT(R4); Set phys unit shadow unit field
      78 A4   55  D0  102B         MOVL    R5,UQB$L_SHDW_MST(R4)    ; Set pointer to shadow master
              0E  E2  102F         BBSS    #MSCP$V_UF_SSMEM,-       ; Set shadow set member flag
     00 12 A4         1031                 UQB$W_UNT_FLGS(R4),120$
                      1034  ;
                      1034  ; Now link shadow UQB into unit chain
                      1034  ;
   51    F048 CF  DE  1034  120$:  MOVAL   UQB_LIST,R1              ; UQB list head
      50      51  D0  1039         MOVL    R1,R0                    ; Copy
    51    04 A1   D0  103C  130$:  MOVL    4(R1),R1                 ; Get next entry, backwards
          51  50  D1  1040         CMPL    R0,R1                    ; End of the line?
              07  13  1043         BEQL    140$                     ; Yes
   A8 A1   04 AE  B1  1045         CMPW    4(SP),UQB$W_UNIT- -
                      104A                 UQB$Q_UNITQ(R1)          ; Compare unit numbers
              F0  1F  104A         BLSSU   130$                     ; Current is still higher than target
   61    0080 C5  0E  104C  140$:  INSQUE  UQB$Q_UNITQ(R5),(R1)     ; Insert ahead of this unit
                      1051  ;
                      1051  ; Send AVAIL attention message announcing the new shadow unit
                      1051  ;
          52  55  D0  1051         MOVL    R5,R2                    ; Transfer UQB
            F4B1  30  1054         BSBW    SEND_ATTN                ; Send the attn message
              0E  BA  1057         POPR    #^M<R1,R2,R3>            ; Restore registers
                      1059
                      1059  CASE2A_DONE:
              38  BA  1059         POPR    #^M<R3,R4,R5>
           22 A1  B5  105B         TSTW    MSCP$W_COPY_SPD(R1)      ; Is this a shadow copy
              05  12  105E         BNEQ    ONL_COPY                 ; Yes, go do it
              50  D4  1060  CCX:   CLRL    R0                       ; Set success status
            02AF  31  1062         BRW     COPY_CHAR                ; Transfer the ONLINE information
```

```
                                 1065        .SBTTL  -                 ONLINE shadow copy
                                 1065   ;
                                 1065   ;       R1 -->  MSCP packet
                                 1065   ;       R4 -->  UQB of physical unit
                                 1065   ;       R5 -->  CDRP for ONL command
                                 1065   ;
                                 1065
                     00000024    1065           WRT_UQB = ^X24
                     00000028    1065           RED_UQB = ^X28
                     0000002C    1065           MST_UQB = ^X2C
                                 1065
                                 1065   ONL_COPY:
                 20        8A    1065           BICB    #1@CDRP$V_NOHSTBUF,-    ; Clear special flag
              FA A5              1067                    CDRP$B_FLAGS(R5)
              0C A4        95    1069           TSTB    UQB$B_HOST_CNT(R4)      ; Is it already online and copied?
                 F2        14    106C           BGTR    CCX                     ; Yes, leave with success
                 10        88    106E           BISB    #UQB$M_SEQ,-            ; Set seq mode on write (physical)
              2B A4              1070                    UQB$B_FLAGS(R4)        ;   unit - lock IO until this is done
           52    51        D0    1072           MOVL    R1,R2                   ; Transfer MSCP packet pointer
        24 A2    54        D0    1075           MOVL    R4,WRT_UQB(R2)          ; Store the write UQB
        50    E8 A5        0F    1079           REMQUE  CDRP$L_UQB_FL(R5),R0    ; Pull CDRP off physical unit que
              50 A4        B7    107D           DECW    UQB$W_NUM_QUE(R4)       ;  and show it
      0088 C4    55        D0    1080           MOVL    R5,UQB$L_CPY_CDRP(R4)   ; Save a pointer to this cdrp
        54    78 A4        D0    1085           MOVL    UQB$L_SHDW_MST(R4),R4   ; Get pointer to shadow master
        2C A2    54        D0    1089           MOVL    R4,MST_UQB(R2)          ; Store the shadow master UQB
                                 108D   ;
                                 108D   ; Handle the excluded block area
                                 108D   ;
        51    18 A2        D0    108D           MOVL    MSCP$W_EXCL_LBC(R2),R1  ; Pick up count
        50    14 A2        D0    1091           MOVL    MSCP$L_EXCL_LBA(R2),R0  ; Pick up exclude address
                 09        12    1095           BNEQ    10$                     ; Non-zero
        1C A2    51        D0    1097           MOVL    R1,MSCP$L_LBN(R2)       ; Set as starting address
               01FF        30    109B           BSBW    GET_MAX_BLOCK           ; Find the maximum block
                 10        11    109E           BRB     20$                     ; Go set the byte count
                                 10A0
              1C A2        D4    10A0   10$:    CLRL    MSCP$L_LBN(R2)          ; Set to start at beginning
           51    50        C0    10A3           ADDL    R0,R1                   ; Find end of exclude area
        48 A4    51        D1    10A6           CMPL    R1,UQB$L_MAXBLOCK(R4)   ; Is it the end of the disk?
                 04        1E    10AA           BGEQU   20$                     ; Yes
                 04        88    10AC           BISB    #UQB$M_2PASS,-
              2B A4              10AE                    UQB$B_FLAGS(R4)        ; No, set a flag for later
  0C A2    50    09        78    10B0   20$:    ASHL    #9,R0,MSCP$L_BYTE_CNT(R2) ; Convert to number of bytes
                                 10B5   ;
                                 10B5   ; Now check that there are no IO's in progress on the shadow unit.
                                 10B5   ; If there are we must block the copy CDRP, set sequential mode (rundown)
                                 10B5   ; and wait for the last IO to wake the CDRP.
                                 10B5   ;
              50 A4        B5    10B5   30$:    TSTW    UQB$W_NUM_QUE(R4)       ; Check the shadow master for IO
                 09        13    10B8           BEQL    40$                     ; None, all okay
                 10        88    10BA           BISB    #UQB$M_SEQ,-            ; Set sequential mode while waiting
              2B A4              10BC                    UQB$B_FLAGS(R4)
                                 10BE           BLOCKH                          ; IO in progress, rundown and wait
                 F2        11    10C1           BRB     30$
                                 10C3
                 01        88    10C3   40$:    BISB    #UQB$M_CIP,-            ; Set the copy in progress flag
              2B A4              10C5                    UQB$B_FLAGS(R4)
           51    24 A2     D0    10C7           MOVL    WRT_UQB(R2),R1          ; Write UQB pointer
```

```
           01   88  10CB          BISB    #UQB$M_CIP,-              ; Set the copy in progress flag
        2B A1       10CD                  UQB$B_FLAGS(R1)
                    10CF  ;
                    10CF  ; Start of read/write loop
                    10CF  ;
                    10CF  ONL_LOOP:
                    10CF  ;       BBS     #CDRP$V_ABORT,-           ; Leave if aborted
                    10CF  ;               CDRP$B_FLAGS(R5),90$
           10   8A  10CF          BICB    #UQB$M_SEQ,-              ; Turn off sequential mode
        2B A4       10D1                  UQB$B_FLAGS(R4)
     50    0C   B0  10D3          MOVW    #IO$_READPBLK,R0          ; Set up for read
         06E6   30  10D6          BSBW    SETUP                     ; Common set up
     52 B0 A5   D0  10D9          MOVL    CDRP$L_MSGBUFX(R5),R2     ; Refresh MSCP packet pointer
        C4 A5   D0  10DD          MOVL    CDRP$L_UQB(R5),-          ; Pick up read UQB
        28 A2       10E0                  RED_UQB(R2)               ;   and refresh the copy
                    10E2
                    10E2  ONL_READ:
           0C   B0  10E2          MOVW    #IO$_READPBLK,-           ; Set up for read
        C0 A5       10E4                  CDRP$W_FUNC(R5)
     52 B0 A5   D0  10E6          MOVL    CDRP$L_MSGBUFX(R5),R2     ; Refresh MSCP packet pointer
     54 28 A2   D0  10EA          MOVL    RED_UQB(R2),R4            ; Pick up read UQB
  C4 A5    54   D0  10EE          MOVL    R4,CDRP$L_UQB(R5)         ; Stuff it in CDRP
        30 A4   D0  10F2          MOVL    UQB$L_UCB(R4),-           ; Refresh the read UCB
        BC A5       10F5                  CDRP$L_UCB(R5)
     54 2C A2   D0  10F7          MOVL    MST_UQB(R2),R4            ; Pick up shadow master UQB
        48 A5   D0  10FB          MOVL    CDRP$L_LBN(R5),-          ; Stuff the starting fence lbn
        58 A4       10FE                  UQB$L_FENCEL(R4)
50   D2 A5 F7 8F  78  1100        ASHL    #-9,CDRP$L_BCNT(R5),R0    ; Pick up the byte count
  58 A4    50   C1  1106          ADDL3   R0,UQB$L_FENCEL(R4),-     ; Caclulate the high fence
        5C A4       110A                  UQB$L_FENCEH(R4)
  D6 A5    02   B0  110C          MOVW    #2,CDRP$W_STATE(R5)       ; Set the state
         094A   30  1110          BSBW    DO_DISK                   ; Set up fro disk transfer
           04   E0  1113          BBS     #CDRP$V_ABORT,-           ; Leave if aborted
        FA A5       1115                  CDRP$B_FLAGS(R5),-
           2B       1117                  ONL_LOOP_CHK
        C0 A5   B4  1118          CLRW    CDRP$W_FUNC(R5)           ; Allow modifier in error routine
        03 50   E8  111B          BLBS    R0,ONL_WRITE              ; Success
         00E8   31  111E          BRW     ONL_READ_ERR              ; Attempt to recover the error
                    1121  ;
                    1121  ; Read has finished, now switch some parameters for writes
                    1121  ;
                    1121  ONL_WRITE:
           0B   A8  1121          BISW    #IO$_WRITEPBLK,-          ; Set up for write
        C0 A5       1123                  CDRP$W_FUNC(R5)
     52 B0 A5   D0  1125          MOVL    CDRP$L_MSGBUFX(R5),R2     ; Grab the MSCP packet pointer
     54 24 A2   D0  1129          MOVL    WRT_UQB(R2),R4            ; Now get the write UQB
  C4 A5    54   D0  112D          MOVL    R4,CDRP$L_UQB(R5)         ; Stuff it in CDRP
        30 A4   D0  1131          MOVL    UQB$L_UCB(R4),-           ; Transfer the write UCB
        BC A5       1134                  CDRP$L_UCB(R5)
  D6 A5    04   B0  1136          MOVW    #4,CDRP$W_STATE(R5)       ; Bump the state
         0920   30  113A          BSBW    DO_DISK                   ; Do the disk transfer
        03 50   E8  113D          BLBS    R0,ONL_LOOP_CHK           ; Success
         0106   31  1140          BRW     ONL_WRITE_ERR             ; Error out
                    1143  ;
                    1143  ; Write has finished, now check if need to loop again or possibly
                    1143  ; wait for a pending write.
                    1143  ;
```

```
                            1143 ONL_LOOP_CHK:
         52    B0 A5   D0   1143          MOVL    CDRP$L_MSGBUFX(R5),R2   ; Grab pointer to MSCP packet
         54    2C A2   D0   1147          MOVL    MST_UQB(R2),R4          ; Pick up master UQB
   E0 A5       D2 A5   C0   114B          ADDL    CDRP$L_BCNT(R5), -      ; Calc accumulated bytes xfer
                            1150                   CDRP$L_ABCNT(R5)
                  04   E1   1150          BBC     #CDRP$V_ABORT,-         ; Leave if aborted
         03 FA A5           1152                   CDRP$B_FLAGS(R5),5$
                0111   31   1155          BRW     ONL_ABORT_ERR
                            1158
50    E4 A5    E0 A5   C3   1158 5$:      SUBL3   CDRP$L_ABCNT(R5), -
                            115E                   CDRP$L_OBCNT(R5),R0    ; Calc how much left to do
                  26   15   115E          BLEQ    30$                     ; None, we are finished
51    D2 A5    F7 8F   78   1160          ASHL    #-9,CDRP$L_BCNT(R5),R1  ; Select the correct disk block
         51    48 A5   C0   1166          ADDL    CDRP$L_LBN(R5),R1       ; Compute next start LBN
      48 A5       51   D0   116A          MOVL    R1,CDRP$L_LBN(R5)       ; Store it away
               D2 A5   D1   116E          CMPL    CDRP$L_BCNT(R5),-       ; If the bytecount has been reduced
               B4 A5        1171                   CDRP$L_BUFLEN(R5)      ;  as a result of error recovery...
                  05   1E   1173          BGEQU   10$                     ; Then
               B4 A5   D0   1175          MOVL    CDRP$L_BUFLEN(R5),-     ;  Set it back to original
               D2 A5        1178                   CDRP$L_BCNT(R5)
      D2 A5       50   D1   117A 10$:     CMPL    R0,CDRP$L_BCNT(R5)      ; If piece left less than BCNT
                  04   18   117E          BGEQ    20$                     ; .. then ..
      D2 A5       50   D0   1180          MOVL    R0,CDRP$L_BCNT(R5)      ; Use it
                  21   11   1184 20$:     BRB     60$
                            1186 ;
                            1186 ; Finished all the copy specified in the packet, now check if a possible
                            1186 ; second pass is needed because the exclude area is in the middle of the disk.
                            1186 ;
                  02   E4   1186 30$:     BBSC    #UQB$V_2PASS,-          ; Skip if no second pass needed
         03 2B A4           1188                   UQB$B_FLAGS(R4),50$
                0053   31   118B 40$:     BRW     ONL_FINI
                            118E
         50    14 A2   7D   118E 50$:     MOVQ    MSCP$L_EXCL_LBA(R2),R0  ; Pick the exclude parameters
            51    50   C0   1192          ADDL    R0,R1                   ; Starting LBN after exclude area
      48 A5       51   D0   1195          MOVL    R1,CDRP$L_LBN(R5)       ; Put it in CDRP
                0101   30   1199          BSBW    GET_MAX_BLOCK           ; Find the maximum blocks to copy
    50    50      09   78   119C          ASHL    #9,R0,R0                ; Convert to bytes for next copy loop
      E4 A5       50   D0   11A0          MOVL    R0,CDRP$L_OBCNT(R5)     ; Store in IRP
               E0 A5   D4   11A4          CLRL    CDRP$L_ABCNT(R5)        ; Reset accumulated count
                            11A7 ;
                            11A7 ; Another round is needed.  Check if there is a write pending, if so
                            11A7 ; stall the copy and do the write.  At the end of the write, the copy
                            11A7 ; will be restarted.
                            11A7 ;
                            11A7 ;        R0   =    #bytes left to copy
                            11A7 ;        R1   =    LBN to start copy at
                            11A7 ;        R4 -->    Master UQB
                            11A7 ;
                  03   E4   11A7 60$:     BBSC    #UQB$V_CMDPEND,-        ; Go around again if no write pend
         03 2B A4           11A9                   UQB$B_FLAGS(R4),70$
                FF33   31   11AC          BRW     ONL_READ
                            11AF
      0C A2       50   D0   11AF 70$:     MOVL    R0,MSCP$L_BYTE_CNT(R2)  ; Update copy in packet
      1C A2       51   D0   11B3          MOVL    R1,MSCP$L_LBN(R2)       ; Ditto
      58 A4       01   CE   11B7          MNEGL   #1,UQB$L_FENCEL(R4)     ; Set special value for tests in SETUP
0C A5     FF10 CF      9E   11BB          MOVAB   ONL_LOOP,CDRP$L_FPC(R5) ; Store resume point
                  10   88   11C1          BISB    #UQB$M_SEQ,-            ; Set sequential mode while waiting
```

```
      2B A4        11C3                 UQB$B_FLAGS(R4)
       00F0    30  11C5         BSBW    BLOCK_CDRP                  ; Block the CDRP
                   11C8  ;
                   11C8  ; Now empty the special holding queue.
                   11C8  ;
                   11C8  EMPTY_SHQ:
50    70 B4    0F  11C8         REMQUE  @UQB$Q_SHQ(R4),R0           ; Remove from special que
         12    1D  11CC         BVS     10$                         ; None, leave
55    18 A0    DE  11CE         MOVAL   CDRP$L_FQFL- -
                   11D2                 CDRP$L_UQB_FL(R0),R5        ; Set new CDRP ptr
         54    DD  11D2         PUSHL   R4                          ; Save old UQB
53    10 A5    7D  11D4         MOVQ    CDRP$L_FR3(R5),R3           ; Restore old context
      0C B5    16  11D8         JSB     @CDRP$L_FPC(R5)             ; Finish off the pkt
         54  8ED0  11DB         POPL    R4                          ; Restore old UQB
         E8    11  11DE         BRB     EMPTY_SHQ                   ; No, get next command
                   11E0
               05  11E0  10$:   RSB                                 ; Finish this thread
                   11E1  ;
                   11E1  ; All copying is done.  Clean up status flags and do the copy of parameters.
                   11E1  ; At this point the CDRP is still queued to shadow master UQB.  We must
                   11E1  ; remque it, and adjust other fields so this CDRP looks like it was on
                   11E1  ; the physical unit que.
                   11E1  ;
                   11E1  ONL_FINI:
         50    D4  11E1         CLRL    R0                          ; Set good status
                   11E3
                   11E3  ONL_FINIX:
         50    DD  11E3         PUSHL   R0                          ; Save status
50    E8 A5    0F  11E5         REMQUE  CDRP$L_UQB_FL(R5),R0        ; Strip it from que
         60    D4  11E9         CLRL    (R0)                        ;  show it not queued
      50 A4    B7  11EB         DECW    UQB$W_NUM_QUE(R4)           ;  and account for it
       009B    30  11EE         BSBW    CLEAN_UP_ONLCPY             ; Do a clean up
         D5    10  11F1         BSBB    EMPTY_SHQ                   ; Finish off and held commands
         50  8ED0  11F3         POPL    R0                          ; Recover status
51    B0 A5    D0  11F6         MOVL    CDRP$L_MSGBUFX(R5),R1       ; Recover MSCP pointer
54    24 A1    D0  11FA         MOVL    WRT_UQB(R1),R4              ;  and write unit UQB pointer
78 A5    54    D0  11FE         MOVL    R4,CDRP$L_MST_UQB(R5)       ; Stuff it CDRP
  0088 C4      D4  1202         CLRL    UQB$L_CPY_CDRP(R4)          ; Clear out pointer
       010B    31  1206         BRW     COPY_CHAR                   ;  and leave
```

```
                                1209        .SBTTL  -               ERROR HANDLING
                                1209
                                1209  ;++
                                1209  ;
                                1209  ; Read error recovery for bad blocks.  First we attempt an ordinary read
                                1209  ; read repair operation.  That operation can return with four cases:
                                1209  ;
                                1209  ;       1) R0 = SS$_NORMAL: Read partially succeded, up to a parity or
                                1209  ;               forcederror, but all blocks before the one in error have
                                1209  ;               been successfully read into our buffer  Continue the loop
                                1209  ;               and write out the good blocks to the target unit. The next
                                1209  ;               read will either succeded or will encounter a forcederror/
                                1209  ;               parity error immediately on the first block.  If the latter
                                1209  ;               then we retry until success or no more units.  If no more
                                1209  ;               units we will return here with SS$_PARITY or SS$_FORCEDERROR.
                                1209  ;
                                1209  ;       2) R0 = SS$_SHACHASTA: An unrepairable, fatal error occurred on one
                                1209  ;               of the source units or the target unit.  We abort the copy
                                1209  ;               operation, failout the offending unit, and report to host.
                                1209  ;
                                1209  ;       3) R0 = SS$_PARITY: This error means we have a parity error in the
                                1209  ;               same LBN of all the source units.  We then just write out
                                1209  ;               one block of whatever is in the buffer to the target unit.
                                1209  ;               Hopefully, the host's file structure will be able to deal
                                1209  ;               with this crazy block.  After the one block write, we continue
                                1209  ;               the normal loop.  The shadow set remains intact.
                                1209  ;
                                1209  ;       4) R0 = SS$_FORCEDERROR: As case 3 above, only the block is written
                                1209  ;               out with a FORCERR modifier.  The host's file structure will
                                1209  ;               eventually catch this.
                                1209  ;
                                1209  ;--
                                1209
                                1209        .ENABLE LSB
                                1209
                                1209  ONL_READ_ERR:
50 A5    FED5 CF      9E        1209        MOVAB   ONL_READ,CDRP$L_FPCX(R5); Where to send blocked reads
              039D    30        120F        BSBW    READ_REPAIR             ; Attempt read repair
              17 50   E9        1212        BLBC    R0,10$                  ; A failure of some sort
      52      B0 A5   D0        1215        MOVL    CDRP$L_MSGBUFX(R5),R2   ; Refresh MSCP packet pointer
              C4 A5   D0        1219        MOVL    CDRP$L_UQB(R5),-        ; Pick up new read UQB
              28 A2             121C                 RED_UQB(R2)            ;  and refresh the copy
                 55   DD        121E        PUSHL   R5                      ; Save R5
      54      2C A2   D0        1220        MOVL    MST_UQB(R2),R4          ; Pick up master UQB
              F94E    30        1224        BSBW    DRAIN_BLOCKQ            ; See if anything blocked while repair
              55 8ED0           1227        POPL    R5                      ; Restore R5
                 1A   11        122A        BRB     30$                     ; Repair succeded, continue
                                122C  ;
                                122C  ; Either PARITY or FORCEDERROR.  Set up for a one block write.
                                122C  ;
   01F4 8F    50      B1        122C  10$:  CMPW    R0,#SS$_PARITY          ; Is it parity error?
              0D      13        1231        BEQL    20$                     ; Yes, adjust buffer
   2144 8F    50      B1        1233        CMPW    R0,#SS$_FORCEDERROR     ; No, is it forced errro?
              26      12        1238        BNEQ    40$                     ; No, real hard error
         0100 8F      B0        123A        MOVW    #IO$M_FORCERR,-         ; Set for writing forced error
           C0 A5                123E                 CDRP$W_FUNC(R5)
D2 A5    0200 8F      3C        1240  20$:  MOVZWL  #512,CDRP$L_BCNT(R5)    ; Adjust byte counts
```

