

```

MMM          MMM          SSSSSSSSSSSSSS          CCCCCCCCCCCCCC          PPPPPPPPPPPPP
MMM          MMM          SSSSSSSSSSSSSS          CCCCCCCCCCCCCC          PPPPPPPPPPPPP
MMM          MMM          SSSSSSSSSSSSSS          CCCCCCCCCCCCCC          PPPPPPPPPPPPP
MMMMMMM      MMMMMM      SSS          CCC          PPP          PPP
MMMMMMM      MMMMMM      SSS          CCC          PPP          PPP
MMMMMMM      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          SSSSSSSSSS          CCC          PPPPPPPPPPPPP
MMM          MMM          SSSSSSSSSS          CCC          PPPPPPPPPPPPP
MMM          MMM          SSSSSSSSSS          CCC          PPPPPPPPPPPPP
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          SSS          CCC          PPP
MMM          MMM          SSSSSSSSSSSSSS          CCCCCCCCCCCCCC          PPP
MMM          MMM          SSSSSSSSSSSSSS          CCCCCCCCCCCCCC          PPP
MMM          MMM          SSSSSSSSSSSSSS          CCCCCCCCCCCCCC          PPP

```

**MSC**

SC  
SC  
SC  
SC  
SC  
SC  
SC  
  
SC  
SC  
SC

SC  
SC  
  
SC  
SC  
  
SC  
SC

```
MM      MM      SSSSSSSS  CCCCCCCC  PPPPPPPP
MM      MM      SSSSSSSS  CCCCCCCC  PPPPPPPP
MMM     MMM     SS          CC          PP      PP
MMM     MMM     SS          CC          PP      PP
MM      MM      SS          CC          PP      PP
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      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
```

|     |      |                                           |
|-----|------|-------------------------------------------|
| (1) | 129  | DECLARATIONS                              |
| (1) | 279  | COMMUNICATIONS AREA                       |
| (1) | 430  | MSU ACTION ROUTINES                       |
| (1) | 546  | LISTEN ACTION ROUTINE                     |
| (1) | 644  | VIRTUAL CIRCUIT ERROR ROUTINE             |
| (1) | 913  | WAIT FOR INT                              |
| (1) | 953  | MAIN LINE ROUTINE                         |
| (1) | 1090 | IMMEDIATE class commands                  |
| (1) | 1107 | - 1 - ABORT                               |
| (1) | 1177 | - 2 - GET COMMAND STATUS                  |
| (1) | 1192 | - 3 - GET UNIT STATUS                     |
| (1) | 1256 | - 4 - SET CONTROLLER CHARACTERISTICS      |
| (1) | 1387 | SEQUENTIAL class commands                 |
| (1) | 1458 | - 8 - AVAILABLE                           |
| (1) | 1656 | - 9 - ONLINE                              |
| (1) | 1825 | - Shadow units                            |
| (1) | 2033 | - ONLINE shadow copy                      |
| (1) | 2244 | - ERROR HANDLING                          |
| (1) | 2393 | - 10 - SET UNIT CHARACTERISTICS           |
| (1) | 2439 | - 11 - DETERMINE ACCESS PATHS             |
| (1) | 2449 | NON-SEQUENTIAL NON-BUFFER class commands  |
| (1) | 2486 | - 16 - ACCESS                             |
| (1) | 2499 | - 17 - COMPARE CONTROLLER DATA            |
| (1) | 2509 | - 18 - ERASE                              |
| (1) | 2529 | - 19 - FLUSH                              |
| (1) | 2539 | - 20 - REPLACE                            |
| (1) | 2549 | MAINTENANCE class commands                |
| (1) | 2554 | NON-SEQUENTIAL BUFFER class commands      |
| (1) | 2589 | - 32 - COMPARE HOST DATA                  |
| (1) | 2600 | - 33 - READ                               |
| (1) | 2817 | - READ REPAIR routine                     |
| (1) | 3011 | - 34 - WRITE                              |
| (1) | 3112 | SETUP - Setup the CDRP for transfers      |
| (1) | 3414 | DO DISK & BACK - DISK IO utility routines |
| (1) | 3555 | UTILITY ROUTINES                          |
| (1) | 3635 | DATA STRUCTURE ALLOCATION                 |
| (1) | 3744 | READ ONLY DATA                            |
| (1) | 3878 | INITIALIZATION ROUTINE                    |

```
0000 1      .TITLE  MSCP - MSCP Server - Emulator
0000 2      .IDENT  'V04-001'
0000 3
0000 4      *****
0000 5      *
0000 6      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      *  ALL RIGHTS RESERVED.
0000 9      *
0000 10     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     *  TRANSFERRED.
0000 16     *
0000 17     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     *  CORPORATION.
0000 20     *
0000 21     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     *
0000 24     *
0000 25     *****
0000 26
0000 27     ++
0000 28
0000 29     FACILITY:      MSCP Server
0000 30
0000 31     ABSTRACT:
0000 32
0000 33         This program implements an MSCP(Mass Storage Control Protocol) server.
0000 34         It accepts as input valid MSCP command packets. It then accesses the
0000 35         disk(s) connected to it to perform the indicated operations. It
0000 36         returns MSCP status packets and possibly data to the driver.
0000 37
0000 38     ENVIORNMENT:  User mode
0000 39
0000 40     AUTHOR:       Kerbey T. Altmann
0000 41
0000 42     CREATION DATE:06-AUG-1980
0000 43
0000 44     MODIFIED BY:
0000 45
0000 46         V04-001 DWT0240      David W. Thiel      7-Sep-1984
0000 47         Disable advertisement of volume shadowing capabilities.
0000 48
0000 49         V03-015 DWT0232      David W. Thiel/Ellen M. Batbouta      31-Aug-1984
0000 50         Update storage allocation scheme. Correct references
0000 51         to CDRPSL_CDI and CDRPSL_CDTX. Tie local CDRP
0000 52         definitions to global symbols better. Detect and
0000 53         bugcheck when response sent using stale CDT address.
0000 54         Deal with CDRPSW STATE more rigorously. Add
0000 55         bug traps to packet deallocation routine. Correct
0000 56         packet initialization on invalid commands. Support
0000 57         determine access paths. Include every incoming
```