```
                FED8   31  1246  30$:    BRW     ONL_WRITE                       ; Return inline
                           1249
                           1249  ;
                           1249  ; Write error recovery for bad blocks
                           1249  ;
                           1249  ONL_WRITE_ERR:
        01F4 8F    50  B1  1249          CMPW    R0,#SS$_PARITY                  ; Is it parity error?
                   10  12  124E          BNEQ    40$                             ; No, hard error
  51     DA A5   F7 8F 78  1250          ASHL    #-9,CDRP$L_IOST1+2(R5),R1      ; Number of blocks transfered
                   51  D6  1256          INCL    R1                              ; Skip over bad block
      D2 A5   51   09  78  1258          ASHL    #9,R1,CDRP$L_BCNT(R5)           ; Update number of bytes to copy
                 FEE3  31  125D          BRW     ONL_LOOP_CHK                    ; Return inline
                           1260  ;
                           1260  ; Fatal error - take care of shadow set housekeeping
                           1260  ;
           54   C4 A5  D0  1260  40$:    MOVL    CDRP$L_UQB(R5),R4               ; Pick up pointer to phy unit
                 FAE0  30  1264          BSBW    AVAIL_ALL                       ; Throw it out of shadow set
                   02  11  1267          BRB     50$                             ; Finish up
                           1269
                           1269  ONL_ABORT_ERR:
                   54  D4  1269          CLRL    R4                              ; Always want the write unit
           52   B0 A5  D0  126B  50$:    MOVL    CDRP$L_MSGBUFX(R5),R2           ; Recover MSCP pointer
           50   24 A2  D0  126F          MOVL    WRT_UQB(R2),R0                  ; Pick up write (target) UQB
              54   50  D1  1273          CMPL    R0,R4                           ; Was the error on the write?
                   06  13  1276          BEQL    60$                             ; Yes, then just fail it out
              54   50  D0  1278          MOVL    R0,R4                           ; No, we must also fail write out
                 FAC9  30  127B          BSBW    AVAIL_ALL                       ; Throw it out of shadow set
              50   0C  3C  127E  60$:    MOVZWL  #MSCP$K_ST_SHST,R0              ; Set error code
           52   B0 A5  D0  1281          MOVL    CDRP$L_MSGBUFX(R5),R2           ; Recover MSCP pointer
           54   2C A2  D0  1285          MOVL    MST_UQB(R2),R4                  ; Recover master UQB pointer
                 FF57  31  1289          BRW     ONL_FINIX                       ; Leave
                           128C
                           128C          .DISABLE LSB
                           128C  ;
                           128C  ; CLEAN_UP_ONLCPY
                           128C  ;
                           128C  CLEAN_UP_ONLCPY:
           52   B0 A5  D0  128C          MOVL    CDRP$L_MSGBUFX(R5),R2           ; Recover MSCP pointer
                   11  8A  1290          BICB    #<UQB$M_CIP!UQB$M_SEQ>,-        ; Clear flags on shadow unit
                2B A4      1292           UQB$B_FLAGS(R4)
           51   24 A2  D0  1294          MOVL    WRT_UQB(R2),R1                  ; Grab write unit UQB pointer
                       8A  1298          BICB    #<UQB$M_CIP!UQB$M_SEQ!-         ; Clear flags on physical unit
                           1299            UQB$M_BLOCKED>,-
        2B A1      13      1299           UQB$B_FLAGS(R1)
                       05  129C          RSB
                           129D
                           129D  ;
                           129D  ; GET_MAX_BLOCK
                           129D  ;
                           129D          .ENABLE LSB
                           129D
                           129D  GET_MAX_BLOCK:
                   52  DD  129D          PUSHL   R2                              ; Get a register
  50       48 A4   51  C3  129F          SUBL3   R1,UQB$L_MAXBLOCK(R4),R0        ; Calculate number of blocks to copy
           52   30 A4  D0  12A4          MOVL    UQB$L_UCB(R4),R2                ; Pick up UCB pointer
                   08  E0  12A8          BBS     #DEV$V_RCT,-                    ; Skip next if no last track device
             07 38 A2      12AA           UCB$L_DEVCHAR(R2),10$
```

```
     52   44 A2     9A  12AD  2358          MOVZBL  UCB$B_SECTORS(R2),R2      ; Pick up #sectors in last track
        50    52    C2  12B1  2359          SUBL    R2,R0                     ; Subtract it out from total
              52  8ED0  12B4  2360 10$:     POPL    R2                        ; Restore register
                    05  12B7  2361 20$:     RSB
                        12B8  2362 ;
                        12B8  2363 ; BLOCK_CDRP
                        12B8  2364 ;
                        12B8  2365 BLOCK_CDRP:
     10 A5    53    7D  12B8  2366          MOVQ    R3,CDRP$L_FR3(R5)         ; Save R3 & R4
     50    E8 A5    0F  12BC  2367          REMQUE  CDRP$L_UQB_FL(R5),R0      ; Remove from head of execute queue and
           50 A4    B7  12C0  2368          DECW    UQB$W_NUM_QUE(R4)         ;   (make count agree)
     68 A4    60    0E  12C3  2369          INSQUE  (R0),UQB$Q_BLOCKQ(R4)     ;  insert at head of block queue
           7C A4    B6  12C7  2370          INCW    UQB$W_NUM_BLK(R4)         ;   (make count agree)
     D6 A5    08    B0  12CA  2371          MOVW    #8,CDRP$W_STATE(R5)       ; Show proper state
                        12CE  2372
                        12CE  2373 DEALLOC_BUFF:
              01    E5  12CE  2374          BBCC    #CDRP$V_BUFFER,-          ; Clear the buffer flag
        E4 FA A5        12D0  2375                   CDRP$B_FLAGS(R5),20$
     51    B4 A5    D0  12D3  2376          MOVL    CDRP$L_BUFLEN(R5),R1      ; Pick up the buffer
     50    B8 A5    D0  12D7  2377          MOVL    CDRP$L_BUFADR(R5),R0      ; Ditto
                        12DB  2378 ; ??? BUG TRAP
   F1F0 CF    50    D1  12DB  2379          CMPL    R0,BUFF_ADDR              ; Does address precede allocation region
              16    1F  12E0  2380          BLSSU   70$                       ; Branch if yes
     53    51    50    C1  12E2  2381          ADDL3   R0,R1,R3                  ; End of region+1
   F1E9 CF    53    D1  12E6  2382          CMPL    R3,BUFF_END               ; Is it beyond+1
              0B    1A  12EB  2383          BGTRU   70$
                        12ED  2384 ; ??? END BUG TRAP
     53   ED5F CF   9E  12ED  2385          MOVAB   BUFF_HEAD,R3              ; Allocation list head
                        12F2  2386
    00000000'9F     17  12F2  2387          JMP     @#EXE$DEALLOCATE          ; Get rid of buffer
                        12F8  2388
                        12F8  2389 70$:     BUG_CHECK       MSCPSERV,FATAL    ; Invalid buffer deallocation
                        12FC  2390
                        12FC  2391          .DISABLE LSB
```

```
                        12FC         .SBTTL  -       10 - SET UNIT CHARACTERISTICS
                        12FC   ;*************************************************************
                        12FC   ;                                                           *
                        12FC   ;       # 10 -  S E T   U N I T   C H R                     *
                        12FC   ;                                                           *
                        12FC   ;*************************************************************
                        12FC   ;
                        12FC         .ENABLE LSB
                        12FC
                        12FC   SET_UNIT_CHR:
            50     B5   12FC         TSTW    R0                          ; Check status
            3C     12   12FE         BNEQ    50$                         ; Either AVAIL or OFFLINE
                        1300   ;
                        1300   ; Unit ONLINE to this host - set the unit characteristics
                        1300   ;
            02     E0   1300   10$:  BBS     #MSCP$V_MD_STWRP,-          ; Check for correct modifier
      05 0A A1          1302                 MSCP$W_MODIFIER(R1),20$
            0C     E5   1305         BBCC    #MSCP$V_UF_WRTPS,-          ; Not allowed, remove it
      00 0E A1          1307                 MSCP$W_UNT_FLGS(R1),20$
         0E A1     A8   130A   20$:  BISW    MSCP$W_UNT_FLGS(R1),-
         12 A4          130D                 UQB$W_UNT_FLGS(R4)          ; Host settable flags
24 A4    20 A1     B0   130F         MOVW    MSCP$W_SHDW_UNT(R1), -
                        1314                 UQB$W_SHDW_UNT(R4)          ; Shadow unit
                        1314   ;
                        1314   ; Move in returned info to end packet
                        1314   ;
                        1314   COPY_CHAR:
24 A1    48 A4     D0   1314         MOVL    UQB$L_MAXBLOCK(R4), -       ; Size
                        1319                 MSCP$L_UNT_SIZE(R1)
            19     12   1319         BNEQ    40$                         ; All okay
   52    30 A4     D0   131B         MOVL    UQB$L_UCB(R4),R2            ; Pick up UCB pointer
            08     12   131F         BNEQ    30$                         ; There is one
   52    78 A4     D0   1321         MOVL    UQB$L_SHDW_LOW(R4),R2       ; For shadow master, use first phys unit
   52    30 A2     D0   1325         MOVL    UQB$L_UCB(R2),R2            ; Pick up UCB pointer
       00B0 C2     D0   1329   30$:  MOVL    UCB$L_MAXBLOCK(R2),-        ; Get it from the UCB
         48 A4          132D                 UQB$L_MAXBLOCK(R4)
24 A1    48 A4     D0   132F         MOVL    UQB$L_MAXBLOCK(R4), -       ; Size (try again)
                        1334                 MSCP$L_UNT_SIZE(R1)
28 A1    2C A4     D0   1334   40$:  MOVL    UQB$L_VOL_SER(R4), -
                        1339                 MSCP$L_VOL_SER(R1)          ; VOL_ID
          F74F     31   1339         BRW     FIN_UNIT_STS                ; Finish packet elsewhere
                        133C
          F795     31   133C   50$:  BRW     SEND_PKT                    ; Return AVAIL/OFFLINE error
                        133F
                        133F         .DISABLE lsb
```

```
                 133F        .SBTTL  -      11 - DETERMINE ACCESS PATHS
                 133F  ;*********************************************************
                 133F  ;                                                       *
                 133F  ;          # 11 - D E T   A C C   P A T H               *
                 133F  ;                                                       *
                 133F  ;*********************************************************
                 133F  ;
                 133F  DET_ACC_PATH:
      F790   31  133F        BRW     SEND_SUCCESS            ; NOP
```

```
                        1342          .SBTTL  NON-SEQUENTIAL NON-BUFFER class commands
                        1342  ;
                        1342  ;       R0   scratch
                        1342  ;       R1 = Ptr to message buffer
                        1342  ;       R2 = low 3 bits of command
                        1342  ;       R3 = Ptr to HQB
                        1342  ;       R4 = Unit number
                        1342  ;       R5 = Ptr to CDRP
                        1342  ;
      54    78 A5   D0  1342  NSEQ:   MOVL    CDRP$L_MST_UQB(R5),R4     ; Recover UQB pointer
      50    0E A4   3C  1346          MOVZWL  UQB$W_STATUS(R4),R0       ; Pick up status
               03   13  134A          BEQL    10$                       ; Success - ONLINE
             F785   31  134C          BRW     SEND_PKT                  ; Return error code
                        134F
      50    30 A4   D0  134F  10$:    MOVL    UQB$L_UCB(R4),R0          ; Ucb address
06 64 A0       0E   E1  1353          BBC     #UCB$V_MNTVERIP,UCB$L_STS(R0),15$
                        1358                                            ; Branch if mount verification is not
                        1358                                            ; progress
         50    04   D0  1358          MOVL    #MSCP$K_ST_AVLBL,R0       ; mark unit as available
             F776   31  135B          BRW     SEND_PKT                  ; return unit available
                        135E
                    93  135E  15$:    BITB    #<UQB$M_SEQ!-             ; Check if in SEQ mode
                        135F                    UQB$M_RIP!-             ;  or in repair
                        135F                    UQB$M_BLOCKED>,-        ;  or blocked
         2B A4     32   135F                   UQB$B_FLAGS(R4)
               03   13  1362          BEQL    20$                       ; If so ...
                        1364          BLOCKT                            ;  then block the command
                        1367
                        1367  20$:    CASE    CDRP$B_OPCODE(R5),-       ; Dispatch on command subtype
                        1367                   TYPE=B,LIMIT=#MSCP$K_OP_ACCES-
                        1367                  <ACCESS,-                 ; 16-ACCESS
                        1367                   COMP_CTRL_DAT,-          ; 17-COMPARE CONTROLLER DATA
                        1367                   ERASE,-                  ; 18-ERASE
                        1367                   FLUSH,-                  ; 19-FLUSH
                        1367                   REPLACE>                 ; 20-REPLACE
             F5B0   31  1376          BRW     ERR_1
```

```
                   1379      .SBTTL  -       16 - ACCESS
                   1379 ;**************************************************
                   1379 ;                                                 *
                   1379 ;         # 16 -  A C C E S S                     *
                   1379 ;                                                 *
                   1379 ;**************************************************
                   1379 ;
                   1379 ACCESS:
        20   88    1379          BISB    #1@CDRP$V_NOHSTBUF,-     ; Set for no host buffer
     FA A5         137B                  CDRP$B_FLAGS(R5)
                   137D          BUG_CHECK       MSCPSERV,FATAL   ; ?? Tied off here and in opcode table
      005D   31    1381          BRW     READ                     ; And join read code
```

```
                1384          .SBTTL  -       17 - COMPARE CONTROLLER DATA
                1384     ;***************************************************************
                1384     ;                                                               *
                1384     ;          # 17 -  C O M P   C T R L   D A T                    *
                1384     ;                                                               *
                1384     ;***************************************************************
                1384     ;
                1384     COMP_CTRL_DAT:
  F74B    31    1384             BRW     SEND_SUCCESS            ; NOP
```

```
                         1387       .SBTTL  -       18 - ERASE
                         1387  ;*********************************************************
                         1387  ;                                                        *
                         1387  ;              # 18 - E R A S E                          *
                         1387  ;                                                        *
                         1387  ;*********************************************************
                         1387  ;
                         1387  ERASE:
                    B3   1387       BITW    #<MSCP$M_UF_WRTPS!-
                         1388                 MSCP$M_UF_WRTPH>,-
   12 A4   3000 8F       1388               UQB$W_UNT_FLGS(R4)      ; If WRITE_PROTECT
                06  13   138D       BEQL    10$
           50   06  D0   138F       MOVL    #MSCP$K_ST_WRTPR, R0    ; Set error status
              F73F  31   1392       BRW     SEND_PKT                ; ... and exit
                         1395
                20  88   1395  10$: BISB    #1@CDRP$V_NOHSTBUF,-    ; Set for no host buffer
             FA A5       1397               CDRP$B_FLAGS(R5)
           50   15  B0   1399       MOVW    #IO$_DSE,R0             ; Indicate write
              0360  31   139C       BRW     DO_WRT                  ; Join WRITE command
```

```
               139F        .SBTTL  -       19 - FLUSH
               139F  ;***************************************************************
               139F  ;                                                              *
               139F  ;         # 19 -  F L U S H                                    *
               139F  ;                                                              *
               139F  ;***************************************************************
               139F  ;
               139F  FLUSH:
F730    31     139F        BRW     SEND_SUCCESS              ; Nop
```

```
                13A2  2539         .SBTTL  -      20 - REPLACE
                13A2  2540 ;*****************************************************
                13A2  2541 ;                                                    *
                13A2  2542 ;        # 20 -  R E P L A C E                       *
                13A2  2543 ;                                                    *
                13A2  2544 ;*****************************************************
                13A2  2545 ;
                13A2  2546 REPLACE:
      F584  31  13A2  2547         BRW     ERR_1                   ; Invalid opcode - Host Bad Not Supp
```

```
                   13A5  2549            .SBTTL  MAINTENANCE class commands
                   13A5  2550 ;
                   13A5  2551 MAINT:
      F581   31    13A5  2552            BRW     ERR_1
```

```
                        13A8         .SBTTL  NON-SEQUENTIAL BUFFER class commands
                        13A8     ;
                        13A8     ;       R0    scratch
                        13A8     ;       R1 = Ptr to message buffer
                        13A8     ;       R2 = low 3 bits of command
                        13A8     ;       R3 = Ptr to HQB
                        13A8     ;       R4 = Unit number
                        13A8     ;       R5 = Ptr to CDRP
                        13A8     ;
      54    78 A5   D0  13A8     NSEQB:  MOVL    CDRP$L_MST_UQB(R5),R4      ; Recover UQB pointer
      50    0E A4   3C  13AC             MOVZWL  UQB$W_STATUS(R4),R0        ; Pick up status
               03   13  13B0             BEQL    10$                        ; Success - ONLINE
             F71F   31  13B2             BRW     SEND_PKT                   ; Return error code
                        13B5
      50    30 A4   D0  13B5     10$:    MOVL    UQB$L_UCB(R4),R0           ; Ucb address
   06 64 A0    0E   E1  13B9             BBC     #UCB$V_MNTVERIP,UCB$L_STS(R0),15$
                        13BE                                                ; Branch if mount verification is not
                        13BE                                                ; progress
         50    04   D0  13BE             MOVL    #MSCP$K_ST_AVLBL,R0        ; mark unit as available
             F710   31  13C1             BRW     SEND_PKT                   ; return unit available
                        13C4
                    93  13C4     15$:    BITB    #<UQB$M_SEQ!-              ; Check if in SEQ mode
                        13C5                       UQB$M_RIP!-              ;  or in repair
                        13C5                       UQB$M_BLOCKED>,-         ;  or blocked
      2B A4    32       13C5                     UQB$B_FLAGS(R4)
               03   13  13C8             BEQL    20$                        ; If so ...
                        13CA             BLOCKT                             ;  then block the command
                        13CD
                        13CD     20$:    CASE    CDRP$B_OPCODE(R5),-        ; Dispatch on command subtype
                        13CD                     TYPE=B,LIMIT=#MSCP$K_OP_COMP-
                        13CD                     <COMP_HOST_DAT,-           ; 32-COMPARE HOST DATA
                        13CD                      READ,-                    ; 33-READ
                        13CD                      WRITE>                    ; 34-WRITE
             F54E   31  13D8             BRW     ERR_1
```

```
                   13DB  2589          .SBTTL  -       32 - COMPARE HOST DATA
                   13DB  2590  ;***********************************************************
                   13DB  2591  ;                                                          *
                   13DB  2592  ;          # 32 -  C O M P   H O S T   D A T               *
                   13DB  2593  ;                                                          *
                   13DB  2594  ;***********************************************************
                   13DB  2595  ;
                   13DB  2596  COMP_HOST_DAT:
  50    0A    B0   13DB  2597          MOVW    #IO$_WRITECHECK,R0      ; Set function code
      031E    31   13DE  2598          BRW     DO_WRT                  ;  and go do it
```

```
                        13E1         .SBTTL  -       33 - READ
                        13E1  ;**********************************************************
                        13E1  ;                                                         *
                        13E1  ;         # 33 -  R E A D                                 *
                        13E1  ;                                                         *
                        13E1  ;**********************************************************
                        13E1  ;
                        13E1  READ:
      50    0C     B0   13E1         MOVW    #IO$_READPBLK,R0           ; Set up for read
         03D8      30   13E4         BSBW    SETUP                      ; Common set up
           20      93   13E7         BITB    #1@CDRP$V_NOHSTBUF,-       ; Test flags
        FA A5           13E9                  CDRP$B_FLAGS(R5)
           07      12   13EB         BNEQ    READ_LOOP                  ; Skip if no host buffer
   D6 A5    01     B0   13ED         MOVW    #1,CDRP$W_STATE(R5)        ; Show it in mapping resource wait
                        13F1         MAP_IRP                            ; Map the buffer
                        13F4  ;
                        13F4  ; Read the data from the disk
                        13F4  ;
                        13F4  READ_LOOP:
   D6 A5    02     B0   13F4         MOVW    #2,CDRP$W_STATE(R5)        ; Set the state
         0662      30   13F8         BSBW    DO_DISK                    ; Set up fro disk transfer
           10      93   13FB         BITB    #1@CDRP$V_ABORT,-          ; Check if aborted
        FA A5           13FD                  CDRP$B_FLAGS(R5)
           3B      12   13FF         BNEQ    FINI                       ; Leave if so
        07 50      E8   1401         BLBS    R0,READ_CONT               ; Success
         0168      31   1404         BRW     READ_ERR                   ; Leave now for error
                        1407
                        1407  READ_INVALID_CDT:
                        1407         BUG_CHECK       MSCPSERV,FATAL     ; Stale CDT
                        140B  ;
                        140B  ; Send the data to the host buffer
                        140B  ;
                        140B  READ_CONT:
           20      93   140B         BITB    #1@CDRP$V_NOHSTBUF,-       ; Test flags
        FA A5           140D                  CDRP$B_FLAGS(R5)
           2B      12   140F         BNEQ    FINI                       ; Skip if no host buffer
   D6 A5    03     B0   1411         MOVW    #3,CDRP$W_STATE(R5)        ; Show it queued to SCS block transfer
                        1415  ; ??? Begin bug trap
   52    B0 A5     D0   1415         MOVL    CDRP$L_MSGBUFX(R5),R2      ; Get message buffer address for validity ch
   51    24 A5     D0   1419         MOVL    CDRP$L_CDT(R5),R1          ; Get CDT address
F8 A2    18 A1     D1   141D         CMPL    CDT$L_CONID(R1), -         ; Check validity of remote conid
                        1422                 SCS$L_DST_CONID(R2)
           E3      12   1422         BNEQ    READ_INVALID_CDT           ; Trying to use stale connection
                        1424  ; ??? End bug trap
        1C A5      D4   1424         CLRL    CDRP$L_MSG_BUF(R5)         ; Mark it not in use
                        1427         SEND_DATA                          ; Send to host buffer
        46 50      E9   1436         BLBC    R0,ABX_ER                  ; Aborted
        F4 A5      D7   1439         DECL    CDRP$L_CMD_STS(R5)         ; Set the command status
        B5 AF      9F   143C  FINI:  PUSHAB  READ_LOOP                  ; Loop again
                        143F
                        143F  ;
                        143F  ; TST_END: Update the accumulated byte count and compare to original to
                        143F  ; determine if another round is needed.  Also check for errors and translate
                        143F  ; VMS errors to MSCP errors.  Finally, clean up resources and branch off
                        143F  ; to return end packet with status.
                        143F  ;
                        143F         .ENABLE LSB
```

```
                           143F
         53    DA A5   D0  143F              MOVL    CDRP$L_IOST1+2(R5),R3    ; Pick up transfered bytes
               04      11  1443              BRB     10$                      ; Join common code
                           1445
                           1445   TST_END:
         53    D2 A5   D0  1445              MOVL    CDRP$L_BCNT(R5),R3       ; Pick up transfered bytes
  51     53    F7 8F   78  1449   10$:       ASHL    #-9,R3,R1                ; Make it blocks
       E0 A5   53      C0  144E              ADDL    R3,CDRP$L_ABCNT(R5)      ; Calc accumulated bytes xfer
               10      93  1452              BITB    #1@CDRP$V_ABORT,-        ; Check if aborted
            FA A5          1454                CDRP$B_FLAGS(R5)
               25      12  1456              BNEQ    ABO_ER                   ; Leave if so
         29    50      E9  1458              BLBC    R0,OUT_ER                ; Leave if error
               50      D4  145B              CLRL    R0                       ; Assume success
  52    E4 A5  E0 A5   C3  145D              SUBL3   CDRP$L_ABCNT(R5), -      ; Calc how much left to do
                           1463                CDRP$L_OBCNT(R5),R2
               47      15  1463              BLEQ    OUT                      ; None, we are finished
       D2 A5   52      D1  1465              CMPL    R2,CDRP$L_BCNT(R5)       ; If piece left less than BCNT
               08      18  1469              BGEQ    20$                      ; .. then ..
       D2 A5   52      D0  146B              MOVL    R2,CDRP$L_BCNT(R5)       ; Use it
       3C A5   52      D0  146F              MOVL    R2,CDRP$L_XCT_LEN(R5)    ; Ditto
  38 A5        E0 A5   D0  1473   20$:       MOVL    CDRP$L_ABCNT(R5), -      ; Set the remote byte offset
                           1478                CDRP$L_RBOFF(R5)               ;  to be accumlated xfer size
       48 A5   51      C0  1478              ADDL    R1,CDRP$L_LBN(R5)        ; Update LBN
               05          147C              RSB                              ; Loop again
                           147D
                           147D              .DISABLE LSB
                           147D   ;
                           147D   ; Command was aborted while on disk queue
                           147D   ;
               8E      D5  147D   ABO_ER: TSTL   (SP)+                        ; Clean stack
                           147F   ;
                           147F   ; Command was aborted during SEND/RECEIVE DATA to host
                           147F   ;
         50    02      D0  147F   ABX_ER: MOVL   #MSCP$K_ST_ABRTD,R0          ; Set status
               2A      11  1482              BRB     OUT1
                           1484   ;
                           1484   ; Command received a disk error - translate from VMS to MSCP status.
                           1484   ;
               8E      D5  1484   OUT_ER: TSTL   (SP)+                        ; Get rid of return address
               04      E0  1486              BBS     #CDRP$V_ABORT,-          ; Check if aborted
       F4 FA A5            1488                CDRP$B_FLAGS(R5),ABX_ER; Leave if so (has priority over errors)
  52     14F6'CF       3E  148B              MOVAW   ERR_TBL,R2               ; Pick up error table address
         51    82      3C  1490   10$:       MOVZWL  (R2)+,R1                 ; Grab VMS error entry
               0A      13  1493              BEQL    20$                      ; End of table, quit
         50    51      B1  1495              CMPW    R1,R0                    ; Try for a match
               05      13  1498              BEQL    20$                      ; Success
         52    04      C0  149A              ADDL    #4,R2                    ; Step to next entry
               F1      11  149D              BRB     10$                      ; Loop
                           149F
         50    82      3C  149F   20$:       MOVZWL  (R2)+,R0                 ; Pick up MSCP error
         51    62      32  14A2              CVTWL   (R2),R1                  ; Get possible routine
               07      13  14A5              BEQL    OUT1                     ; None
             6142      16  14A7              JSB     (R1)[R2]                 ; Invoke the routine
               02      11  14AA              BRB     OUT1                     ; Continue on
                           14AC   ;
                           14AC   ; Final clean up for transfer commands.  Set status, release resources.
                           14AC   ;
```

```
           8E  D5 14AC  OUT:      TSTL    (SP)+                        ; Get rid of return
 52     B0 A5  D0 14AE  OUT1:     MOVL    CDRP$L_MSGBUFX(R5),R2        ; Pick up old message buffer ptr
           09  13 14B2            BEQL    10$                          ; None, probably an abort
        E0 A5  D0 14B4            MOVL    CDRP$L_ABCNT(R5),-           ; Set byte count
        0C A2     14B7             MSCP$L_BYTE_CNT(R2)
  0A A2    50  B0 14B9            MOVW    R0,MSCP$W_STATUS(R2)         ; Store the status
  D6 A5    05  B0 14BD  10$:      MOVW    #5,CDRP$W_STATE(R5)          ; Set state
           20  93 14C1            BITB    #1@CDRP$V_NOHSTBUF,-
        FA A5     14C3             CDRP$B_FLAGS(R5)
           0B  12 14C5            BNEQ    20$                          ; Skip if no host buffer
 53     60 A5  D0 14C7            MOVL    CDRP$L_CDTX(R5),R3           ; Pick up the CDT address
 54     10 A3  D0 14CB            MOVL    CDT$L_PDT(R3),R4             ; Set up PDT address
                  14CF            UNMAP                                ; Get rid of mapping resources
 54     4C A5  D0 14D2  20$:      MOVL    CDRP$L_SHDW_LNK(R5),R4       ; Check possible IRP link
           0F  13 14D6            BEQL    40$                          ; None, continue on
      50   54  D0 14D8  30$:      MOVL    R4,R0                        ; Transfer IRP pointer
           0A  13 14DB            BEQL    40$                          ; Done
 54   00AC C4  D0 14DD            MOVL    IRP$L_SHDW_LNK(R4),R4        ; Transfer next IRP pointer
         F169  30 14E2            BSBW    DEALMEM                      ; Get rid of it
           F1  11 14E5            BRB     30$                          ; Loop for more IRP's
                  14E7
 52     B0 A5  D0 14E7  40$:      MOVL    CDRP$L_MSGBUFX(R5),R2        ; Pick up old message buffer ptr
           06  13 14EB            BEQL    50$                          ; None, probably an abort
        1C A2  D4 14ED            CLRL    MSCP$L_FRST_BAD(R2)          ; Clear unused field
         F5EB  31 14F0            BRW     SEND_PKT_STS                 ; Return the end packet
                  14F3
         F62B  31 14F3  50$:      BRW     CLEAN_UP                     ; Finish up
                  14F6
                  14F6  ERR_TBL:
                  14F6            ACTION  SS$_MEDOFL,MSCP$K_ST_AVLBL,ERR_AVAIL
                  14FC            ACTION  SS$_VOLINV,MSCP$K_ST_AVLBL,ERR_AVAIL
                  1502            ACTION  SS$_WRITLCK,MSCP$K_ST_WRTPR,ERR_WRITLCK
                  1508            ACTION  SS$_DATACHECK,MSCP$K_ST_COMP
                  150E            ACTION  SS$_CTRLERR,MSCP$K_ST_CNTLR
                  1514            ACTION  SS$_FORMAT,MSCP$K_ST_MFMTE
                  151A            ACTION  SS$_FORCEDERROR,MSCP$K_ST_DATA
                  1520            ACTION  SS$_PARITY,<<MSCP$K_ST_DATA>!<1*MSCP$K_ST_SBCOD>>
                  1526            ACTION  SS$_IVBUFLEN,MSCP$K_ST_HSTBF
                  152C            ACTION  SS$_SHACHASTA,MSCP$K_ST_SHST,ERR_SHACHA
                  1532            ACTION  SS$_TIMEOUT,<MSCP$K_ST_OFFLN!MSCP$K_SC_UNKNO>,ERR_OFFLINE
                  1538            ACTION  0,MSCP$K_ST_DRIVE            ; PATCH SPACE
                  153E            ACTION  0,MSCP$K_ST_DRIVE            ; End of table - default error
                  1544
                  1544  ERR_AVAIL:
 54     C4 A5  D0 1544            MOVL    CDRP$L_UQB(R5),R4            ; Pick up pointer
           04  B0 1548            MOVW    #MSCP$K_ST_AVLBL,-           ; Set new status in unit
        0E A4     154A             UQB$W_STATUS(R4)
               05 154C            RSB
                  154D
                  154D  ERR_OFFLINE:
 54     C4 A5  D0 154D            MOVL    CDRP$L_UQB(R5),R4            ; Pick up pointer
           03  B0 1551            MOVW    #MSCP$K_ST_OFFLN,-           ; Set new status in unit
        0E A4     1553             UQB$W_STATUS(R4)
               05 1555            RSB
                  1556
                  1556  ERR_WRITLCK:
 51     C4 A5  D0 1556            MOVL    CDRP$L_UQB(R5),R1            ; Pick up pointer
```

```
00 12 A1    0D   E2  155A           BBSS    #MSCP$V_UF_WRTPH, -
                     155F                   UQB$W_UNT_FLGS(R1),20$  ; Set in unit flags
                 05  155F  20$:     RSB
                     1560
                     1560  ERR_SHACHA:
            8E   D5  1560           TSTL    (SP)+                    ; Remove return address
      54 C4 A5   D0  1562           MOVL    CDRP$L_UQB(R5),R4        ; Pick up pointer
          F7DE   30  1566           BSBW    AVAIL_ALL                ; Kick unit out of shadow set
      50    0C   D0  1569           MOVL    #MSCP$K_ST_SHST,R0       ; Restore status
          FF3F   31  156C           BRW     OUT1                     ; Continue
                     156F
                     156F  ;
                     156F  ; Error recovery routine for shadow reads.  First, verify that this is
                     156F  ; really a shadow read.  If so, attempt the read repair.  There are only
                     156F  ; two cases to consider:
                     156F  ;
                     156F  ;        1) R0 = SS$_NORMAL: Read partially succeded, up to a parity or
                     156F  ;                forcederror.  Continue the SEND_DATA part of the loop
                     156F  ;                and retry the read at the failed block.  The read will
                     156F  ;                either succede or fail back here with an error.
                     156F  ;
                     156F  ;        2) R0 = failure: The repair/retry logic failed.  Terminate the
                     156F  ;                read and report the error to host.
                     156F  ;
                     156F  READ_ERR:
      51 C4 A5   D0  156F           MOVL    CDRP$L_UQB(R5),R1        ; Pick up UQB pointer
      52 78 A5   D0  1573           MOVL    CDRP$L_MST_UQB(R5),R2    ; Pick up master UQB pointer
            0E   E1  1577           BBC     #MSCP$V_UF_SSMEM,-       ; If this unit is shadow member
   30 12 A1          1579                   UQB$W_UNT_FLGS(R1),20$ ;  and
      52    51   D1  157C           CMPL    R1,R2                    ; If IO directed to shadow master
            2B   13  157F           BEQL    20$                      ;  then
50 A5   FE6F CF  9E  1581           MOVAB   READ_LOOP,CDRP$L_FPCX(R5) ; Where to send blocked reads
            26   10  1587           BSBB    READ_REPAIR              ; Attempt read repair
      20    50   E9  1589           BLBC    R0,20$                   ; Failed to find good block
51  DA A5  F7 8F 78  158C           ASHL    #-9,CDRP$L_IOST1+2(R5),R1 ; Find how much was transfered
3C A5   51  09   78  1592           ASHL    #9,R1,CDRP$L_XCT_LEN(R5); Back to bytes
      B4 A5      D0  1597           MOVL    CDRP$L_BUFLEN(R5),-      ; Restore the old bytecnt
      D2 A5          159A                   CDRP$L_BCNT(R5)
            05   E1  159C           BBC     #CDRP$V_NOHSTBUF,-       ; If this is a passthru,
   08 FA A5          159E                   CDRP$B_FLAGS(R5),10$
   51   51  02   78  15A1           ASHL    #2,R1,R1                 ; Make it number of longwords
      CC A5 51   C0  15A5           ADDL    R1,CDRP$L_SVAPTE(R5)     ; Update PTE pointer
          FE5F   31  15A9  10$:     BRW     READ_CONT                ; Succeded in repair, continue SEND
                     15AC
          FE8D   31  15AC  20$:     BRW     FINI                     ; Leave for good
```

```
                  15AF          .SBTTL  -                       READ REPAIR routine
                  15AF
                  15AF  ;++
                  15AF  ;
                  15AF  ; READ_REPAIR
                  15AF  ;
                  15AF  ; Read error recovery for bad blocks. This operation has two phases:
                  15AF  ;
                  15AF  ;     Phase 1:    If error is not SS$_PARITY or SS$_FORCEDERROR, then
                  15AF  ;                 leave immediately with R0 = SS$_SHACHASTA.  Next we check
                  15AF  ;                 the bytecnt.  If at least one block has been successfully
                  15AF  ;                 read, then we leave with R0 = SS$_NORMAL after updating
                  15AF  ;                 transfer parameters to reflect a transfer of up to but not
                  15AF  ;                 including the bad block.  The next read will then cause us
                  15AF  ;                 to return here for phase 2.  Otherwise we drop immediately
                  15AF  ;                 into phase 2.
                  15AF  ;
                  15AF  ;     Phase 2:    We enter with an immediate error in the first block.  We
                  15AF  ;                 can return with four cases:
                  15AF  ;
                  15AF  ;         1) R0 = SS$_NORMAL: The block in error on all source units has been
                  15AF  ;                 successfully repaired if a FORCEDERROR or a good block on an
                  15AF  ;                 alternate unit has been found for PARITY error.  We leave
                  15AF  ;                 with transfer parameters indicating a good one block read.
                  15AF  ;
                  15AF  ;         2) R0 = SS$_SHACHASTA: An unrepairable, fatal error occurred on one
                  15AF  ;                 of the source units or the target unit.  The calling routine
                  15AF  ;                 must fail the offending unit out of the shadow set.
                  15AF  ;
                  15AF  ;         3) R0 = SS$_PARITY: We have a parity error in the same LBN of all
                  15AF  ;                 the source units.  No repair is possible. The shadow set
                  15AF  ;                 remains intact.
                  15AF  ;
                  15AF  ;         4) R0 = SS$_FORCEDERROR: As case 3 above, only a forcederror.
                  15AF  ;
                  15AF  ; Any error that occurs while a repair is pending is put on the blocked queue
                  15AF  ; and will be restarted from the read when the repair operation is finished.
                  15AF  ; This routine is non-abortable.
                  15AF  ;
                  15AF  ; Inputs:
                  15AF  ;         R0  =   Disk failure code
                  15AF  ;         R1 -->  Failing UQB
                  15AF  ;         R2 -->  Master UQB
                  15AF  ;         R3/R4   saved
                  15AF  ;         R5 -->  CDRP
                  15AF  ;         CDRP$L_UQB  --> Failing unit UQB
                  15AF  ;         CDRP$L_FPCX --> Where to restart blocked reads
                  15AF  ;
                  15AF  ; Outputs:
                  15AF  ;         R0  =   Status code
                  15AF  ;         CDRP$L_IOST1 = upated
                  15AF  ;
                  15AF  ;--
                  15AF
                  15AF  READ_REPAIR:
51   C4 A5   D0   15AF          MOVL    CDRP$L_UQB(R5),R1         ; Pick up UQB pointer
52   78 A5   D0   15B3          MOVL    CDRP$L_MST_UQB(R5),R2     ; Pick up master UQB pointer
```

```
        2144 8F    50   B1 15B7       CMPW    R0,#SS$_FORCEDERROR     ; Is this a repairable error?
                   10   13 15BC       BEQL    10$                     ; Yes, go off to repair logic
        01F4 8F    50   B1 15BE       CMPW    R0,#SS$_PARITY          ; Is it parity error
                   09   13 15C3       BEQL    10$                     ; Yes, go off to repair logic
              DA A5     D4 15C5       CLRL    CDRP$L_IOST1+2(R5)      ; No bytes transfered to host
   50    2284 8F        3C 15C8       MOVZWL  #SS$_SHACHASTA,R0       ; Set fatal error
                        05 15CD       RSB                             ; Leave
                           15CE ;
                           15CE ; Set RIP mode (rundown) for the shadow unit.  If this is not the first
                           15CE ; error for this unit, hold it on the blocked queue until the first
                           15CE ; repair is finished.  ALL REPAIR OPERATIONS MUST BE SEQUENTIAL!!!
                           15CE ; When the repair is finished, these operations will be restarted.
                           15CE ;
           5C A5 8ED0      15CE 10$:  POPL    CDRP$L_SAVD_RTX(R5)     ; Save return address
                   05   E1 15D2       BBC     #UQB$V_RIP,-            ; If already a RIP
         1C 2B A2          15D4               UQB$B_FLAGS(R2),20$
            54     52   D0 15D7       MOVL    R2,R4                   ; Transfer to proper register
      50     E8 A5      0F 15DA       REMQUE  CDRP$L_UQB_FL(R5),R0    ; Remove from execute queue
             50 A4      B7 15DE       DECW    UQB$W_NUM_QUE(R4)       ; Account for it
                           15E1       BLOCKT                          ; Block until first RIP over
             E8 A5      0E 15E4       INSQUE  CDRP$L_UQB_FL(R5),-     ; Insert packet on unit que
             64 B4         15E7               @UQB$L_CDRP_BL(R4)
             50 A4      B6 15E9       INCW    UQB$W_NUM_QUE(R4)       ; Account for it
      54     58 A5      D0 15EC       MOVL    CDRP$L_FR4X(R5),R4      ; Restore R4 (PDT)
             50 B5      17 15F0       JMP     @CDRP$L_FPCX(R5)        ; Attempt the read again
                           15F3 ;
                           15F3 ; First repairable error on this volume.  Check for any I/O.  If so
                           15F3 ; block the RIP CDRP until all other I/O done, saving current state.
                           15F3 ;
             C0 A5      B0 15F3 20$:  MOVW    CDRP$W_FUNC(R5),-       ; Save old function code
             56 A5         15F6               CDRP$L_FR3X+2(R5)
 50    DA A5   F7 8F    78 15F8       ASHL    #-9,CDRP$L_IOST1+2(R5),R0; Number of blocks transfered
                   0D   13 15FE       BEQL    30$                     ; None, we are in real repair
    D2 A5     50   09   78 1600       ASHL    #9,R0,CDRP$L_BCNT(R5)   ; Update transfer parameter
             D2 A5      D0 1605       MOVL    CDRP$L_BCNT(R5),-       ; Also update status
             DA A5         1608               CDRP$L_IOST1+2(R5)
               00C6     31 160A       BRW     140$                    ; Leave now, first part done
                           160D
                   20   88 160D 30$:  BISB    #UQB$M_RIP,-            ; Set the repair in progress flag
             2B A2         160F               UQB$B_FLAGS(R2)         ;  on shadow master unit
                   20   88 1611       BISB    #UQB$M_RIP,-            ; Set the repair in progress flag
             2B A1         1613               UQB$B_FLAGS(R1)         ;  on physical unit
             D8 A5      B0 1615       MOVW    CDRP$L_IOST1(R5),-      ; Save old status
             54 A5         1618               CDRP$L_FR3X(R5)
      01     50 A2      B1 161A       CMPW    UQB$W_NUM_QUE(R2),#1    ; Check the shadow master for IO
                   1D   1B 161E       BLEQU   40$                     ; None, all okay
            54     52   D0 1620       MOVL    R2,R4                   ; Transfer to proper register
      50     E8 A5      0F 1623       REMQUE  CDRP$L_UQB_FL(R5),R0    ; Remove from execute queue
             50 A4      B7 1627       DECW    UQB$W_NUM_QUE(R4)       ; Account for it
                           162A       BLOCKT                          ; Block until all I/O rundown
             E8 A5      0E 162D       INSQUE  CDRP$L_UQB_FL(R5),-     ; Insert packet on unit que
             64 B4         1630               @UQB$L_CDRP_BL(R4)
             50 A4      B6 1632       INCW    UQB$W_NUM_QUE(R4)       ; Account for it
      51     C4 A5      D0 1635       MOVL    CDRP$L_UQB(R5),R1       ; Pick up failed UQB pointer
      52     78 A5      D0 1639       MOVL    CDRP$L_MST_UQB(R5),R2   ; Pick up master UQB pointer
    D2 A5     0200 8F   3C 163D 40$:  MOVZWL  #512,CDRP$L_BCNT(R5)    ; Only do one block
                           1643 ;     MOVL    R1,CDRP$L_STRT_UQB(R5)  ; Set a end condition
```