0000 58 : message in OPC\_COUNT. Add reference count to HQB.  
0000 59 : Adjust values of PACKETS and SMALL to guarantee that  
0000 60 : things will not hang. Make access command return  
0000 61 : invalid opcode. Validate I/O function code before  
0000 62 : queuing request to driver. Return correct setting  
0000 63 : of MSCPSB\_CPYS for UDA disks. Add many bug traps.  
0000 64 :  
0000 65 : V03-014 EMB0108 Ellen M. Batbouta 19-JUL-1984  
0000 66 : Check for mount verification in progress on a local  
0000 67 : device in routines, NSEQ and NSEQB. If MV is in  
0000 68 : progress, mark unit as available and return without  
0000 69 : processing the request. This prevents a deadlock  
0000 70 : situation.  
0000 71 :  
0000 72 : V03-013 KTA3130 Kerbey T. Altmann 15-May-1984  
0000 73 : Fix error in subcode.  
0000 74 :  
0000 75 : V03-012 KTA3127 Kerbey T. Altmann 23-Apr-1984  
0000 76 : Fix problem with VC error.  
0000 77 :  
0000 78 : V03-011 KTA3116 Kerbey T. Altmann 02-Apr-1984  
0000 79 : Decrement UCB queue length cell, and add entry to  
0000 80 : error table for SSS\_TIMEOUT.  
0000 81 :  
0000 82 : V03-010 KTA3112 Kerbey T. Altmann 12-Mar-1984  
0000 83 : Redo way UQB pointers are handled. Also redo handling  
0000 84 : of blocked commands.  
0000 85 :  
0000 86 : V03-009 KTA3108 Kerbey T. Altmann 06-Mar-1984  
0000 87 : Fix two small bugs and allow for new error return from  
0000 88 : IOS\_PACKACK.  
0000 89 :  
0000 90 : V03-008 KTA3089 Kerbey T. Altmann 12-Nov-1983  
0000 91 : Rewrite to add shadowing.  
0000 92 :  
0000 93 : V03-007 ROW0278 Ralph O. Weber 12-JAN-1984  
0000 94 : Remove reference to MSCPSM\_UF\_INACT, the inactive shadow set  
0000 95 : unit flag. This is no longer a unit flag.  
0000 96 :  
0000 97 : V03-006 ROW0246 Ralph O. Weber 26-OCT-1983  
0000 98 : Move the 'no units at this address' exit label in GET\_UNIT\_STS  
0000 99 : such that a 'next unit modifier' get unit status command gets  
0000 100 : a zero unit number returned when there are no units to  
0000 101 : describe.  
0000 102 :  
0000 103 : V03-005 ROW0236 Ralph O. Weber 10-OCT-1983  
0000 104 : Move increment of number of online hosts counter in UQB to  
0000 105 : before increment of UCBSB\_ONLCNT in online routine. Also make  
0000 106 : increment of UCBSB\_ONLCNT conditional on UQBSB\_HOST\_CNT  
0000 107 : indicating that this is the first host to bring the unit  
0000 108 : online. This makes online operations consistent with their  
0000 109 : description by Kerbey and with the operations performed when a  
0000 110 : available command is processed.  
0000 111 :  
0000 112 : V03-004 KTA3081 Kerbey T. Altmann 13-Sep-1983  
0000 113 : Upgrade AVAIL support. Add underpinnings for shadowing.  
0000 114 :