```
                            1643  ;
                            1643  ; Attempt to find a good copy of the block, reading all units if necesary.
                            1643  ; Keep a list of units that are unreadable.  If no units are readable, we
                            1643  ; will return the IO with the FORCEDERROR status.  If any unit has an
                            1643  ; unrepairable error, we will fail out with Shadow Set Status has changed.
                            1643  ;
         50    54 A5    3C  1643  50$:    MOVZWL  CDRP$L_FR3X(R5),R0          ; Pick up true status code
                 03EA   30  1647          BSBW    SELECT_NEXT_UNIT            ; Get the next unit
                   56   13  164A          BEQL    90$                         ; None, left error out
                        B0  164C          MOVW    #<IO$_READPBLK!-            ; Set for READ/COMPARE
                            164D                   IO$M_DATACHECK>,-
      C0 A5   400C 8F       164D                   CDRP$W_FUNC(R5)
         D6 A5     02   B0  1652          MOVW    #2,CDRP$W_STATE(R5)         ; Set the state
                 0404   30  1656          BSBW    DO_DISK                     ; Do the read
         51     C4 A5   D0  1659          MOVL    CDRP$L_UQB(R5),R1           ; Pick up UQB pointer
                0C 50   E8  165D          BLBS    R0,60$                      ; Success now go to write
         54 A5     50   B1  1660          CMPW    R0,CDRP$L_FR3X(R5)          ; Was the error repairable?
                   37   12  1664          BNEQ    80$                         ; No, leave with failure
                   20   88  1666          BISB    #UQB$M_RIP,-                ; Set the repair in progress flag
                2B A1       1668                   UQB$B_FLAGS(R1)
                   D7   11  166A          BRB     50$                         ; Try for another unit
                            166C  ;
                            166C  ; Read phase succeded in locating a good copy of the block.  Now run thru
                            166C  ; the list of units with a bad copy and repair them.
                            166C  ;
         74 A5     51   D0  166C  60$:    MOVL    R1,CDRP$L_STRT_UQB(R5)      ; Set end condition & save good unit UQB
                        B0  1670          MOVW    #<IO$_WRITEPBLK!-           ; Now change to write
                            1671                   IO$M_DATACHECK>,-
      C0 A5   400B 8F       1671                   CDRP$W_FUNC(R5)
                 03BB   30  1676  70$:    BSBW    SELECT_NEXT_UNIT            ; Get the next unit
                   41   13  1679          BEQL    130$                        ; None left, leave
                   05   E5  167B          BBCC    #UQB$V_RIP,-                ; Check for units needing repair
             F6 2B A1       167D                   UQB$B_FLAGS(R1),70$
                54 A5   B1  1680          CMPW    CDRP$L_FR3X(R5),-           ; If parity error,
              01F4 8F       1683                   #SS$_PARITY                ;  then skip write
                   EE   13  1686          BEQL    70$                         ;  since it cannot be written
         D6 A5     04   B0  1688          MOVW    #4,CDRP$W_STATE(R5)         ; Set the state
                 03CE   30  168C          BSBW    DO_DISK                     ; Do the write
         51     C4 A5   D0  168F          MOVL    CDRP$L_UQB(R5),R1           ; Pick up UQB pointer
                E0 50   E8  1693          BLBS    R0,70$                      ; Success, look for next unit
      025C 8F      50   B1  1696          CMPW    R0,#SS$_WRITLCK             ; Was the error writelock?
                   D9   13  169B          BEQL    70$                         ; If so, ignore it
                            169D  ;
                            169D  ; A hard, unrepairable error has occurred.  Clean up any status and
                            169D  ; then fail this unit out of shadow set and report status to host.
                            169D  ;
         50   2284 8F   B0  169D  80$:    MOVW    #SS$_SHACHASTA,R0           ; Set the fatal error code
         51     74 A5   D0  16A2  90$:    MOVL    CDRP$L_STRT_UQB(R5),R1      ; Get starting point
                   09   11  16A6          BRB     110$                        ; Do it
                            16A8
         51        61   D0  16A8  100$:   MOVL    UQB$L_SHDW_FL(R1),R1        ; Advance to next unit
         74 A5     51   D1  16AB          CMPL    R1,CDRP$L_STRT_UQB(R5)      ; End?
                   06   13  16AF          BEQL    120$                        ; Yes
                   20   8A  16B1  110$:   BICB    #UQB$M_RIP,-                ; Clear flags
                2B A1       16B3                   UQB$B_FLAGS(R1)
                   F1   11  16B5          BRB     100$                        ; Loop for more
                            16B7
```

```
          DA A5   D4  16B7  120$:   CLRL     CDRP$L_IOST1+2(R5)        ; Show nothing transfered
             1A   11  16BA          BRB      150$                      ; Leave
                      16BC  ;
                      16BC  ; Repair is finished.  Clean up, restore original parameters and
                      16BC  ; restart the read from the beginning.
                      16BC  ;
             20   8A  16BC  130$:   BICB     #UQB$M_RIP,-              ; Clear status bit
          2B A1       16BE                     UQB$B_FLAGS(R1)
       51 74 A5   D0  16C0          MOVL     CDRP$L_STRT_UQB(R5),R1    ; Recover original UQB
       C4 A5 51   D0  16C4          MOVL     R1,CDRP$L_UQB(R5)         ; Restore it
          30 A1   D0  16C8          MOVL     UQB$L_UCB(R1),-           ; Transfer UCB pointer
          BC A5       16CB                     CDRP$L_UCB(R5)
DA A5   0200 8F   3C  16CD          MOVZWL   #512,CDRP$L_IOST1+2(R5)   ; Show one block read
       50     01  3C  16D3  140$:   MOVZWL   #SS$_NORMAL,R0            ; Set success status
          56 A5   B0  16D6  150$:   MOVW     CDRP$L_FR3X+2(R5),-       ; Restore function code
          C0 A5       16D9                     CDRP$W_FUNC(R5)
       52 78 A5   D0  16DB          MOVL     CDRP$L_MST_UQB(R5),R2     ; Grab shadow master UQB
             20   8A  16DF          BICB     #UQB$M_RIP,-              ; Clear status bit
          2B A2       16E1                     UQB$B_FLAGS(R2)
       54 60 A5   D0  16E3          MOVL     CDRP$L_CDTX(R5),R4        ; Pick up CDT address
       54 10 A4   D0  16E7          MOVL     CDT$L_PDT(R4),R4          ; Pick up PDT address
          5C B5   17  16EB          JMP      @CDRP$L_SAVD_RTX(R5)      ; Return
```

```
                          16EE          .SBTTL  -       34 - WRITE
                          16EE  ;*********************************************************
                          16EE  ;                                                        *
                          16EE  ;           # 34 -  W R I T E                            *
                          16EE  ;                                                        *
                          16EE  ;*********************************************************
                          16EE  ;
                          16EE          .ENABLE LSB
                          16EE
                          16EE  WRITE:
                      B3  16EE          BITW    #<MSCP$M_UF_WRTPS!-
                          16EF                    MSCP$M_UF_WRTPH>,-
       12 A4  3000 8F     16EF                  UQB$W_UNT_FLGS(R4)      ; If WRITE_PROTECT
                06    13  16F4          BEQL    10$
          50    06    D0  16F6          MOVL    #MSCP$K_ST_WRTPR, R0    ; Set error status
              F3D8    31  16F9          BRW     SEND_PKT                ; ... and exit
                          16FC
          50    0B    B0  16FC  10$:    MOVW    #IO$_WRITEPBLK,R0       ; Set up for write
              00BD    30  16FF  DO_WRT: BSBW    SETUP
                20    93  1702          BITB    #1@CDRP$V_NOHSTBUF,-    ; Test flags
             FA A5        1704                  CDRP$B_FLAGS(R5)
                07    12  1706          BNEQ    20$                     ; Skip if no host buffer
       D6 A5    01    B0  1708          MOVW    #1,CDRP$W_STATE(R5)     ; Mapping resource wait state
                          170C          MAP_IRP                         ; Map the data
                          170F  ;
                          170F  ; Request data from host buffer
                          170F  ;
       D6 A5    03    B0  170F  20$:    MOVW    #3,CDRP$W_STATE(R5)     ; Bump the state
                20    93  1713          BITB    #1@CDRP$V_NOHSTBUF,-    ; Test flags
             FA A5        1715                  CDRP$B_FLAGS(R5)
                2B    12  1717          BNEQ    30$                     ; Skip if no host buffer
                          1719  ; ??? Begin bug trap
          52 B0 A5    D0  1719          MOVL    CDRP$L_MSGBUFX(R5),R2   ; Get message buffer address for validity ch
          51 24 A5    D0  171D          MOVL    CDRP$L_CDT(R5),R1       ; Get CDT address
    F8 A2    18 A1    D1  1721          CMPL    CDT$L_CONID(R1), -      ; Check validity of remote conid
                          1726                  SCS$L_DST_CONID(R2)
                18    12  1726          BNEQ    25$                     ; Trying to use stale connection
                          1728  ; ??? End bug trap
             1C A5    D4  1728          CLRL    CDRP$L_MSG_BUF(R5)      ; Mark it not in use
                          172B          REQUEST_DATA                    ; Get the data from host
             07 50    E8  173A          BLBS    R0,30$                  ; Success
              FD3F    31  173D          BRW     ABX_ER                  ; Aborted
                          1740
                          1740  25$:    BUG_CHECK       MSCPSERV,FATAL  ; Trying to use stale CDT
                          1744  ;
                          1744  ; Write the data to the disk
                          1744  ;
             F4 A5    D7  1744  30$:    DECL    CDRP$L_CMD_STS(R5)      ; Set the command status
             D6 A5    B6  1747          INCW    CDRP$W_STATE(R5)        ; Set state
          53 4C A5    D0  174A          MOVL    CDRP$L_SHDW_LNK(R5),R3  ; Pick up the first IRP
                16    13  174E          BEQL    60$                     ; No shadowing, skip
       D8 A5    01    D0  1750          MOVL    #1,CDRP$L_IOST1(R5)     ; Set default success
           00AC C3    DD  1754  40$:    PUSHL   IRP$L_SHDW_LNK(R3)      ; Save pointer to next IRP
             7E 54    7D  1758          MOVQ    R4,-(SP)                ; Save registers
                12    10  175B          BSBB    70$                     ; Do it
          54    8E    7D  175D          MOVQ    (SP)+,R4                ; Restore registers
             53 8ED0      1760          POPL    R3                      ; Restore pointer
```

```
                  EF   12  1763         BNEQ    40$                          ;  and loop
                  05       1765  50$:   RSB                                  ; None left, leave now
                           1766  ;
                           1766  ; Non-shadowed writes
                           1766  ;
                02F4   30  1766  60$:   BSBW    DO_DISK                      ; Do the disk transfer
               A3 AF   9F  1769         PUSHAB  20$                          ; Loop return
                FCD6   31  176C         BRW     TST_END                      ; Check if finished
                           176F  ;
                           176F  ; Shadowed writes
                           176F  ;
               C2 A5   96  176F  70$:   INCB    CDRP$B_SHDW_CNT(R5)          ; Count the number of copies
                02EC   30  1772  80$:   BSBW    DO_DISKA                     ; Do the disk transfer
               0F 50   E9  1775         BLBC    R0,WRITE_ERR                 ; Error
               C2 A5   97  1778  90$:   DECB    CDRP$B_SHDW_CNT(R5)          ; Drop the count
                  E8   12  177B         BNEQ    50$                          ; Not finished, go back for more
               8F AF   9F  177D         PUSHAB  20$                          ; Loop return
          50   D8 A5   D0  1780         MOVL    CDRP$L_IOST1(R5),R0          ; Pick up composite status
                FCBE   31  1784         BRW     TST_END                      ; Check if finished
                           1787  ;
                           1787  ; Shadowed write error recovery.
                           1787  ;
                           1787  WRITE_ERR:
                  04   E0  1787         BBS     #CDRP$V_ABORT,-              ; Leave if abort
            EC FA A5       1789                 CDRP$B_FLAGS(R5),90$
        01F4 8F   50   B1  178C         CMPW    R0,#SS$_PARITY               ; If not parity
                  08   13  1791         BEQL    110$                         ;  then
             2284 8F   3C  1793         MOVZWL  #SS$_SHACHASTA,-             ; Update composite status
               D8 A5       1797                 CDRP$L_IOST1(R5)
                  DD   11  1799         BRB     90$                          ; Continue
                           179B
             50   01   D0  179B  110$:  MOVL    #1,R0                        ; Set for success
  51    3A A3   F7 8F  78  179E         ASHL    #-9,IRP$L_IOST1+2(R3),R1; Number of blocks transfered
                  51   D6  17A4         INCL    R1                           ; Skip this block
        00A8 C3   51   C0  17A6         ADDL    R1,IRP$L_LBN(R3)             ; Show it
          51  51  02   78  17AB         ASHL    #2,R1,R1                     ; Number of bytes of svapte
           2C A3  51   C0  17AF         ADDL    R1,IRP$L_SVAPTE(R3)          ; Update buffer address
          51  51  07   78  17B3         ASHL    #7,R1,R1                     ; Number of bytes "transfered"
           32 A3  51   C2  17B7         SUBL    R1,IRP$L_BCNT(R3)            ; Left to go
                  B5   12  17BB         BNEQ    80$                          ; Redo the IO
                  B9   11  17BD         BRB     90$                          ; Nothing left, fake success
                           17BF
                           17BF         .DISABLE LSB
```

```
                          17BF          .SBTTL  SETUP - Setup the CDRP for transfers
                          17BF
                          17BF     ;++
                          17BF     ;
                          17BF     ; SETUP - Set up the all the entries for both disk and CI transfers
                          17BF     ;
                          17BF     ; Inputs:
                          17BF     ;        R0  =    Function code
                          17BF     ;        R2 -->   MSCP packet
                          17BF     ;        R4 -->   UQB
                          17BF     ;        R5 -->   CDRP
                          17BF     ;
                          17BF     ; Output:
                          17BF     ;        R4 -->   PDT
                          17BF     ;        R5 -->   CDRP
                          17BF     ;
                          17BF     ;--
                          17BF
       52    B0 A5   D0   17BF     SETUP:  MOVL    CDRP$L_MSGBUFX(R5),R2     ; Refresh R2 => MSCP message.
       C4 A5    54   D0   17C3             MOVL    R4,CDRP$L_UQB(R5)         ; Save the UQB pointer
       78 A5    54   D0   17C7             MOVL    R4,CDRP$L_MST_UQB(R5)     ; Save master UQB pointer
    64 B4    E8 A5   0E   17CB             INSQUE  CDRP$L_UQB_FL(R5), -      ; Insert this packet on unit que
                          17D0                     @UQB$L_CDRP_BL(R4)
          50 A4      B6   17D0             INCW    UQB$W_NUM_QUE(R4)         ; Bump number of packet queue to unit
          50 A4      B1   17D3             CMPW    UQB$W_NUM_QUE(R4),-       ; Is this number <= max?
          52 A4           17D6                     UQB$W_MAX_QUE(R4)
             05      1B   17D8             BLEQU   10$                       ; Yes, skip
          50 A4      B0   17DA             MOVW    UQB$W_NUM_QUE(R4),-       ; No, set new max
          52 A4           17DD                     UQB$W_MAX_QUE(R4)
          D6 A5      B4   17DF     10$:    CLRW    CDRP$W_STATE(R5)          ; Set state
        4000 8F      B3   17E2             BITW    #1@MSCP$V_MD_COMP,-
          0A A2           17E6              MSCP$W_MODIFIER(R2)
             05      13   17E8             BEQL    20$                       ; If COMPARE modifier
    50  4000 8F      A8   17EA             BISW    #IO$M_DATACHECK,R0        ; ... set DATACHECK modifier
        0100 8F      B3   17EF     20$:    BITW    #1@MSCP$V_MD_SEREC,-
          0A A2           17F3              MSCP$W_MODIFIER(R2)
             05      13   17F5             BEQL    30$                       ; If SUPPRESS ERROR CORRECT modifier
    50  8000 8F      A8   17F7             BISW    #IO$M_INHRETRY,R0         ; ... set INHIBIT RETRY modifier
       C0 A5    50   B0   17FC     30$:    MOVW    R0,CDRP$W_FUNC(R5)        ; Set completed code in packet
       51    0C A2   D0   1800             MOVL    MSCP$L_BYTE_CNT(R2),R1    ; Pick up size of request
             5A      13   1804             BEQL    60$                       ; Special case zero length
          30 A4      D0   1806             MOVL    UQB$L_UCB(R4),-           ; Transfer UCB address
          BC A5           1809              CDRP$L_UCB(R5)                   ;   to UCB addr in IRP
          E0 A5      D4   180B             CLRL    CDRP$L_ABCNT(R5)          ; Clear accumulated byte count
       E4 A5    51   D0   180E             MOVL    R1,CDRP$L_OBCNT(R5)       ; Set original byte count
          53    51   D0   1812             MOVL    R1,R3                     ; Copy to another register
             53      D7   1815             DECL    R3                        ; Round byte count down
    53    53 F7 8F   78   1817             ASHL    #-VA$S_BYTE,R3,R3         ; Convert to block count
          50    53   D0   181C             MOVL    R3,R0                     ; Block count
 0000007F 8F    50   D1   181F             CMPL    R0,#127                   ; If less than, skip on
             04      1B   1826             BLEQU   40$
    50    7F 8F      9A   1828             MOVZBL  #127,R0                   ; Else make 127
     EA83 CF40       D6   182C     40$:    INCL    BLK_COUNT+4[R0]           ; Keep statistics
    50    1C A2      D0   1831             MOVL    MSCP$L_LBN(R2),R0         ; Transfer the logical block number
          53    50   C0   1835             ADDL    R0,R3                     ; Calculate ending block number
       48 A4    53   D1   1838             CMPL    R3,UQB$L_MAXBLOCK(R4)     ; Legal block number
             18      1A   183C             BGTRU   50$                       ; If GTRU no
```

```
      4C A4    50    C1  183E            ADDL3   R0,UQB$L_START_LBN(R4),-
            48 A5        1842                    CDRP$L_LBN(R5)                ; Set 'real' starting LBN
               20    93  1844            BITB    #1@CDRP$V_NOHSTBUF,-          ; Is this a no host buffer IO?
            FA A5        1846                    CDRP$B_FLAGS(R5)
               1F    13  1848            BEQL    70$                           ; No, continue with setup
      B4 A5    51    D0  184A            MOVL    R1,CDRP$L_BUFLEN(R5)          ; Yes, save length
         10 A2       7D  184E            MOVQ    MSCP$B_BUFFER(R2),-           ; Xfer SVAPTE and BOFF
            CC A5        1851                    CDRP$L_SVAPTE(R5)
             00CD    31  1853            BRW     130$                          ; And finish
                         1856
                         1856     ;
                         1856     ; Out of line error routine
                         1856     ;
                         1856
               8E    D5  1856     50$:   TSTL    (SP)+                         ; Get rid of return address
    50    0C01 8F    3C  1858            MOVZWL  #MSCP$K_ST_ICMD -
                         185D                    !<MSCP$L_BYTE_CNT@8>,R0 ; Set error status
             F274    31  185D            BRW     SEND_PKT                      ; Leave in disgrace
                         1860
          EA4C CF    D6  1860     60$:   INCL    BLK_COUNT                     ; Count zero block transfer
               8E    D5  1864            TSTL    (SP)+                         ; Get rid of return address
             F269    31  1866            BRW     SEND_SUCCESS
                         1869
                         1869     ;
                         1869     ; Not a passthru so grab an internal buffer, and map it
                         1869     ;
                         1869
    34 A5    10 A2   DE  1869     70$:   MOVAL   MSCP$B_BUFFER(R2), -          ; Calculate offset of remote buf hdl
                         186E                    CDRP$L_RBUFH_AD(R5)
            38 A5    D4  186E            CLRL    CDRP$L_RBOFF(R5)              ; Clear remote offset
      E7E6 CF    51  D1  1871            CMPL    R1,FRACTION                   ; Greater than the fraction?
               05    1B  1876            BLEQU   80$                           ; No, continue
      51   E7E0 CF   D0  1878            MOVL    FRACTION,R1                   ; Yes, use the fraction
      51   01FF C1   9E  187D     80$:   MOVAB   511(R1),R1                    ; Round up
  51    000001FF 8F  CA  1882            BICL2   #511,R1
               21    13  1889            BEQL    90$                           ; Less than a page
      53   E7C1 CF   DE  188B            MOVAL   BUFF_HEAD,R3                  ; Set up
               51    DD  1890            PUSHL   R1                            ; Save request size
        00000000'9F  16  1892            JSB     @#EXE$ALLOCATE                ; Grab the block
            42 50    E8  1898            BLBS    R0,110$                       ; Space
            51 8ED0      189B            POPL    R1                            ; Retrieve request size
      E7B5 CF    51  D1  189E            CMPL    R1,SMALL                      ; Less than smallest fragment?
               07    1B  18A3            BLEQU   90$                           ; Yes, wait for pool
  51    51    FF 8F  78  18A5            ASHL    #-1,R1,R1                     ; Try for half
               D1    11  18AA            BRB     80$                           ; And go again
                         18AC
         18 A5 8ED0      18AC     90$:   POPL    CDRP$L_SAVD_RTN(R5)           ; Store return address
               65    0E  18B0            INSQUE  CDRP$L_FQFL(R5),-             ; Insert on the waiting que
          E7C7 DF        18B2                    @MEM_WAIT+4
          E7A7 CF    B6  18B5            INCW    NUM_QUE                       ; Bump memory wait counter
  E7A2 CF    E7A3 CF B1  18B9            CMPW    NUM_QUE,MAX_QUE               ; Maximize with
               07    1B  18C0            BLEQU   100$                          ;  current
  E799 CF    E79A CF B0  18C2            MOVW    NUM_QUE,MAX_QUE               ;  maximum
                         18C9     100$:  WFINT                                 ; Stall until resource avail
                         18CC                                                  ; (Will be activated at CLEAN_UP)
          E790 CF    B7  18CC            DECW    NUM_QUE                       ; Decr memory wait counter
            18 A5    DD  18D0            PUSHL   CDRP$L_SAVD_RTN(R5)           ; Set return address
```

```
          51    E4 A5   D0  18D3           MOVL    CDRP$L_OBCNT(R5),R1       ; Pick up needed length
                   A4   11  18D7           BRB     80$                       ; Try again
                            18D9
                            18D9  105$:    BUG_CHECK        MSCPSERV,FATAL   ; Allocated region out of bounds
                            18DD
                5E  04  C0  18DD  110$:    ADDL2   #4,SP                     ; Flush request size
                            18E0  ; ??? BUG TRAP
          EBEB CF   52  D1  18E0           CMPL    R2,BUFF_ADDR              ; Does address precede allocation region
                    F2  1F  18E5           BLSSU   105$                      ; Branch if yes
            50  51  52  C1  18E7           ADDL3   R2,R1,R0                  ; End of region+1
          EBE4 CF   50  D1  18EB           CMPL    R0,BUFF_END               ; Is it beyond+1
                    E7  1A  18F0           BGTRU   105$
                            18F2  ; ??? END BUG TRAP
                 D6 A5  B6  18F2           INCW    CDRP$W_STATE(R5)          ; Bump the state
             B4 A5  51  7D  18F5           MOVQ    R1,CDRP$L_BUFLEN(R5)      ; Save length and address
             E4 A5  51  D1  18F9           CMPL    R1,CDRP$L_OBCNT(R5)       ; If the length has been rounded
                    04  15  18FD           BLEQ    120$                      ;  then
          51    E4 A5   D0  18FF           MOVL    CDRP$L_OBCNT(R5),R1       ; Restore the original byte count
             3C A5  51  D0  1903  120$:    MOVL    R1,CDRP$L_XCT_LEN(R5)     ; Set length in place for S/R DATA
             FA A5  02  88  1907           BISB    #1@CDRP$V_BUFFER, -
                            190B            CDRP$B_FLAGS(R5)         ; Set buffer flag
   D0 A5  52   FE00 8F  AB  190B           BICW3   #^XFE00,R2, -
                            1912            CDRP$W_BOFF(R5)          ; Set the byte offset
             15    09   EF  1912           EXTZV   #VA$V_VPN,#VA$S_VPN,-
             52    52       1915            R2,R2                    ; Virtual page number
     50   00000000'9F   D0  1917           MOVL    @#MMG$GL_SPTBASE, R0      ; Get address of sys page table
       CC A5   6042     DE  191E           MOVAL   (R0)[R2],CDRP$L_SVAPTE(R5)  ; Set in CDRP
       D2 A5    51      D0  1923  130$:    MOVL    R1,CDRP$L_BCNT(R5)        ; Set the length = byte count
                            1927  ;
                            1927  ; Almost done, now check for shadowing and allocate multiple IRP's
                            1927  ;
                 4C A5  D4  1927           CLRL    CDRP$L_SHDW_LNK(R5)       ; Clear out shadow irp link
          51     A0 A5  DE  192A           MOVAL   CDRP$L_IOQFL(R5),R1       ; Cover IRP portion w/R1
               0200 8F  B3  192E           BITW    #1@MSCP$V_UF_SSMST,-      ; If shadow unit
                 12 A4      1932            UQB$W_UNT_FLGS(R4)
                    09  12  1934           BNEQ    SETUP_SHADOW              ;  then do shadow specific
                            1936
                            1936  SETUP_OUT:
          54     60 A5  D0  1936           MOVL    CDRP$L_CDTX(R5),R4        ; Pick up the CDT address
          54     10 A4  D0  193A           MOVL    CDT$L_PDT(R4),R4          ; Set up PDT address
                        05  193E           RSB
                            193F
                            193F  ;
                            193F  ; Shadowing enabled, grab extra IRP's for the multiple operations.
                            193F  ;
                            193F
                            193F  SETUP_SHADOW:
                    22  91  193F           CMPB    #MSCP$K_OP_WRITE,-
                 FB A5      1941            CDRP$B_OPCODE(R5)        ; Leave if not write
                    03  13  1943           BEQL    SETUP_SHADOW_WRITE
                  00D7  31  1945           BRW     SETUP_SHADOW_READ
                            1948  ;
                            1948  ; Shadow writes.  Check for CIP.  If so may need to hold write.  If not
                            1948  ; spawn extra IRP's for the multiple write operations.
                            1948  ;
                            1948  SETUP_SHADOW_WRITE:
                    01  B3  1948           BITW    #UQB$M_CIP,-
```

```
         2B A4       194A  3283                  UQB$B_FLAGS(R4)            ; If CIP not in progress
            45   13  194C  3284          BEQL    20$                        ;  just continue on
         58 A5   D4  194E  3285          CLRL    CDRP$L_FR4X(R5)            ; Clear a flag
      50 48 A5   D0  1951  3286          MOVL    CDRP$L_LBN(R5),R0          ; Pick up starting block
51  E4 A5 F7 8F  78  1955  3287          ASHL    #-9,CDRP$L_OBCNT(R5),R1    ; Get block range
      51    50   C0  195B  3288          ADDL    R0,R1                      ; Now ending block
   5C A4    50   D1  195E  3289          CMPL    R0,UQB$L_FENCEH(R4)        ; Check
            2F   1E  1962  3290          BGEQU   20$                        ; Outside (over)
         58 A5   D6  1964  3291          INCL    CDRP$L_FR4X(R5)            ; Set a flag
   58 A4    51   D1  1967  3292          CMPL    R1,UQB$L_FENCEL(R4)        ; Check block range
            26   1B  196B  3293          BLEQU   20$                        ; Outside (under)
            08   88  196D  3294 10$:     BISB    #UQB$M_CMDPEND,-           ; Set a write pending
         2B A4       196F  3295                  UQB$B_FLAGS(R4)
       18 A5 8ED0    1971  3296          POPL    CDRP$L_SAVD_RTN(R5)        ; Hold return adress
          F956   30  1975  3297          BSBW    DEALLOC_BUFF               ; Get rid of buffer
      50 E8 A5   0F  1978  3298          REMQUE  CDRP$L_UQB_FL(R5),R0       ; Remove from executing queue
         50 A4   B7  197C  3299          DECW    UQB$W_NUM_QUE(R4)          ;  (make count agree)
   74 B4    60   0E  197F  3300          INSQUE  (R0),@UQB$Q_SHQ+4(R4)      ; Insert on special queue
   D6 A5    06   B0  1983  3301          MOVW    #6,CDRP$W_STATE(R5)        ; Set state as special queued
                     1987  3302          WFINT                              ; Block and wait for copy to complete
         18 A5   DD  198A  3303          PUSHL   CDRP$L_SAVD_RTN(R5)        ; Restore
      50    0B   B0  198D  3304          MOVW    #IO$_WRITEPBLK,R0          ; Set for write
          FE2C   31  1990  3305          BRW     SETUP                      ; Now try again
                     1993  3306 ;
                     1993  3307 ; Now either we had no copy-in-progress on the shadow unit or we did but
                     1993  3308 ; the IO was outside the fence.  If the latter, then we write to all
                     1993  3309 ; units if under the fence; and to all units except the CIP unit if over.
                     1993  3310 ;
   01    26 A4   B1  1993  3311 20$:     CMPW    UQB$W_SHDW_STS(R4),#1      ; If shadow count <= 1,
            03   1A  1997  3312          BGTRU   25$                        ;  then
          FF9A   31  1999  3313          BRW     SETUP_OUT                  ;  do NOT shadow
                     199C  3314
   7E    26 A4   3C  199C  3315 25$:     MOVZWL  UQB$W_SHDW_STS(R4),-(SP); Set up unit counter
   51    A0 A5   DE  19A0  3316          MOVAL   CDRP$L_IOQFL(R5),R1        ; Cover IRP portion w/R1
      53    64   D0  19A4  3317 30$:     MOVL    UQB$L_SHDW_FL(R4),R3       ; Pick up possible shadown UQB's
            01   93  19A7  3318          BITB    #UQB$M_CIP,-               ; Check for copy in progress
         2B A3       19A9  3319                  UQB$B_FLAGS(R3)
            09   13  19AB  3320          BEQL    40$                        ; None, all is okay
      05 58 A5   E8  19AD  3321          BLBS    CDRP$L_FR4X(R5),40$        ; CIP, but IO is under the fence
      54    53   D0  19B1  3322          MOVL    R3,R4                      ; Ooops, try another
            60   11  19B4  3323          BRB     50$                        ; Check for end of line
                     19B6  3324 ;
                     19B6  3325 ; Grab an IRP for each active shadow set member
                     19B6  3326 ;
            51   DD  19B6  3327 40$:     PUSHL   R1                         ; Save R1
51     00C4 8F   3C  19B8  3328          MOVZWL  #IRP$C_LENGTH,R1           ; Size for an IRP
          EC94   30  19BD  3329          BSBW    ALLOMEM                    ; Grab one
          51 8ED0    19C0  3330          POPL    R1                         ; Restore
         03 50   E8  19C3  3331          BLBS    R0,45$                     ; Success
          0089   31  19C6  3332          BRW     SETUP_ERR                  ; No space left
                     19C9  3333 ;
                     19C9  3334 ; Fill in the IRP with the essential transfer fields
                     19C9  3335 ;
                 D0  19C9  3336 45$:     MOVL    #<DYN$C_IRP@16>!-          ; Set size and DYN code
                     19CA  3337                  <<IRP$C_LENGTH+15>&-16>,-
08 A2  000A00D0 8F   19CA  3338                  IRP$W_SIZE(R2)
            01   88  19D1  3339          BISB    #1@CDRP$V_SLAVE,-          ; Set the slave IRP flag
```

```
      FA A5        19D3                    CDRP$B_FLAGS(R5)
 00B4 C2   55   D0 19D5        MOVL        R5,IRP$L_MAINCDRP(R2)     ; Set pointer to CDRP
   24 A2   53   D0 19DA        MOVL        R3,IRP$L_UQB(R2)          ; Set pointer to UQB
    00AC C2     D4 19DE        CLRL        IRP$L_SHDW_LNK(R2)        ; Clear shadowing pointer
 00AC C1   52   D0 19E2        MOVL        R2,IRP$L_SHDW_LNK(R1)     ; Set back pointer
      CA A5     7D 19E7        MOVQ        CDRP$W_STS(R5),-          ; Copy transfer parameters
      2A A2        19EA                    IRP$W_STS(R2)
      D2 A5     D0 19EC        MOVL        CDRP$L_BCNT(R5),-         ; Ditto
      32 A2        19EF                    IRP$L_BCNT(R2)
      48 A5     D0 19F1        MOVL        CDRP$L_LBN(R5),-          ; Ditto
    00A8 C2        19F4                    IRP$L_LBN(R2)
      54   53   D0 19F7        MOVL        R3,R4                     ; Copy UQB pointer
      C0 A5     D0 19FA        MOVL        CDRP$W_FUNC(R5),-         ; Ditto
      20 A2        19FD                    IRP$W_FUNC(R2)
      30 A4     D0 19FF        MOVL        UQB$L_UCB(R4),-           ; Ditto
      1C A2        1A02                    IRP$L_UCB(R2)
      53   52   D0 1A04        MOVL        R2,R3                     ; Cover IRP
      F8 A5     D0 1A07        MOVL        CDRP$W_SLOT(R5),-         ; Copy flags and unit
      58 A2        1A0A                    IRP$W_SLOT(R2)
      51   52   D0 1A0C        MOVL        R2,R1                     ; Copy IRP pointer
      36 A2     B4 1A0F        CLRW        IRP$W_STATE(R2)           ; Clear out indicator
         01     88 1A12        BISB        #1@CDRP$V_SLAVE,-         ; Set the slave IRP flag
      5A A1        1A14                    IRP$B_FLAGS(R1)
         6E     D7 1A16 50$:   DECL        (SP)                      ; Count down
         8A     1A 1A18        BGTRU       30$                       ; More to go
         8E     D5 1A1A        TSTL        (SP)+                     ; Clean stack
       FF17     31 1A1C        BRW         SETUP_OUT                 ; End of the line
                   1A1F ;
                   1A1F ; Shadow reads.  Pick the next unit to read from.  Advance the ring pointer.
                   1A1F ;
                   1A1F SETUP_SHADOW_READ:
      51   64   D0 1A1F        MOVL        UQB$L_SHDW_FL(R4),R1      ; Pick up first unit
   74 A5   51   D0 1A22        MOVL        R1,CDRP$L_STRT_UQB(R5)    ; Set end condition
         15     10 1A26        BSBB        SELECT_FIRST_UNIT         ; Pick up first useable reader
         06     13 1A28        BEQL        999$                      ; ERROR
         61     D0 1A2A        MOVL        UQB$L_SHDW_FL(R1),-       ; Update master UQB pointer
         64        1A2C                    UQB$L_SHDW_FL(R4)
       FF06     31 1A2D        BRW         SETUP_OUT                 ; Leave
                   1A30
                   1A30 999$:  BUG_CHECK   MSCPSERV,FATAL            ; ERROR
                   1A34 ;
                   1A34 ; SELECT_NEXT_UNIT - Loop thru the shadow members
                   1A34 ;
                   1A34 ;      R1 -->  Base UQB to start from
                   1A34 ; **** USE CIP IF BELOW FENCE?
                   1A34 ;
                   1A34        .ENABLE LSB
                   1A34
                   1A34 SELECT_NEXT_UNIT:
      51   61   D0 1A34 10$:   MOVL        UQB$L_SHDW_FL(R1),R1      ; Pick up next candidate
   74 A5   51   D1 1A37        CMPL        R1,CDRP$L_STRT_UQB(R5)    ; Have we gone around the ring
         14     13 1A3B        BEQL        20$                       ; Yes, just error out
                   1A3D
                   1A3D SELECT_FIRST_UNIT:
         01     93 1A3D        BITB        #UQB$M_CIP,-              ; Check for possible copy in progress
      2B A1        1A3F                    UQB$B_FLAGS(R1)
         F1     12 1A41        BNEQ        10$                       ; Yes, disqualify
```

```
      0E A1   B5  1A43  3397           TSTW    UQB$W_STATUS(R1)         ; Check for ONLINE
         EC   12  1A46  3398           BNEQ    10$                      ; No, disqualify
   C4 A5  51  D0  1A48  3399           MOVL    R1,CDRP$L_UQB(R5)        ; Transfer new UQB pointer
      30 A1   D0  1A4C  3400           MOVL    UQB$L_UCB(R1),-          ; Transfer UCB pointer
      BC A5       1A4F  3401                    CDRP$L_UCB(R5)
              05  1A51  3402  20$:     RSB                              ; Leave with status
                  1A52  3403
                  1A52  3404           .DISABLE LSB
                  1A52  3405  ;
                  1A52  3406  ; ERROR handler
                  1A52  3407  ;
                  1A52  3408  SETUP_ERR:
      5E  08  C0  1A52  3409           ADDL    #8,SP                    ; Get rid of return address & count
 50   00AA 8F 3C  1A55  3410           MOVZWL  #<5*MSCP$K_ST_SBCOD!-
                  1A5A  3411                    MSCP$K_ST_CNTLR>,R0     ; Set error status ;*** TEMP
       F077   31  1A5A  3412           BRW     SEND_PKT                 ; Leave in disgrace
```

```
                                  1A5D          .SBTTL  DO_DISK & BACK - DISK IO utility routines
                                  1A5D
                                  1A5D  ;++
                                  1A5D  ;
                                  1A5D  ; DO_DISK - Routine to set up IRP for disk I/O
                                  1A5D  ;
                                  1A5D  ; Inputs:
                                  1A5D  ;
                                  1A5D  ;               R3 -    Current IRP address
                                  1A5D  ;               R4 -    PDT address
                                  1A5D  ;               R5 -    (Master) CDRP address
                                  1A5D  ;
                                  1A5D  ;--
                                  1A5D
                                  1A5D          .ENABLE LSB
                                  1A5D
                                  1A5D  DO_DISK:
              53    A0 A5     9E  1A5D          MOVAB   CDRP$L_IOQFL(R5),R3     ; Cover IRP portion
                                  1A61
                                  1A61  DO_DISKA:
              51    C4 A5     D0  1A61          MOVL    CDRP$L_UQB(R5),R1       ; Pick up UQB ptr
                    0E A1     B5  1A65          TSTW    UQB$W_STATUS(R1)        ; Has it gone AVAIL while we waited?
                       0A     13  1A68          BEQL    DO_DISKX                ; No, finish IO
              50  01A4 8F     3C  1A6A          MOVZWL  #SS$_MEDOFL,R0          ; Yes, give error code
                              05  1A6F          RSB                             ;  and leave
                                  1A70
                                  1A70  INV_FUNC:
                                  1A70          BUG_CHECK       MSCPSERV,FATAL  ; Invalid function code
                                  1A74
                                  1A74  DO_DISKX:
                00B0 C3     8ED0  1A74          POPL    IRP$L_FPCX(R3)          ; Where to return after IOPOST
              00B8 C3    54   D0  1A79          MOVL    R4,IRP$L_FR4X(R3)       ; Save R4
                                  1A7E          ASSUME  CDRP$V_SLAVE EQ 0
              05 5A A3        E8  1A7E          BLBS    IRP$B_FLAGS(R3),5$      ; If this is a slave IRP, then
                    3C A5     D0  1A82          MOVL    CDRP$L_XCT_LEN(R5),-    ;  Save length in a safe place
                    64 A5         1A85                  CDRP$L_XCT_LENX(R5)
        0C A3   1B3F'CF       9E  1A87  5$:     MOVAB   W^BACK,IRP$L_PID(R3)    ; Addr to which IOPOST will return IRP
                    1C A5     D4  1A8D          CLRL    CDRP$L_MSG_BUF(R5)      ; Mark it not in use
                                  1A90  ; ??? begin bug trap
     50   20 A3    06    00   EF  1A90          EXTZV   #IO$V_FCODE, -          ; Isolate function code
                                  1A96                  #IO$S_FCODE, -
                                  1A96                  IRP$W_FUNC(R3),R0
                   0C    50   B1  1A96          CMPW    R0,#IO$_READPBLK        ; Read?
                         1D   13  1A99          BEQL    7$                      ; Branch if read
                   0B    50   B1  1A9B          CMPW    R0,#IO$_WRITEPBLK       ; Write?
                         18   13  1A9E          BEQL    7$                      ; Branch if write
                         55   DD  1AA0          PUSHL   R5                      ; Save CDRP address
              55    1C A3     D0  1AA2          MOVL    IRP$L_UCB(R3),R5        ; Get UCB address
            55    0088 C5     D0  1AA6          MOVL    UCB$L_DDT(R5),R5        ; DDT address
            C0 08 B5     50   E1  1AAB          BBC     R0,@DDT$L_FDT(R5),INV_FUNC      ; Branch if function code is invalid
                       55 8ED0    1AB0          POPL    R5                      ; Restore CDRP address
                   0A    50   B1  1AB3          CMPW    R0,#IO$_WRITECHECK      ; Writecheck?
                         38   12  1AB6          BNEQ    9$
           B4 A5     32 A3    D1  1AB8  7$:     CMPL    IRP$L_BCNT(R3),CDRP$L_BUFLEN(R5)
                         2D   1A  1ABD          BGTRU   8$                      ; Tilt!
     50   2C A3   00000000'9F C3  1ABF          SUBL3   @#MMG$GL_SPTBASE, -     ; Get address of buffer page table
                                  1AC8                  IRP$L_SVAPTE(R3),R0
```

```
  50   50   07   78  1AC8        ASHL    #9-2,R0,R0              ; Convert to address in segment
  00   50   1F   E2  1ACC        BBSS    #31,R0,71$              ; Set 80000000 bit
  50   D0   A5   A0  1AD0 71$:   ADDW2   CDRP$W_BOFF(R5),R0
  B8   A5   50   D1  1AD4        CMPL    R0,CDRP$L_BUFADR(R5)    ; Is buffer address consistent?
            12   12  1AD8        BNEQ    8$
  E9F1 CF   50   D1  1ADA        CMPL    R0,BUFF_ADDR
            0B   1F  1ADF        BLSSU   8$                      ; Start of buffer out of bounds
  50   32   A3   C0  1AE1        ADDL2   IRP$L_BCNT(R3),R0
  E9EA CF   50   D1  1AE5        CMPL    R0,BUFF_END             ; Is buffer contained within range?
            04   1B  1AEA        BLEQU   9$                      ; Branch if yes
                     1AEC 8$:    BUG_CHECK       MSCPSERV,FATAL  ; Invalid disk buffer address
                     1AF0
                     1AF0 9$:
                     1AF0 ; ??? end bug trap
  50   00A8 C3   D0  1AF0        MOVL    IRP$L_LBN(R3),R0        ; Pick up starting logical block
  55   24   A3   D0  1AF5        MOVL    IRP$L_UQB(R3),R5        ; Pick up UQB ptr
       54   A5   B6  1AF9        INCW    UQB$W_NUM_OPS(R5)       ; Bump number of packets queued to unit
       54   A5   B1  1AFC        CMPW    UQB$W_NUM_OPS(R5),-     ; Greater than current max?
       56   A5       1AFF                UQB$W_MAX_OPS(R5)
            05   1B  1B01        BLEQU   10$                     ; No, skip on
       54   A5   B0  1B03        MOVW    UQB$W_NUM_OPS(R5),-     ; Yes, replace current max
       56   A5       1B06                UQB$W_MAX_OPS(R5)
  55   1C   A3   D0  1B08 10$:   MOVL    IRP$L_UCB(R3),R5        ;  and now UCB
  00000000'9F    D5  1B0C        TSTL    @#PMS$GL_IOPFMPDB       ; I/O performance monitoring enabled ?
            23   12  1B12        BNEQ    30$                     ; BR if yes
                     1B14        DSBINT  #15                     ; Protect I/O sequence # manipulation
  00000000'9F    D0  1B1A        MOVL    @#PMS$GL_IOPFMSEQ,-     ; Insert I/O sequence # into IRP
       50   A3       1B20                IRP$L_SEQNUM(R3)
  00000000'9F    D6  1B22        INCL    @#PMS$GL_IOPFMSEQ       ; and bump global counter
                     1B28        ENBINT                          ; Restore previous IPL
  00000000'9F    16  1B2B 20$:   JSB     @#IOC$CVTLOGPHY         ; Convert logical to physical block
  00000000'9F    17  1B31        JMP     @#EXE$INSIOQ            ; Queue the IRP to the driver
                     1B37
  00000000'9F    16  1B37 30$:   JSB     @#PMS$START_RQ          ; Gather Start I/O Request statistics
            EC   11  1B3D        BRB     20$
                     1B3F
                     1B3F        .DISABLE LSB
                     1B3F ;
                     1B3F ; Routine will RSB back to our caller's caller which is in the PORT
                     1B3F ; DRIVER.  Thread is now suspended and will resume when IOPOST calls us
                     1B3F ; at entry BACK.
                     1B3F ;
                     1B3F ;          R5 -->  IRP
                     1B3F ;
                     1B3F BACK:  DSBINT  #IPL$_SCS               ; Set IPL to FORK
  0100 8F        AA  1B45        BICW    #IRP$M_PHYSIO,-         ; Clear the physical IO flag
       2A   A5       1B49                IRP$W_STS(R5)           ;  only needed for ONLINE
  00000000'9F    D5  1B4B        TSTL    @#PMS$GL_IOPFMPDB       ; I/O performance monitoring enabled ?
            06   13  1B51        BEQL    10$                     ; BR if no
  00000000'9F    16  1B53        JSB     @#PMS$END_RQ            ; Gather End I/O Request statistics
       53   55   D0  1B59 10$:   MOVL    R5,R3                   ; R3-->head of IRP
  54   00B8 C3   D0  1B5C        MOVL    IRP$L_FR4X(R3),R4       ; Restore PDT
  51   24   A3   D0  1B61        MOVL    IRP$L_UQB(R3),R1        ; Pick up UQB pointer
       54   A1   B7  1B65        DECW    UQB$W_NUM_OPS(R1)       ; Drop count of operations
  51   1C   A3   D0  1B68        MOVL    IRP$L_UCB(R3),R1        ; Pick up UCB pointer
       6A   A1   B7  1B6C        DECW    UCB$W_QLEN(R1)          ; Drop queue length counter
                     1B6F        ASSUME  CDRP$V_SLAVE EQ 0
```