|      |     |     |         |                                                  |                   |             |
|------|-----|-----|---------|--------------------------------------------------|-------------------|-------------|
| 0000 | 115 | :   | V03-003 | SSA0001                                          | Stan Amway        | 12-Sep-1983 |
| 0000 | 116 | :   |         | Add I/O performance monitoring support.          |                   |             |
| 0000 | 117 | :   |         | Redefined CDRP\$L_LBN to avoid conflicting use   |                   |             |
| 0000 | 118 | :   |         | of CDRP\$L_SEQNUM7IRP\$L_SEQNUM by PMS routines. |                   |             |
| 0000 | 119 | :   |         |                                                  |                   |             |
| 0000 | 120 | :   | V03-002 | KTA3076                                          | Kerbey T. Altmann | 12-Aug-1983 |
| 0000 | 121 | :   |         | Add support for explicit PACK_ACK and AVAIL.     |                   |             |
| 0000 | 122 | :   |         | Redo error routines. Return ALLOCLASS.           |                   |             |
| 0000 | 123 | :   |         |                                                  |                   |             |
| 0000 | 124 | :   | V03-001 | KTA3070                                          | Kerbey T. Altmann | 09-Jul-1983 |
| 0000 | 125 | :   |         | Fix bug in host que search.                      |                   |             |
| 0000 | 126 | :   |         |                                                  |                   |             |
| 0000 | 127 | :-- |         |                                                  |                   |             |

```
0000 129      .SBTTL  DECLARATIONS
0000 130
0000 131      :
0000 132      : INCLUDE FILES:
0000 133      :
0000 134
0000 135      .NOCROSS
0000 136      $CDRPDEF
0000 137      $CDTDEF
0000 138      $DDTDEF
0000 139      $DEVDEF
0000 140      $DYNDEF
0000 141      $IODEF
0000 142      $IPLDEF
0000 143      $IRPDEF
0000 144      $MSCPDEF
0000 145      $PDTDEF
0000 146      $PRDEF
0000 147      $SCSDEF      ; SCS message header definitions
0000 148      $SLVDEF
0000 149      $SSDEF
0000 150      $UCBDEF
0000 151      $VADEF
0000 152
0000 153      $MSUDEF
0000 154      $HQBDEF
0000 155      $UQBDEF
0000 156
0000 157      :
0000 158      : MACROS:
0000 159      :
0000 160
0000 161      .MACRO  BLOCKH
0000 162      BSBW  BLOCKEDH
0000 163      .ENDM
0000 164
0000 165      .MACRO  BLOCKT
0000 166      BSBW  BLOCKEDT
0000 167      .ENDM
0000 168
0000 169      .MACRO  WFINT
0000 170      BSBW  WAIT_FOR_INT
0000 171      .ENDM
0000 172
0000 173      .MACRO  WFINTX
0000 174      BSBW  WAIT_FOR_INTX
0000 175      .ENDM
0000 176      ;
0000 177      .MACRO  QUE HEAD LABEL
0000 178 LABEL: .LONG  LABEL-MSCP_BEGIN