```
                           1BB8          .SBTTL  UTILITY ROUTINES
                           1BB8
                           1BB8  ;*******************************************************************
                           1BB8  ;                                                                   *
                           1BB8  ;                                                                   *
                           1BB8  ;                          S E A R C H _ Q U E                      *
                           1BB8  ;                                                                   *
                           1BB8  ;                                                                   *
                           1BB8  ;*******************************************************************
                           1BB8  ;+
                           1BB8  ; Input:
                           1BB8  ;        R3 = Address of HQB possible having a CDRP with that command
                           1BB8  ;        R4 = Unit number
                           1BB8  ;        R5 = Address of CDRP containing command reference # to search for
                           1BB8  ;
                           1BB8  ; Output:
                           1BB8  ;        R0 = Status
                           1BB8  ;                Low bit clear if not found
                           1BB8  ;                Low bit set if found
                           1BB8  ;        R1 --> CDRP$L_UQB_FL of the CDRP found
                           1BB8  ;-
                           1BB8
                           1BB8  SEARCH_QUE:
              50      D4   1BB8          CLRL    R0                          ; Set for failure
    54   E4D1 CF44    D0   1BBA          MOVL    UNIT_VEC[R4],R4             ; Grab UQB
              54      13   1BC0          BEQL    40$                         ; None there
              56      DD   1BC2          PUSHL   R6                          ; Grab a register
       56  B0 A5      D0   1BC4          MOVL    CDRP$L_MSGBUFX(R5),R6       ; Pick up buffer
       56  0C A6      D0   1BC8          MOVL    MSCP$L_OUT_REF(R6),R6       ; Id to search for
           50 A4      B5   1BCC          TSTW    UQB$W_NUM_QUE(R4)           ; Any entries on normal que?
              09      13   1BCF          BEQL    10$                         ; No, leave now
       51  60 A4      9E   1BD1          MOVAB   UQB$L_CDRP_FL(R4),R1        ; Set to head of que
              40      10   1BD5          BSBB    50$                         ; Look for the packet
           39 50      E8   1BD7          BLBS    R0,30$                      ; Found it
                           1BDA
           7C A4      B5   1BDA  10$:    TSTW    UQB$W_NUM_BLK(R4)           ; Any entries on blocked que?
              09      13   1BDD          BEQL    20$                         ; No, leave now
       51  68 A4      9E   1BDF          MOVAB   UQB$Q_BLOCKQ(R4),R1         ; Set to head of que
              32      10   1BE3          BSBB    50$                         ; Look for the packet
           2B 50      E8   1BE5          BLBS    R0,30$                      ; Found it
                           1BE8
       51  70 A4      9E   1BE8  20$:    MOVAB   UQB$Q_SHQ(R4),R1            ; Set to head of special hold que
              29      10   1BEC          BSBB    50$                         ; Look for the packet
           22 50      E8   1BEE          BLBS    R0,30$                      ; Found it
              00      E1   1BF1          BBC     #UQB$V_CIP,-                ; If unit is part of shadow copy
        1D 2B A4           1BF3                   UQB$B_FLAGS(R4),30$        ;  then
    51     0088 C4    D0   1BF6          MOVL    UQB$L_CPY_CDRP(R4),R1       ; Pick up pointer to its CDRP
       51  E8 A1      9E   1BFB          MOVAB   CDRP$L_UQB_FL(R1),R1        ; Adjust pointer
           14 A3      D1   1BFF          CMPL    HQB$L_CDT(R3),-             ; Is this CDRP for this HQB?
           78 A1           1C02                   CDRP$L_CDTX-CDRP$L_UQB_FL(R1)
              0D      12   1C04          BNEQ    30$                         ; No
                           1C06          ASSUME  MSCP$L_CMD_REF EQ 0
           C8 A1      D5   1C06          TSTL    CDRP$L_MSGBUFX- -           ; Is there a message buffer?
                           1C09                   CDRP$L_UQB_FL(R1)
              08      13   1C09          BEQL    30$                         ; No, so can never match
       56  C8 B1      D1   1C0B          CMPL    @CDRP$L_MSGBUFX- -
                           1C0F                   CDRP$L_UQB_FL(R1),R6       ; Match com ref no in host chain
```

```
         02   12  1C0F  3612            BNEQ    30$                     ; No, leave
         50   D6  1C11  3613            INCL    R0                      ; Set success
      56 8ED0     1C13  3614 30$:       POPL    R6
         05       1C16  3615 40$:       RSB
                  1C17  3616
         50   D4  1C17  3617 50$:       CLRL    R0                      ; Set for failure
    52   51   D0  1C19  3618            MOVL    R1,R2                   ; Store pointer
    51   61   D0  1C1C  3619 60$:       MOVL    (R1),R1                 ; No, try next entry
    52   51   D1  1C1F  3620            CMPL    R1,R2                   ; End of que?
         14   13  1C22  3621            BEQL    70$                     ; Yes
      14 A3   D1  1C24  3622            CMPL    HQB$L_CDT(R3),-         ; Is this CDRP for this HQB?
      60 A5       1C27  3623              CDRP$L_CDTX(R5)
         F1   12  1C29  3624            BNEQ    60$                     ; No, flink on
                  1C2B  3625            ASSUME  MSCP$L_CMD_REF EQ 0
      C8 A1   D5  1C2B  3626            TSTL    CDRP$L_MSGBUFX- -       ; Is there a message buffer?
                  1C2E  3627              CDRP$L_UQB_FL(R1)
         EC   13  1C2E  3628            BEQL    60$                     ; No, so can never match
56    C8 B1   D1  1C30  3629            CMPL    @CDRP$L_MSGBUFX- -
                  1C34  3630              CDRP$L_UQB_FL(R1),R6      ; Match com ref no in host chain
         E6   12  1C34  3631            BNEQ    60$                     ; No, loop
         50   D6  1C36  3632            INCL    R0                      ; Set success
         05       1C38  3633 70$:       RSB
```

```
                            1C39        .SBTTL  DATA STRUCTURE ALLOCATION
                            1C39
                            1C39 ;*******************************************************************
                            1C39 ;                                                                   *
                            1C39 ;                                                                   *
                            1C39 ;                           GET_STRUCT                              *
                            1C39 ;                                                                   *
                            1C39 ;                                                                   *
                            1C39 ;*******************************************************************
                            1C39 ;+
                            1C39 ; Input:
                            1C39 ;          NONE
                            1C39 ;
                            1C39 ; Output:
                            1C39 ;          R0 = Status
                            1C39 ;                  Low bit clear if none available
                            1C39 ;                  Low bit set if found
                            1C39 ;          R5 --> CDRP found
                            1C39 ;-
                            1C39
                            1C39 GET_STRUCT:
        55   E433 DF   0F   1C39        REMQUE  @CDRP_LIST,R5               ; Pick up a CDRP
               10   1D      1C3E        BVS     30$                         ; None left
             E404 CF   B7   1C40        DECW    NUM_PKT                     ; Drop number of free packets
             50   01   D0   1C44        MOVL    S^#SS$_NORMAL,R0            ; Set success status
                       05   1C47        RSB
                            1C48
        50   0124 8F   3C   1C48 20$:   MOVZWL  #SS$_INSFMEM,R0             ; ERROR
                  1E   BA   1C4D        POPR    #^M<R1,R2,R3,R4>            ; Restore volatile registers
                       05   1C4F        RSB                                 ; Leave
                            1C50
                  1E   BB   1C50 30$:   PUSHR   #^M<R1,R2,R3,R4>            ; Save volatile registers
        51   00E0 8F   3C   1C52        MOVZWL  #BIG_CDRP,R1                ; Get total size
                E9FA   30   1C57        BSBW    ALLOMEM                     ; Allocate them
               EB 50   E9   1C5A        BLBC    R0,20$                      ; No memory available
                            1C5D ;
                            1C5D ; Fill in skeletal CDRP
                            1C5D ;
             7E   51   7D   1C5D        MOVQ    R1,-(SP)                    ; Save
62   51   00 6E   00   2C   1C60        MOVC5   #0,(SP),#0,R1,(R2)          ; Clear it all out
             54   8E   7D   1C66        MOVQ    (SP)+,R4                    ; Recover length
          55   60 A5   DE   1C69        MOVAL   -CDRP$L_IOQFL(R5), R5       ; Set to the CDRP portion
          AA A5   0A   90   1C6D        MOVB    #DYN$C_IRP, -
                            1C71                CDRP$B_IRP_TYPE(R5)         ; Fill in size/type/rmod fields
          A8 A5   54   B0   1C71        MOVW    R4,CDRP$W_IRP_SIZE(R5)      ; Fill in size/type/rmod fields
          0B A5   08   90   1C75        MOVB    #IPL$_SCS,CDRP$B_FIPL(R5)   ; CDRP IPL
          0A A5   39   90   1C79        MOVB    #DYN$C_CDRP, -
                            1C7D                CDRP$B_CD_TYPE(R5)          ; CDRP block type
             FFA0 8F   B0   1C7D        MOVW    #CDRP$C_IOQFL, -
                08 A5       1C81                CDRP$W_CDRPSIZE(R5)         ; Ditto for smaller portion
     C3 A5   E851 CF   90   1C83        MOVB    IO_PRI,CDRP$B_PRI(R5)       ; Set the IO priority
     2C A5     68 A5   DE   1C89        MOVAL   CDRP$B_LBUFF(R5), -         ; Set pointer to local buffer handle
                            1C8E                CDRP$L_LBUFH_AD(R5)
             E3B4 CF   B6   1C8E        INCW    INI_PKT                     ; Total packets
                  1E   BA   1C92        POPR    #^M<R1,R2,R3,R4>            ; Restore volatile registers
                  02   10   1C94        BSBB    FREE_STRUCT                 ; Put on free list -- has side effect
                            1C96                                            ;  of initializing CDRP$L_FQFL linkage
```

```
               A1   11  1C96         BRB     GET_STRUCT                ; Try again
                        1C98
                        1C98  ;*****************************************************************
                        1C98  ;                                                                *
                        1C98  ;                                                                *
                        1C98  ;                         FREE_STRUCT                            *
                        1C98  ;                                                                *
                        1C98  ;                                                                *
                        1C98  ;*****************************************************************
                        1C98  ;+
                        1C98  ; Input:
                        1C98  ;          R5 --> CDRP
                        1C98  ;
                        1C98  ; Output:
                        1C98  ;          NONE
                        1C98  ;-
                        1C98
                        1C98  FREE_STRUCT:
                        1C98  ; Mung structure to trap bugs
       A4 A5      01 D0 1C98         MOVL    #1,CDRP$L_IOQBL(R5)       ; Zap IRP queue linkage to catch bugs
       04 A5      01 D0 1C9C         MOVL    #1,CDRP$L_FQBL(R5)        ; Zap CDRP queue linkage to catch bugs
       EC A5      01 D0 1CA0         MOVL    #1,CDRP$L_UQB_BL(R5)      ; Zap another queue linkage
       24 A5      01 D0 1CA4         MOVL    #1,CDRP$L_CDT(R5)         ; ZAP CDT address
       60 A5      01 D0 1CA8         MOVL    #1,CDRP$L_CDTX(R5)        ; ZAP CDT address
       B0 A5      01 D0 1CAC         MOVL    #1,CDRP$L_MSGBUFX(R5)     ; ZAP SCS message buffer address
       1C A5      01 D0 1CB0         MOVL    #1,CDRP$L_MSG_BUF(R5)     ; ZAP SCS message buffer address
       C4 A5      01 D0 1CB4         MOVL    #1,CDRP$L_UQB(R5)         ; ZAP UQB address
       78 A5      01 D0 1CB8         MOVL    #1,CDRP$L_MST_UQB(R5)     ; ZAP master UQB address
       54 A5      01 D0 1CBC         MOVL    #1,CDRP$L_FR3X(R5)        ; Alia IRP$L_MAINCDRP
       7C A5      01 D0 1CC0         MOVL    #1,CDRP$L_HQB(R5)         ; ZAP HQB address
       D6 A5      3F B0 1CC4         MOVW    #63,CDRP$Q_STATE(R5)      ; Set state invalid
    0C A5      E1'AF 9E 1CC8         MOVAB   B^20$,CDRP$L_FPC(R5)      ; Zap CDRP fork PC
    AC A5      E1'AF 9E 1CCD         MOVAB   B^20$,CDRP$L_PID(R5)      ; Zap IRP fork PC
    50 A5      E1'AF 9E 1CD2         MOVAB   B^20$,CDRP$L_FPCX(R5)     ; Zap another fork PC
                        1CD7  ; Now deallocate the structure
                        1CD7  ;; Avoid problems (maybe) by hanging onto packets forever
                        1CD7  ;;         CMPW    NUM_PKT,MAX_PKT           ; Are enough packets cached?
                        1CD7  ;;         BGEQU   10$                       ; Branch if yes
     E398 DF      65 0E 1CD7         INSQUE  (R5),@CDRP_LIST+4         ; Return the cdrp
        E368 CF      B6 1CDC         INCW    NUM_PKT                   ; Account for it
                     05 1CE0         RSB
                        1CE1
                        1CE1  ;;10$:     PUSHR   #^M<R0,R1,R2,R3>          ; Save volatile registers
                        1CE1  ;;         MOVAL   CDRP$L_IOQFL(R5),R0       ; Structure address
                        1CE1  ;;         BSBW    DEALMEM                   ; Deallocate structure
                        1CE1  ;;         POPR    #^M<R0,R1,R2,R3>          ; Restore volatile registers
                        1CE1  ;;         DECW    INI_PKT                   ; Drop count of allocated structures
                        1CE1  ;;         RSB                               ; Return
                        1CE1
                        1CE1  20$:       BUG_CHECK       MSCPSERV,FATAL
                        1CE5
```

```
                                  1CE5           .SBTTL  READ ONLY DATA
                                  1CE5
                                  1CE5           .ALIGN  LONG
                                  1CE8
                        00000000  1CE8  LIS_CDT:.LONG   0
                                  1CEC  ;
                                  1CEC  SERV_INFO:
                                  1CEC  SERV_NAME:
20 20 20 4B 53 49 44 24 50 43 53 4D  1CEC          .ASCII  /MSCP$DISK       /
                     20 20 20 20  1CF8
                                  1CFC  ;
                                  1CFC  COM_PKT_LEN:
         00 00 00 1C 0C 14 10 00  1CFC           .BYTE   0*4,4*4,5*4,3*4,7*4,0*4,0*4,0
         00 00 00 00 0C 24 24 0C  1D04           .BYTE   3*4,9*4,9*4,3*4,0*4,0*4,0*4,0
         00 00 00 20 20 20 20 00  1D0C           .BYTE   0*4,8*4,8*4,8*4,8*4,0*4,0*4,0
         00 00 00 00 00 00 00 00  1D14           .BYTE   0*4,0*4,0*4,0*4,0*4,0*4,0*4,0
         00 00 00 00 00 20 20 20  1D1C           .BYTE   8*4,8*4,8*4,0*4,0*4,0*4,0*4,0
                                  1D24  ;
                                  1D24  END_PKT_LEN:
         00 00 00 1C 30 14 10 00  1D24           .BYTE   0*4,4*4,5*4,12*4,7*4,0*4,0*4,0
         00 00 00 00 0C 2C 2C 0C  1D2C           .BYTE   3*4,11*4,11*4,3*4,0*4,0*4,0*4,0
         00 00 00 0C 20 20 20 20  1D34           .BYTE   8*4,8*4,8*4,8*4,3*4,0*4,0*4,0
         00 00 00 00 00 00 00 00  1D3C           .BYTE   0*4,0*4,0*4,0*4,0*4,0*4,0*4,0
         00 00 00 00 00 20 20 20  1D44           .BYTE   8*4,8*4,8*4,0*4,0*4,0*4,0*4,0
                                  1D4C  ;
                                  1D4C  MOD_TBL:
                            FFFF  1D4C           .WORD   ^C0
                            FFFF  1D4E           .WORD   ^C0                     ; 1-ABORT
                            FFFF  1D50           .WORD   ^C0                     ; 2-GET CMD STS
                            DFFE  1D52           .WORD   ^C<MSCP$M_MD_NXUNT!-
                                  1D54                   MSCP$M_MD_CLSEX>        ; 3-GET UNIT STS
                            FFFF  1D54           .WORD   ^C0                     ; 4-SET CTRL CHAR
                            FFFF  1D56           .WORD   ^C0                     ; 5
                            FFFF  1D58           .WORD   ^C0                     ; 6
                            FFFF  1D5A           .WORD   ^C0                     ; 7
                            DFFC  1D5C           .WORD   ^C<MSCP$M_MD_SPNDW!-    ; 8-AVAILABLE
                                  1D5E                   MSCP$M_MD_ALLCD!-
                                  1D5E                   MSCP$M_MD_CLSEX>
                            DFE8  1D5E           .WORD   ^C<MSCP$M_MD_RIP!-      ; 9-ONLINE
                                  1D60                   MSCP$M_MD_IGNMF!-
                                  1D60                   MSCP$M_MD_STWRP!-
                                  1D60                   MSCP$M_MD_SHDSP!-
                                  1D60                   MSCP$M_MD_CLSEX>
                            DFEB  1D60           .WORD   ^C<MSCP$M_MD_STWRP!-    ; 10-SET UNIT CHAR
                                  1D62                   MSCP$M_MD_SHDSP!-
                                  1D62                   MSCP$M_MD_CLSEX>
                            FFFF  1D62           .WORD   ^C0                     ; 11-DETERMINE ACCESS PATH
                            FFFF  1D64           .WORD   ^C0                     ; 12
                            FFFF  1D66           .WORD   ^C0                     ; 13
                            FFFF  1D68           .WORD   ^C0                     ; 14
                            FFFF  1D6A           .WORD   ^C0                     ; 15
                            507F  1D6C           .WORD   ^C<MSCP$M_MD_EXPRS!-    ; 16-ACCESS
                                  1D6E                   MSCP$M_MD_CLSEX!-
                                  1D6E                   MSCP$M_MD_SCCHH!-
                                  1D6E                   MSCP$M_MD_SCCHL!-
                                  1D6E                   MSCP$M_MD_SEREC!-
                                  1D6E                   MSCP$M_MD_SECOR!-
```

```
           1D6E                         MSCP$M_MD_SSHDW>
     507F  1D6E              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 17-COMP CTRL DATA
           1D70                         MSCP$M_MD_CLSEX!-
           1D70                         MSCP$M_MD_SCCHH!-
           1D70                         MSCP$M_MD_SCCHL!-
           1D70                         MSCP$M_MD_SEREC!-
           1D70                         MSCP$M_MD_SECOR!-
           1D70                         MSCP$M_MD_SSHDW>
     4E0F  1D70              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 18-ERASE
           1D72                         MSCP$M_MD_SEREC!-
           1D72                         MSCP$M_MD_ERROR!-
           1D72                         MSCP$M_MD_WBKVL!-
           1D72                         MSCP$M_MD_WBKNV!-
           1D72                         MSCP$M_MD_CLSEX!-
           1D72                         MSCP$M_MD_WRSEQ!-
           1D72                         MSCP$M_MD_SSHDW>
     5C7C  1D72              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 19-FLUSH
           1D74                         MSCP$M_MD_FLENU!-
           1D74                         MSCP$M_MD_CLSEX!-
           1D74                         MSCP$M_MD_SECOR!-
           1D74                         MSCP$M_MD_SEREC!-
           1D74                         MSCP$M_MD_SSHDW!-
           1D74                         MSCP$M_MD_VOLTL>
     5FFE  1D74              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 20-REPLACE
           1D76                         MSCP$M_MD_CLSEX!-
           1D76                         MSCP$M_MD_PRIMR>
     FFFF  1D76              .WORD   ^C0                       ; 21
     FFFF  1D78              .WORD   ^C0                       ; 22
     FFFF  1D7A              .WORD   ^C0                       ; 23
     FFFF  1D7C              .WORD   ^C0                       ; 24-MAINT READ
     FFFF  1D7E              .WORD   ^C0                       ; 25-MAINT WRITE
     FFFF  1D80              .WORD   ^C0                       ; 26
     FFFF  1D82              .WORD   ^C0
     FFFF  1D84              .WORD   ^C0                       ; 28
     FFFF  1D86              .WORD   ^C0
     FFFF  1D88              .WORD   ^C0                       ; 30
     FFFF  1D8A              .WORD   ^C0
     507F  1D8C              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 32-COMP HOST DATA
           1D8E                         MSCP$M_MD_CLSEX!-
           1D8E                         MSCP$M_MD_SCCHH!-
           1D8E                         MSCP$M_MD_SCCHL!-
           1D8E                         MSCP$M_MD_SEREC!-
           1D8E                         MSCP$M_MD_SECOR!-
           1D8E                         MSCP$M_MD_SSHDW>
     107F  1D8E              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 33-READ
           1D90                         MSCP$M_MD_COMP!-
           1D90                         MSCP$M_MD_CLSEX!-
           1D90                         MSCP$M_MD_SCCHH!-
           1D90                         MSCP$M_MD_SCCHL!-
           1D90                         MSCP$M_MD_SEREC!-
           1D90                         MSCP$M_MD_SECOR!-
           1D90                         MSCP$M_MD_SSHDW>
     0C0F  1D90              .WORD   ^C<MSCP$M_MD_EXPRS!-      ; 34-WRITE
           1D92                         MSCP$M_MD_COMP!-
           1D92                         MSCP$M_MD_CLSEX!-
           1D92                         MSCP$M_MD_ERROR!-
           1D92                         MSCP$M_MD_WRSEQ!-
```

```
                1D92  3857                          MSCP$M_MD_WBKNV!-
                1D92  3858                          MSCP$M_MD_WBKVL!-
                1D92  3859                          MSCP$M_MD_SEREC!-
                1D92  3860                          MSCP$M_MD_SECOR!-
                1D92  3861                          MSCP$M_MD_SSHDW>
         FFFF   1D92  3862          .WORD   ^C0                       ; 35
         FFFF   1D94  3863          .WORD   ^C0
         FFFF   1D96  3864          .WORD   ^C0                       ; 37
         FFFF   1D98  3865          .WORD   ^C0
         FFFF   1D9A  3866          .WORD   ^C0                       ; 39
                1D9C  3867  ;
                1D9C  3868  ; Patch area
                1D9C  3869  ;
                1D9C  3870
     00001D9C   1D9C  3871          LC=.
     00001DA0   1D9C  3872          .=<LC+15>&-16
     00000078   1DA0  3873  PATCH:  .LONG   120
     00001DA8'  1DA4  3874          .LONG   PATCH+8
     00001E20   1DA8  3875          .BLKB   120
                1E20  3876
```

```
                              1E20         .SBTTL  INITIALIZATION ROUTINE
                              1E20
                              1E20    ;++
                              1E20    ;
                              1E20    ; This routine is called to initialize MSCP server from SYSGEN.
                              1E20    ;
                              1E20    ; Inputs:
                              1E20    ;
                              1E20            AP -->  #Args
                              1E20    ;               action #
                  00000008    1E20                    BUF=8    ; BUFfers
                  0000000C    1E20                    PAC=12   ; PACkets
                  00000010    1E20                    HOS=16   ; HOSts
                  00000014    1E20                    TIM=20   ; TImeout
                  00000018    1E20                    PRI=24   ; PRIority
                  0000001C    1E20                    SMA=28   ; SMAllest buffer
                  00000020    1E20                    FRA=32   ; FRActional buffer
                              1E20    ;
                              1E20    ; Outputs:
                              1E20    ;
                              1E20            R0      Status code
                              1E20    ;
                              1E20    ;--
                              1E20
                              1E20            .ENABLE LSB
                              1E20
        50   0124 8F    3C    1E20    10$:    MOVZWL  #SS$_INSFMEM,R0              ; Insufficient non-paged pool
                00FA    31    1E25            BRW     40$
                              1E28
                              1E28    START_CONT:
         51    08 AC    D0    1E28            MOVL    BUF(AP),R1                   ; Pick up user specified
       51   01FF C1     9E    1E2C            MOVAB   511(R1),R1
       51   01FF 8F     AA    1E31            BICW    #511,R1                      ; Round up to page
       E219 CF   51     D0    1E36            MOVL    R1,MAX_BUF                   ; Save size
            51   10     C0    1E3B            ADDL    #16,R1                       ; Allow for header
                              1E3E            SETIPL  #2                           ; Set us non-deletable
               E810     30    1E41            BSBW    ALLOMEM                      ; Allocate it
              D9 50     E9    1E44            BLBC    R0,10$                       ; Can't allocate buffer
       E680 CF   51     D0    1E47            MOVL    R1,BUFF_LENGTH               ; Length of buffer
       E67F CF   52     D0    1E4C            MOVL    R2,BUFF_ADDR                 ; Buffer header address
  E67D CF   52   51     C1    1E51            ADDL3   R1,R2,BUFF_END               ; First byte beyond buffer
                 82     7C    1E57            CLRQ    (R2)+                        ; Clear FLINK,BLINK
  82    51   00130000 8F C1   1E59            ADDL3   #<DYN$C_BUFIO@16>,R1,(R2)+ ; Set pool info
                 82     D4    1E61            CLRL    (R2)+                        ; Skip to 16-byte boundary
       E1E8 CF   52     D0    1E63            MOVL    R2,BUFF_HEAD                 ; Set pointer to buffer pool
                 82     D4    1E68            CLRL    (R2)+                        ; Set up head
       62   51   10     C3    1E6A            SUBL3   #16,R1,(R2)                  ; Ditto
                              1E6E    ;
                              1E6E    ; Pre-allocate CDRP's
                              1E6E    ;
         57    0C AC    D0    1E6E            MOVL    PAC(AP),R7                   ; Pick up user specified
            02   57     D1    1E72            CMPL    R7,#2                        ; Must be at least 2
                 03     1E    1E75            BGEQU   15$                          ; Branch if 2 or more
            57   02     D0    1E77            MOVL    #2,R7                        ; Make it 2
       E1C5 CF   57     B0    1E7A    15$:    MOVW    R7,PACKET                    ; Send to MSCP
         51    10 AC    D0    1E7F            MOVL    HOS(AP),R1                   ; Pick un number of hosts
       E1C4 CF   51     B0    1E83            MOVW    R1,INI_HOST                  ; Save in comm area
```

```
          E1C1 CF    51    B0  1E88       MOVW    R1,NUM_HOST             ; Save in comm area
     E1B7 CF    57    51    C5  1E8D       MULL3   R1,R7,MAX_PKT           ; Times the hosts = maximum packets to cache
                                1E93  ;
                                1E93  ; Set up the queues.
                                1E93  ;
          55   E169 CF    9E  1E93       MOVAB   MSCP_BEGIN,R5           ; Point to our header
          54   E1D4 CF    DE  1E98       MOVAL   CDRP_LIST,R4            ; Start address of queue
               50    06    D0  1E9D       MOVL    #2*MSU$C_NUMQUE,R0      ; Number of queues
               84    55    C0  1EA0  20$: ADDL2   R5,(R4)+                ; Make addresses absolute
                  FA 50    F5  1EA3       SOBGTR  R0,20$                  ; Loop thru
          50   E60A CF    7E  1EA6       MOVAQ   HQBLIST,R0              ; List of HQB's
               60    50    D0  1EAB       MOVL    R0,(R0)                 ; Initialize forward pointer
               60    80    DE  1EAE       MOVAL   (R0)+,(R0)              ; Initialize backward pointer
                                1EB1
E620 CF    20    18 AC    83  1EB1       SUBB3   PRI(AP),#32,IO_PRI      ; Set the IO priority to internal value
                                1EB8
                                1EB8  ;
                                1EB8  ; Set some miscellaneous values
                                1EB8  ;
          50    20 AC    D0  1EB8  30$: MOVL    FRA(AP),R0              ; Set the fractional buffer per I/O
     0000FE00 8F    50    D1  1EBC       CMPL    R0,#127*512             ; Is it bigger than 127 block?
                    07    1B  1EC3       BLEQU   32$                     ; No, all is well
     50    0000FE00 8F    D0  1EC5       MOVL    #127*512,R0             ; Biggest fraction is 127 block
          E18B CF    50    D0  1ECC  32$: MOVL    R0,FRACTION             ; Must be I/O viable on any disk
               50    1C AC    D0  1ED1       MOVL    SMA(AP),R0              ; Set the smallest fragment per I/O
          E17A CF    50    D1  1ED5       CMPL    R0,MAX_BUF              ; SMALL may not be larger than MAX_BUF
                    05    1B  1EDA       BLEQU   35$                     ; Branch if less than MAX_BUF
          50   E174 CF    D0  1EDC       MOVL    MAX_BUF,R0              ; Use MAX_BUF
          E172 CF    50    D0  1EE1  35$: MOVL    R0,SMALL                ; Set the smallest fragment per I/O
     E2B0 CF    14 AC    B0  1EE6       MOVW    TIM(AP),CTRL_INFO+4     ; Set the controller timeout
          00000000'GF          D0  1EEC       MOVL    G^SCS$GB_SYSTEMID,-
               E29B CF              1EF2               CTRL_ID                 ; Set the controller id
          00000004'GF          B0  1EF5       MOVW    G^SCS$GB_SYSTEMID+4,-
               E296 CF              1EFB               CTRL_ID+4
                                1EFE  ;
                                1EFE  ; Establish link with PORT_DRIVER
                                1EFE  ;
                                1EFE          SETIPL  #IPL$_SCS               ; Coordinate activities
                                1F01          LISTEN  MSGADR=LIS_FORK,-       ; Establish ourselves as listener
                                1F01                  ERRADR=LIS_ERR,-
                                1F01                  LPRNAM=SERV_NAME,-
                                1F01                  PRINFO=SERV_INFO
          FDC9 CF    53    D0  1F1A       MOVL    R3,LIS_CDT              ; Store the CDT away
               50    01    D0  1F1F       MOVL    #SS$_NORMAL,R0          ; Set success
                                1F22  40$: SETIPL  #0                      ; Drop IPL
                          05  1F25       RSB                             ; Leave, finished initialization
                                1F26
                                1F26          .DISABLE        LSB
                                1F26
               00001F26        1F26          LC=.
               00001F30        1F26          .=<LC+15>&-16
                                1F30
                                1F30  MSCP_END:
                                1F30          .END
```

```
ABORT                     00000998 R     02
ABO_ER                    0000147D R     02
ABX_ER                    0000147F R     02
ACCESS                    00001379 R     02
ADD                       000004FE R     02
ALLOMEM                   00000654 R     02
ATTN_CLEANUP              0000056B R     02
AVAIC                     00000C09 R     02
AVAIL_ALL                 00000D47 R     02
AV_CASE2                  00000CFE R     02
AV_CASE2X                 00000D05 R     02
AV_CASE3                  00000C8D R     02
BACK                      00001B3F R     02
BIG_CDRP                = 000000E0
BLK_COUNT                 000002B0 R     02
BLOCKEDH                  00000B40 R     02
BLOCKEDT                  00000B47 R     02
BLOCK_CDRP                000012BB R     02
BUF                     = 00000008
BUFF_ADDR                 000004D0 R     02
BUFF_END                  000004D4 R     02
BUFF_HEAD                 00000050 R     02
BUFF_LENGTH               000004CC R     02
BUG$_MSCPSERV             ********     X 02
CASE2A                    00000EBF R     02
CASE2A_DONE               00001059 R     02
CASE2A_ERR                00000F1B R     02
CASE2A_ERR_OUT            00000F20 R     02
CCHAR                     00000EBC R     02
CCX                       00001060 R     02
CDRP$B_CD_TYPE          = 0000000A
CDRP$B_EFN              = FFFFFFC2
CDRP$B_FIPL             = 0000000B
CDRP$B_FLAGS            = FFFFFFFA
CDRP$B_IRP_TYPE         = FFFFFFAA
CDRP$B_LBUFF            = 00000068
CDRP$B_OPCODE           = FFFFFFF8
CDRP$B_PRI              = FFFFFFC3
CDRP$B_SHDW_CNT         = FFFFFFC2
CDRP$K_CD_LEN           = 00000048
CDRP$K_ST_BLK_QUE       = 00000007
CDRP$K_ST_DATA_WAIT     = 00000003
CDRP$K_ST_DRVRD_WAIT    = 00000002
CDRP$K_ST_DRVWRT_WAIT   = 00000004
CDRP$K_ST_INVALID       = 0000003F
CDRP$K_ST_MAP_WAIT      = 00000001
CDRP$K_ST_MEM_WAIT      = 00000000
CDRP$K_ST_MSGT_WAIT     = 00000009
CDRP$K_ST_MSG_WAIT      = 00000005
CDRP$K_ST_ONBCK_QUE     = 00000008
CDRP$K_ST_SHQ_QUE       = 00000006
CDRP$L_ABCNT            = FFFFFFE0
CDRP$L_ARB              = FFFFFFF8
CDRP$L_AST              = FFFFFFB0
CDRP$L_ASTPRM           = FFFFFFB4
CDRP$L_BCNT             = FFFFFFD2
CDRP$L_BUFADR           = FFFFFFB8

CDRP$L_BUFLEN           = FFFFFFB4
CDRP$L_CDT              = 00000024
CDRP$L_CDTX             = 00000060
CDRP$L_CMD_STS          = FFFFFFF4
CDRP$L_DIAGBUF          = FFFFFFEC
CDRP$L_EXTEND           = FFFFFFF4
CDRP$L_FPC              = 0000000C
CDRP$L_FPCX             = 00000050
CDRP$L_FQBL             = 00000004
CDRP$L_FQFL             = 00000000
CDRP$L_FR3              = 00000010
CDRP$L_FR3X             = 00000054
CDRP$L_FR4X             = 00000058
CDRP$L_HQB              = 0000007C
CDRP$L_IOQBL            = FFFFFFA4
CDRP$L_IOQFL            = FFFFFFA0
CDRP$L_IOSB             = FFFFFFC4
CDRP$L_IOST1            = FFFFFFD8
CDRP$L_LBN              = 00000048
CDRP$L_LBOFF            = 00000030
CDRP$L_LBUFH_AD         = 0000002C
CDRP$L_MSGBUFX          = FFFFFFB0
CDRP$L_MSG_BUF          = 0000001C
CDRP$L_MST_UQB          = 00000078
CDRP$L_OBCNT            = FFFFFFE4
CDRP$L_PID              = FFFFFFAC
CDRP$L_RBOFF            = 00000038
CDRP$L_RBUFH_AD         = 00000034
CDRP$L_RSPID            = 00000020
CDRP$L_RWCPTR           = 00000028
CDRP$L_SAVD_RTN         = 00000018
CDRP$L_SAVD_RTX         = 0000005C
CDRP$L_SEGVBN           = FFFFFFE8
CDRP$L_SHDW_LNK         = 0000004C
CDRP$L_STRT_UQB         = 00000074
CDRP$L_SVAPTE           = FFFFFFCC
CDRP$L_UCB              = FFFFFFBC
CDRP$L_UQB              = FFFFFFC4
CDRP$L_UQB_BL           = FFFFFFEC
CDRP$L_UQB_FL           = FFFFFFE8
CDRP$L_WIND             = FFFFFFB8
CDRP$L_XCT_LEN          = 0000003C
CDRP$L_XCT_LENX         = 00000064
CDRP$V_ABORT            = 00000004
CDRP$V_BUFFER           = 00000001
CDRP$V_NOMSTBUF         = 00000005
CDRP$V_SLAVE            = 00000000
CDRP$W_BOFF             = FFFFFFD0
CDRP$W_CDRPSIZE         = 00000008
CDRP$W_FUNC             = FFFFFFC0
CDRP$W_IRP_SIZE         = FFFFFFA8
CDRP$W_SLOT             = FFFFFFF8
CDRP$W_STATE            = FFFFFFD6
CDRP$W_STS              = FFFFFFCA
CDRP_LIST                 00000070 R     02
CDRP_SAVE                 0000006C R     02
CDSV_SIZE                 00000068 R     02
```

ABORT	00000998	R	02
ABO_ER	0000147D	R	02
ABX_ER	0000147F	R	02
ACCESS	00001379	R	02
ADD	000004FE	R	02
ALLOMEM	00000654	R	02
ATTN_CLEANUP	0000056B	R	02
AVAIL	00000C09	R	02
AVAIL_ALL	00000D47	R	02
AV_CASE2	00000CFE	R	02
AV_CASE2X	00000D05	R	02
AV_CASE3	00000CBD	R	02
BACK	00001B3F	R	02
BIG_CDRP	= 000000E0		
BLK_COUNT	000002B0	R	02
BLOCKEDH	00000840	R	02
BLOCKEDT	00000847	R	02
BLOCK_CDRP	000012B8	R	02
BUF	= 00000008		
BUFF_ADDR	000004D0	R	02
BUFF_END	000004D4	R	02
BUFF_HEAD	00000050	R	02
BUFF_LENGTH	000004CC	R	02
BUG$_MSCPSERV	********	X	02
CASE2A	00000EBF	R	02
CASE2A_DONE	00001059	R	02
CASE2A_ERR	00000F1B	R	02
CASE2A_ERR_OUT	00000F20	R	02
CCHAR	00000EBC	R	02
CCX	00001060	R	02
CDRP$B_CD_TYPE	= 0000000A		
CDRP$B_EFN	= FFFFFFC2		
CDRP$B_FIPL	= 0000000B		
CDRP$B_FLAGS	= FFFFFFFA		
CDRP$B_IRP_TYPE	= FFFFFFAA		
CDRP$B_LBUFF	= 00000068		
CDRP$B_OPCODE	= FFFFFFFB		
CDRP$B_PRI	= FFFFFFC3		
CDRP$B_SHDW_CNT	= FFFFFFC2		
CDRP$K_CD_LEN	= 00000048		
CDRP$K_ST_BLK_QUE	= 00000007		
CDRP$K_ST_DATA_WAIT	= 00000003		
CDRP$K_ST_DRVRD_WAIT	= 00000002		
CDRP$K_ST_DRVWRT_WAIT	= 00000004		
CDRP$K_ST_INVALID	= 0000003F		
CDRP$K_ST_MAP_WAIT	= 00000001		
CDRP$K_ST_MEM_WAIT	= 00000000		
CDRP$K_ST_MSGT_WAIT	= 00000009		
CDRP$K_ST_MSG_WAIT	= 00000005		
CDRP$K_ST_ONBLK_QUE	= 00000008		
CDRP$K_ST_SHQ_QUE	= 00000006		
CDRP$L_ABCNT	= FFFFFFE0		
CDRP$L_ARB	= FFFFFFF8		
CDRP$L_AST	= FFFFFFB0		
CDRP$L_ASTPRM	= FFFFFFB4		
CDRP$L_BCNT	= FFFFFFD2		
CDRP$L_BUFADR	= FFFFFFB8		
CDRP$L_BUFLEN	= FFFFFFB4		
CDRP$L_CDT	= 00000024		
CDRP$L_CDTX	= 00000060		
CDRP$L_CMD_STS	= FFFFFFF4		
CDRP$L_DIAGBUF	= FFFFFFEC		
CDRP$L_EXTEND	= FFFFFFF4		
CDRP$L_FPC	= 0000000C		
CDRP$L_FPCX	= 00000050		
CDRP$L_FQBL	= 00000004		
CDRP$L_FQFL	= 00000000		
CDRP$L_FR3	= 00000010		
CDRP$L_FR3X	= 00000054		
CDRP$L_FR4X	= 00000058		
CDRP$L_HQB	= 0000007C		
CDRP$L_IOQBL	= FFFFFFA4		
CDRP$L_IOQFL	= FFFFFFA0		
CDRP$L_IOSB	= FFFFFFC4		
CDRP$L_IOST1	= FFFFFFD8		
CDRP$L_LBN	= 00000048		
CDRP$L_LBOFF	= 00000030		
CDRP$L_LBUFH_AD	= 0000002C		
CDRP$L_MSGBUFX	= FFFFFFB0		
CDRP$L_MSG_BUF	= 0000001C		
CDRP$L_MST_UQB	= 00000078		
CDRP$L_OBCNT	= FFFFFFE4		
CDRP$L_PID	= FFFFFFAC		
CDRP$L_RBOFF	= 00000038		
CDRP$L_RBUFH_AD	= 00000034		
CDRP$L_RSPID	= 00000020		
CDRP$L_RWCPTR	= 00000028		
CDRP$L_SAVD_RTN	= 00000018		
CDRP$L_SAVD_RTX	= 0000005C		
CDRP$L_SEGVBN	= FFFFFFE8		
CDRP$L_SHDW_LNK	= 0000004C		
CDRP$L_STRT_UQB	= 00000074		
CDRP$L_SVAPTE	= FFFFFFCC		
CDRP$L_UCB	= FFFFFFBC		
CDRP$L_UQB	= FFFFFFC4		
CDRP$L_UQB_BL	= FFFFFFEC		
CDRP$L_UQB_FL	= FFFFFFE8		
CDRP$L_WIND	= FFFFFFB8		
CDRP$L_XCT_LEN	= 0000003C		
CDRP$L_XCT_LENX	= 00000064		
CDRP$V_ABORT	= 00000004		
CDRP$V_BUFFER	= 00000001		
CDRP$V_NOHSTBUF	= 00000005		
CDRP$V_SLAVE	= 00000000		
CDRP$W_BOFF	= FFFFFFD0		
CDRP$W_CDRPSIZE	= 00000008		
CDRP$W_FUNC	= FFFFFFC0		
CDRP$W_IRP_SIZE	= FFFFFFA8		
CDRP$W_SLOT	= FFFFFFF8		
CDRP$W_STATE	= FFFFFFD6		
CDRP$W_STS	= FFFFFFCA		
CDRP_LIST	00000070	R	02
CDRP_SAVE	0000006C	R	02
CDSV_SIZE	00000068	R	02

MS
V0

```
CDT$B_RSTATION          = 00000020
CDT$L_AUXSTRUC          = 0000005C
CDT$L_LCONID            = 00000018
CDT$L_PDT               = 00000010
CHECK_ONLINE              00000DF5 R     02
CLEAN_UP                  00000B21 R     02
CLEAN_UP_ONLCPY           0000128C R     02
CLU$GL_ALLOCLS            ********   X   02
COMP_CTRL_DAT             00001384 R     02
COMP_HOST_DAT             000013DB R     02
COM_PKT_LEN               00001CFC R     02
COPY_CHAR                 00001314 R     02
CTRL_DAT                  000001A0 R     02
CTRL_ID                   00000190 R     02
CTRL_INFO                 00000198 R     02
DDT$L_FDT               = 00000008
DEALLOC_BUFF              000012CE R     02
DEALMEM                   0000064E R     02
DEALRO                    0000064B R     02
DEF_RTN                 = 00000000
DET_ACC_PATH              0000133F R     02
DEV$V_RCT               = 00000008
DG_ERR                    0000065A R     02
DISCON                    00000576 R     02
DO_DISK                   00001A5D R     02
DO_DISKA                  00001A61 R     02
DO_DISKX                  00001A74 R     02
DO_WRT                    000016FF R     02
DRAIN_BLOCKQ              00000B75 R     02
DYN$C_BUFIO             = 00000013
DYN$C_CDRP              = 00000039
DYN$C_IRP               = 0000000A
DYN$C_LC_MSCP           = 00000008
DYN$C_LOADCODE          = 00000062
DYN$C_SCS               = 00000060
DYN$C_SCS_HQB           = 0000000B
DYN$C_SCS_UQB           = 0000000A
EMPTY_SHQ                 000011C8 R     02
END_PKT_LEN               00001D24 R     02
ERASE                     00001387 R     02
ERR_1                     00000929 R     02
ERR_2                     0000092F R     02
ERR_3                     00000933 R     02
ERR_AVAIL                 00001544 R     02
ERR_OFFLINE               0000154D R     02
ERR_SHACHA                00001560 R     02
ERR_TBL                   000014F6 R     02
ERR_WRITLCK               00001556 R     02
ERR_X                     00000937 R     02
EXE$ALLOCATE              ********   X   02
EXE$ALONONPAGED           ********   X   02
EXE$DEALLOCATE            ********   X   02
EXE$DEANONPAGED           ********   X   02
EXE$INSIOQ                ********   X   02
FIND_SLOT                 0000095C R     02
FINI                      0000143C R     02
FIN_HQB                   0000079F R     02
FIN_UNIT_STS              00000A8B R     02
FLUSH                     0000139F R     02
FRA                     = 00000020
FRACTION                  0000005C R     02
FREE_STRUCT               00001C98 R     02
GET_CMD_STS               00000A02 R     02
GET_MAX_BLOCK             0000129D R     02
GET_STRUCT                00001C39 R     02
GET_UNIT_STS              00000A14 R     02
HOS                     = 00000010
HOSTSET                 = 00001C47
HOST_DSBL                 00000088 R     02
HOST_VEC                  00000110 R     02
HQB$B_FLAG                0000001F
HQB$B_PORT                0000000B
HQB$B_TYPE                0000000A
HQB$B_VEC                 0000001E
HQB$C_LEN                 00000048
HQB$K_LEN                 00000048
HQB$L_BLINK               00000004
HQB$L_CDT                 00000014
HQB$L_FLINK               00000000
HQB$L_UNIT_ONLN           00000010
HQB$Q_PENDING             00000030
HQB$Q_TIME                00000020
HQB$W_CNT_FLGS            00000018
HQB$W_HST_TMO             0000001A
HQB$W_MAX_QUE             0000002A
HQB$W_NUM_QUE             00000028
HQB$W_REFC                0000002C
HQB$W_SIZE                00000008
HQB$W_USE_FRAC            0000001C
HQBLIST                   000004B4 R     02
IMMED                     00000989 R     02
INI_HOST                  0000004C R     02
INI_PKT                   00000046 R     02
INV_FUNC                  00001A70 R     02
IO$M_DATACHECK          = 00004000
IO$M_FORCERR            = 00000100
IO$M_INHERLOG           = 00000800
IO$M_INHRETRY           = 00008000
IO$S_FCODE              = 00000006
IO$V_FCODE              = 00000000
IO$_AVAILABLE           = 00000011
IO$_DSE                 = 00000015
IO$_PACKACK             = 00000008
IO$_READPBLK            = 0000000C
IO$_UNLOAD              = 00000001
IO$_WRITECHECK          = 0000000A
IO$_WRITEPBLK           = 0000000B
IOC$CVTLOGPHY             ********   X   02
IO_PRI                    000004D8 R     02
IPL$_SCS                = 00000008
IRP$B_FLAGS             = 0000005A
IRP$B_SHDW_CNT          = 00000022
IRP$C_LENGTH            = 000000C4
IRP$L_BCNT              = 00000032
```

IRP$L_FPCX	= 000000B0		
IRP$L_FR4X	= 000000B8		
IRP$L_IOST1	= 00000038		
IRP$L_LBN	= 000000A8		
IRP$L_MAINCDRP	= 000000B4		
IRP$L_PID	= 0000000C		
IRP$L_SEQNUM	= 00000050		
IRP$L_SHDW_LNK	= 000000AC		
IRP$L_SVAPTE	= 0000002C		
IRP$L_UCB	= 0000001C		
IRP$L_UQB	= 00000024		
IRP$M_PHYSIO	= 00000100		
IRP$V_PHYSIO	= 00000008		
IRP$W_FUNC	= 00000020		
IRP$W_SIZE	= 00000008		
IRP$W_SLOT	= 00000058		
IRP$W_STATE	= 00000036		
IRP$W_STS	= 0000002A		
LC	= 00001F26	R	02
LIS_CDT	00001CE8	R	02
LIS_ERR	00000663	R	02
LIS_FORK	00000585	R	02
MAINT	000013A5	R	02
MAKE_AVAIL	00000C3B	R	02
MAX_BUF	00000054	R	02
MAX_PKT	0000004A	R	02
MAX_QUE	00000062	R	02
MEM_WAIT	00000078	R	02
MMG$GL_SPTBASE	********	X	02
MOD_TBL	00001D4C	R	02
MSCP$B_BUFFER	= 00000010		
MSCP$B_CNT_ALCS	= 00000004		
MSCP$B_OPCODE	= 00000008		
MSCP$B_RBNS	= 0000002E		
MSCP$B_RCT_CPYS	= 0000002F		
MSCP$K_CL_CNTRL	= 00000001		
MSCP$K_CM_EMULA	= 00000004		
MSCP$K_MIN_SIZE	= 0000000C		
MSCP$K_OP_ABORT	= 00000001		
MSCP$K_OP_ACCES	= 00000010		
MSCP$K_OP_AVAIL	= 00000008		
MSCP$K_OP_COMP	= 00000020		
MSCP$K_OP_DTACP	= 0000000B		
MSCP$K_OP_END	= 00000080		
MSCP$K_OP_ONLIN	= 00000009		
MSCP$K_OP_WRITE	= 00000022		
MSCP$K_SC_ALONL	= 00000008		
MSCP$K_SC_UNKNO	= 00000000		
MSCP$K_ST_ABRTD	= 00000002		
MSCP$K_ST_AVLBL	= 00000004		
MSCP$K_ST_CNTLR	= 0000000A		
MSCP$K_ST_COMP	= 00000007		
MSCP$K_ST_DATA	= 00000008		
MSCP$K_ST_DRIVE	= 0000000B		
MSCP$K_ST_HSTBF	= 00000009		
MSCP$K_ST_ICMD	= 00000001		
MSCP$K_ST_MFMTE	= 00000005		
MSCP$K_ST_OFFLN	= 00000003		
MSCP$K_ST_SBCOD	= 00000020		
MSCP$K_ST_SHST	= 0000000C		
MSCP$K_ST_SUCC	= 00000000		
MSCP$K_ST_WRTPR	= 00000006		
MSCP$L_BYTE_CNT	= 0000000C		
MSCP$L_CMD_REF	= 00000000		
MSCP$L_CMD_STS	= 00000010		
MSCP$L_EXCL_LBA	= 00000014		
MSCP$L_FRST_BAD	= 0000001C		
MSCP$L_LBN	= 0000001C		
MSCP$L_MEDIA_ID	= 0000001C		
MSCP$L_OUT_REF	= 0000000C		
MSCP$L_UNT_SIZE	= 00000024		
MSCP$L_VOL_SER	= 00000028		
MSCP$M_CF_MLTHS	= 00000004		
MSCP$M_CF_REPLC	= 00008000		
MSCP$M_MD_ALLCD	= 00000001		
MSCP$M_MD_CLSEX	= 00002000		
MSCP$M_MD_COMP	= 00004000		
MSCP$M_MD_ERROR	= 00001000		
MSCP$M_MD_EXPRS	= 00008000		
MSCP$M_MD_FLENU	= 00000001		
MSCP$M_MD_IGNMF	= 00000002		
MSCP$M_MD_NXUNT	= 00000001		
MSCP$M_MD_PRIMR	= 00000001		
MSCP$M_MD_RIP	= 00000001		
MSCP$M_MD_SCCHH	= 00000800		
MSCP$M_MD_SCCHL	= 00000400		
MSCP$M_MD_SECOR	= 00000200		
MSCP$M_MD_SEREC	= 00000100		
MSCP$M_MD_SHDSP	= 00000010		
MSCP$M_MD_SPNDW	= 00000002		
MSCP$M_MD_SSHDW	= 00000080		
MSCP$M_MD_STWRP	= 00000004		
MSCP$M_MD_VOLTL	= 00000002		
MSCP$M_MD_WBKNV	= 00000040		
MSCP$M_MD_WBKVL	= 00000020		
MSCP$M_MD_WRSEQ	= 00000010		
MSCP$M_UF_576	= 00000004		
MSCP$M_UF_CMPRD	= 00000001		
MSCP$M_UF_CMPWR	= 00000002		
MSCP$M_UF_REPLC	= 00008000		
MSCP$M_UF_RMVBL	= 00000080		
MSCP$M_UF_SCCHH	= 00000800		
MSCP$M_UF_SCCHL	= 00000400		
MSCP$M_UF_SSMEM	= 00004000		
MSCP$M_UF_SSMST	= 00000200		
MSCP$M_UF_WBKNV	= 00000040		
MSCP$M_UF_WRTPH	= 00002000		
MSCP$M_UF_WRTPS	= 00001000		
MSCP$Q_CNT_ID	= 00000014		
MSCP$Q_TIME	= 00000014		
MSCP$Q_UNIT_ID	= 00000014		
MSCP$V_MD_COMP	= 0000000E		
MSCP$V_MD_NXUNT	= 00000000		
MSCP$V_MD_SEREC	= 00000008		

```
MSCP$V_MD_SHDSP                  = 00000004
MSCP$V_MD_SPNDW                  = 00000001
MSCP$V_MD_STWRP                  = 00000002
MSCP$V_UF_SSMEM                  = 0000000E
MSCP$V_UF_SSMST                  = 00000009
MSCP$V_UF_WRTPH                  = 0000000D
MSCP$V_UF_WRTPS                  = 0000000C
MSCP$W_CNT_FLGS                  = 0000000E
MSCP$W_COPY_SPD                  = 00000022
MSCP$W_CYLINDER                  = 00000028
MSCP$W_EXCL_LBC                  = 00000018
MSCP$W_GROUP                     = 00000026
MSCP$W_HST_TMO                   = 00000010
MSCP$W_MODIFIER                  = 0000000A
MSCP$W_MULT_UNT                  = 0000000C
MSCP$W_RCT_SIZE                  = 0000002C
MSCP$W_SHDW_STS                  = 00000022
MSCP$W_SHDW_UNT                  = 00000020
MSCP$W_STATUS                    = 0000000A
MSCP$W_TRACK                     = 00000024
MSCP$W_UNIT                      = 00000004
MSCP$W_UNT_FLGS                  = 0000000E
MSCP$W_VERSION                   = 0000000C
MSCP_BEGIN                         00000000 R     02
MSCP_COMM_AREA                     00000000 R     02
MSCP_END                           00001F30 R     02
MSCP_START                         000004D9 R     02
MSG_IN                             00000876 R     02
MST_UQB                          = 0000002C
MSU$B_CTRL_DAT                     000001A0
MSU$B_SUBTYPE                      0000000B
MSU$B_TYPE                         0000000A
MSU$B_VEC                          00000024
MSU$C_LENGTH                       000004CC
MSU$C_NUMQUE                     = 00000003
MSU$K_AC_ADD                     = 00000003
MSU$K_AC_DISC                    = 00000005
MSU$K_AC_REM                     = 00000004
MSU$K_AC_STOP                    = 00000002
MSU$K_AC_STRT                    = 00000001
MSU$K_LENGTH                       000004CC
MSU$K_MAX_HOST                   = 00000010
MSU$K_MAX_OPC                    = 00000040
MSU$K_MAX_UNIT                   = 00000010
MSU$K_UNIT_SIZ                   = 00000004
MSU$L_ABORT                        000001B4
MSU$L_AVAIL                        000001D0
MSU$L_BLKCOUNT                     000002B0
MSU$L_BUFF_HEAD                    00000050
MSU$L_CDRP_LIST                    00000070
MSU$L_CDRP_SAVE                    0000006C
MSU$L_CDSV_SIZE                    00000068
MSU$L_D1                           00000000
MSU$L_D2                           00000004
MSU$L_FRACTION                     0000005C
MSU$L_GETCMD                       000001B8
MSU$L_GETUNIT                      000001BC
MSU$L_HOST_DSBL                    00000088
MSU$L_HOST_VEC                     00000110
MSU$L_MAX_BUF                      00000054
MSU$L_MEM_WAIT                     00000078
MSU$L_ONLINE                       000001D4
MSU$L_OPCOUNT                      000001B0
MSU$L_P1                           0000000C
MSU$L_P2                           00000010
MSU$L_READ                         00000234
MSU$L_SETCTRL                      000001C0
MSU$L_SETUNIT                      000001D8
MSU$L_SMALL                        00000058
MSU$L_STATE                        00000040
MSU$L_UNIT_VEC                     00000090
MSU$L_UQB_LIST                     00000080
MSU$L_WRITE                        00000238
MSU$Q_CTRL_ID                      00000190
MSU$Q_CTRL_INFO                    00000198
MSU$Q_HQBLIST                      000004B4
MSU$Q_QUEUE                        00000070
MSU$T_NAME                         00000014
MSU$V_START                      = 0000001F
MSU$W_INI_HOST                     0000004C
MSU$W_INI_PKT                      00000046
MSU$W_MAX_PKT                      0000004A
MSU$W_MAX_QUE                      00000062
MSU$W_MIN_PKT                      0000004A
MSU$W_NUM_HOST                     0000004E
MSU$W_NUM_PKT                      00000048
MSU$W_NUM_QUE                      00000060
MSU$W_PACKET                       00000044
MSU$W_SIZE                         00000008
NEW_UQB                            00000F80 R     02
NONHOSTSET                       = 0000E280
NOT_FOUND                          0000094D R     02
NSEQ                               00001342 R     02
NSEQB                              000013A8 R     02
NUM_HOST                           0000004E R     02
NUM_PKT                            00000048 R     02
NUM_QUE                            00000060 R     02
OLD_UQB                            00000F25 R     02
ONLINE                             00000D77 R     02
ONL_ABORT_ERR                      00001269 R     02
ONL_COPY                           00001065 R     02
ONL_FINI                           000011E1 R     02
ONL_FINIX                          000011E3 R     02
ONL_LOOP                           000010CF R     02
ONL_LOOP_CHK                       00001143 R     02
ONL_READ                           000010E2 R     02
ONL_READ_ERR                       00001209 R     02
ONL_WRITE                          00001121 R     02
ONL_WRITE_ERR                      00001249 R     02
OPC_COUNT                          000001B0 R     02
OUT                                000014AC R     02
OUT1                               000014AE R     02
OUT_ER                             00001484 R     02
PAC                              = 0000000C
```

```
PACKET                         00000044 R      02     SERV_NAME                      00001CEC R      02
PATCH                          00001DA0 R      02     SETUP                          000017BF R      02
PDT$B_PDT_TYPE               = 00000007               SETUP_ERR                      00001A52 R      02
PDT$C_PS                     = 00000004               SETUP_OUT                      00001936 R      02
PDT$L_ALLOCMSG               = 00000014               SETUP_SHADOW                   0000193F R      02
PDT$L_DEALLOMSG              = 00000020               SETUP_SHADOW_READ              00001A1F R      02
PDT$L_DEALRGMSG              = 00000024               SETUP_SHADOW_WRITE             00001948 R      02
PDT$L_MAPIRP                 = 00000034               SETX                           00000C9D R      02
PDT$L_RCLMSGBUF              = 00000048               SET_CTRL_CHR                   00000AA0 R      02
PDT$L_REJECT                 = 0000004C               SET_UNIT_CHR                   000012FC R      02
PDT$L_REQDATA                = 00000050               SLV$K_SDATA                  = 00000004
PDT$L_SENDDATA               = 00000054               SMA                          = 0000001C
PDT$L_SNDCNTMSG              = 00000060               SMALL                          00000058 R      02
PDT$L_UNMAP                  = 00000064               SS                             00000A11 R      02
PMS$END_RQ                     ********   X    02     SS$_BADPARAM                 = 00000014
PMS$GL_IOPFMPDB                ********   X    02     SS$_CTRLERR                  = 00000054
PMS$GL_IOPFMSEQ                ********   X    02     SS$_DATACHECK                = 0000005C
PMS$START_RQ                   ********   X    02     SS$_FORCEDERROR              = 00002144
PR$_IPL                      = 00000012               SS$_FORMAT                   = 000000BC
PRI                          = 00000018               SS$_ILLIOFUNC                = 000000F4
PRMSW                        = 00000001               SS$_ILLSEQOP                 = 000002DC
PRT$C_ER                     = 00000007               SS$_INSFMEM                  = 00000124
PRT$C_EW                     = 00000005               SS$_IVBUFLEN                 = 0000034C
QUEUES                         00000070 R      02     SS$_MEDOFL                   = 000001A4
QUE_NUM                      = 00000003               SS$_NORMAL                   = 00000001
READ                           000013E1 R      02     SS$_PARITY                   = 000001F4
READ_CONT                      0000140B R      02     SS$_SHACHASTA                = 00002284
READ_ERR                       0000156F R      02     SS$_TIMEOUT                  = 0000022C
READ_INVALID_CDT               00001407 R      02     SS$_VOLINV                   = 00000254
READ_LOOP                      000013F4 R      02     SS$_WRITLCK                  = 0000025C
READ_REPAIR                    000015AF R      02     START                          000004EC R      02
RED_UQB                      = 00000028               START_CONT                     00001E28 R      02
REMOVE                         00000575 R      02     STATE                          00000040 R      02
REPLACE                        000013A2 R      02     STOP                           0000065E R      02
RSBX                           00000669 R      02     ST_BLK_QUE                     00000752 R      02
SCS$ACCEPT                     ********  GX    02     ST_DAT_WAIT                    0000070E R      02
SCS$ALLOC_RSPID                ********   X    02     ST_DRV_WAIT                    000006F4 R      02
SCS$DEALL_RSPID                ********   X    02     ST_MAP_WAIT                    0000073F R      02
SCS$DISCONNECT                 ********   X    02     ST_MEM_WAIT                    0000073B R      02
SCS$GB_SYSTEMID                ********   X    02     ST_MSG_WAIT                    0000073F R      02
SCS$GL_MSCP                    ********   X    02     ST_ONBLK_QUE                   00000765 R      02
SCS$LISTEN                     ********  GX    02     ST_SHQ_QUE                     00000744 R      02
SCS$LKP_MSGWAIT                ********   X    02     TIM                          = 00000014
SCS$LKP_RDTCDRP                ********   X    02     TST_END                        00001445 R      02
SCS$LKP_RDTWAIT                ********   X    02     UCB$B_ONLCNT                 = 000000AE
SCS$L_DST_CONID              = FFFFFFF8               UCB$B_SECTORS                = 00000044
SEARCH_QUE                     00001BB8 R      02     UCB$B_TRACKS                 = 00000045
SELECT_FIRST_UNIT              00001A3D R      02     UCB$L_DDT                    = 00000088
SELECT_NEXT_UNIT               00001A34 R      02     UCB$L_DEVCHAR                = 00000038
SEND_ATTN                      00000508 R      02     UCB$L_MAXBLOCK               = 000000B0
SEND_END                       00000AED R      02     UCB$L_STS                    = 00000064
SEND_MISMATCH                  00000A9C R      02     UCB$V_MNTVERIP               = 0000000E
SEND_PKT                       00000AD4 R      02     UCB$W_QLEN                   = 0000006A
SEND_PKT_STS                   00000ADE R      02     UNIT_VEC                       00000090 R      02
SEND_SUCCESS                   00000AD2 R      02     UQB$B_CMD_PEND                 0000000D
SEQ                            00000BA0 R      02     UQB$B_FLAGS                    0000002B
SERV_INFO                      00001CEC R      02     UQB$B_HOST_CNT                 0000000C
```

```
UQB$B_SLOT                0000002A
UQB$B_TYPE                0000000A
UQB$C_LEN                 0000009C
UQB$K_LEN                 0000009C
UQB$L_CDRP_BL             00000064
UQB$L_CDRP_FL             00000060
UQB$L_CPY_CDRP            00000088
UQB$L_FENCEH              0000005C
UQB$L_FENCEL              00000058
UQB$L_HOST_ONLN           00000034
UQB$L_MAXBLOCK            00000048
UQB$L_MEDIA_ID            00000020
UQB$L_SHDW_BL             00000004
UQB$L_SHDW_FL             00000000
UQB$L_SHDW_LOW            00000078
UQB$L_SHDW_MST            00000078
UQB$L_START_LBN           0000004C
UQB$L_UCB                 00000030
UQB$L_VOL_SER             0000002C
UQB$M_2PASS             = 00000004
UQB$M_BLOCKED           = 00000002
UQB$M_CIP               = 00000001
UQB$M_CMDPEND           = 00000008
UQB$M_RIP               = 00000020
UQB$M_SEQ               = 00000010
UQB$Q_BLOCKQ              00000068
UQB$Q_SHQ                 00000070
UQB$Q_UNITQ               00000080
UQB$Q_UNIT_ID             00000018
UQB$T_NAME                00000038
UQB$V_2PASS             = 00000002
UQB$V_CIP               = 00000000
UQB$V_CMDPEND           = 00000003
UQB$V_RIP               = 00000005
UQB$W_MAX_BLK             0000007E
UQB$W_MAX_OPS             00000056
UQB$W_MAX_QUE             00000052
UQB$W_MULT_UNT            00000010
UQB$W_NUM_BLK             0000007C
UQB$W_NUM_OPS             00000054
UQB$W_NUM_QUE             00000050
UQB$W_SHDW_STS            00000026
UQB$W_SHDW_UNT            00000024
UQB$W_SIZE                00000008
UQB$W_STATUS              0000000E
UQB$W_UNIT                00000028
UQB$W_UNT_FLGS            00000012
UQB_LIST                  00000080 R      02
VA$S_BYTE               = 00000009
VA$S_VPN                = 00000015
VA$V_VPN                = 00000009
VC_ERR                    0000066A R      02
VERSION                   00000198 R      02
WAIT_FOR_INT              00000863 R      02
WRITE                     000016EE R      02
WRITE_ERR                 00001787 R      02
WRT_UQB                 = 00000024
```

```
                                +----------------+
                                ! Psect synopsis !
                                +----------------+

PSECT name                      Allocation            PSECT No.  Attributes
----------                      ----------            ---------  ----------
.  ABS  .                       00000000  (     0.)   00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           000004CC  (  1228.)   01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
MSCP$CODE                       00001F30  (  7984.)   02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC PAGE

                              +------------------------+
                              ! Performance indicators !
                              +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                      9    00:00:00.05    00:00:02.49
Command processing                 85    00:00:00.73    00:00:08.85
Pass 1                            732    00:00:36.97    00:01:53.01
Symbol table sort                   0    00:00:04.62    00:00:15.83
Pass 2                            418    00:00:10.09    00:00:36.60
Symbol table output                 0    00:00:00.52    00:00:01.62
Psect synopsis output               0    00:00:00.02    00:00:00.29
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             1244    00:00:53.00    00:02:58.69

The working set limit was 2550 pages.
202644 bytes (396 pages) of virtual memory were used to buffer the intermediate code.
There were 160 pages of symbol table space allocated to hold 2769 non-local and 285 local symbols.
3988 source lines were read in Pass 1, producing 31 object records in Pass 2.
57 pages of virtual memory were used to define 54 macros.

                             +--------------------------+
                             ! Macro library statistics !
                             +--------------------------+

Macro library name                         Macros defined
------------------                         --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       33
_$255$DUA28:[MSCP.OBJ]MSCPDEF.MLB;1                   3
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    10
TOTALS (all libraries)                               46

2911 GETS were required to define 46 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:MSCP/OBJ=OBJ$:MSCP MSRC$:MSCP/UPDATE=(ENH$:MSCP)+LIB$:MSCPDEF/LIB+EXECML$/LIB
```

0249 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

MSCP

MSCP
MAP

ADDUNIT
LIS

MSCP
LIS

MPWAIT
LIS

MPTIMER
LIS

XDELTA
LIS

MSCPDEF
MAR

SYSMSG
MAP

MSCPSHOW
LIS

NETWRKMSG
MAP

PRGDEVMSG
MAP

MSGFIL

MESSAGE
MAP

CLIUTLMSG
MAP

DEBTBKMSG
MAP

FILMNTMSG
MAP

SHRIMGMSG
MAP

SYSMGTMSG
MAP

CVTMSG
MAP