0000 179      .LONG  LABEL-MSCP_BEGIN
0000 180      .ENDM
0000 181
0000 182      .MACRO  ACTION,SYSERR,MSCPERR,ROUTINE
0000 183      .WORD  SYSERR
0000 184      .WORD  MSCPERR
0000 185      .IF B ROUTINE
```

```
0000 186      .WORD 0
0000 187      .IFF
0000 188      .WORD ROUTINE-.
0000 189      .ENDC
0000 190      .ENDM
0000 191
0000 192      :
0000 193      : EQUATED SYMBOLS:
0000 194      :
0000 195
0000 196      HOSTSET      =      <MSCP$M_UF_CMPRD!-
0000 197                      MSCP$M_UF_CMPWR!-
0000 198                      MSCP$M_UF_576!-
0000 199                      MSCP$M_UF_SCCHH!-
0000 200                      MSCP$M_UF_SCCHL!-
0000 201                      MSCP$M_UF_WBKNV!-
0000 202                      MSCP$M_UF_WRTPS>
00001C47 0000 203
0000 204      NONHOSTSET  =      <MSCP$M_UF_REPLC!-
0000 205                      MSCP$M_UF_RMVBL!-
0000 206                      MSCP$M_UF_WRTPH!-
0000 207                      MSCP$M_UF_SSMEM!-
0000E280 0000 208                      MSCP$M_UF_SSMST>
0000 209
0000 210      :
FFFFF1B0 0000 211      CDRP$L_MSGBUF_X =      CDRP$L_AST
FFFFF1B4 0000 212      CDRP$L_BUFLN  =      CDRP$L_ASTPRM
FFFFF1B8 0000 213      CDRP$L_BUFADR =      CDRP$L_WIND
FFFFF1C2 0000 214      CDRP$B_SHDW_CNT =      CDRP$B_EFN
FFFFF1C4 0000 215      CDRP$L_UMB  =      CDRP$L_IOSB
FFFFF1D6 0000 216      CDRP$W_STATE =      CDRP$L_BCNT+4
FFFFF1E8 0000 217      CDRP$L_UMB_FL =      CDRP$L_SEGVBN
FFFFF1EC 0000 218      CDRP$L_UMB_BL =      CDRP$L_DIAGBUF
FFFFF1F4 0000 219      CDRP$L_CMD_STS =      CDRP$L_EXTEND
FFFFF1F8 0000 220      CDRP$W_SLOT  =      CDRP$L_ARB
FFFFF1FA 0000 221      CDRP$B_FLAGS =      CDRP$L_ARB+2
FFFFF1FB 0000 222      CDRP$B_OPCODE =      CDRP$L_ARB+3
0000 223      :
0000 224      : Define MSCP server private areas beyond normal CDRP fields
0000 225      :
00000048 0000 226      CDRP$L_LBN      =      CDRP$K_CD_LEN
0000004C 0000 227      CDRP$L_SHDW_LNK =      CDRP$L_LBN+4
00000050 0000 228      CDRP$L_FPCX      =      CDRP$L_SHDW_LNK+4
00000054 0000 229      CDRP$L_FR3X      =      CDRP$L_FPCX+4
00000058 0000 230      CDRP$L_FR4X      =      CDRP$L_FR3X+4
0000005C 0000 231      CDRP$L_SAVD_RTX =      CDRP$L_FR4X+4
00000060 0000 232      CDRP$L_CDTX      =      CDRP$L_SAVD_RTX+4
00000064 0000 233      CDRP$L_XCT_LENX =      CDRP$L_CDTX+4
00000068 0000 234      CDRP$B_LBUFF      =      CDRP$L_XCT_LENX+4
00000074 0000 235      CDRP$L_STRT_UMB =      CDRP$B_LBUFF+12
00000078 0000 236      CDRP$L_MST_UMB =      CDRP$L_STRT_UMB+4
0000007C 0000 237      CDRP$L_HQB      =      CDRP$L_MST_UMB+4
000000E0 0000 238      BIG_CDRP =      <CDRP$L_HQB+4>-CDRP$L_IOQFL
0000 239
00000022 0000 240      IRP$B_SHDW_CNT =      CDRP$B_SHDW_CNT-CDRP$L_IOQFL
00000024 0000 241      IRP$L_UMB      =      CDRP$L_UMB-CDRP$L_IOQFL
00000036 0000 242      IRP$W_STATE =      CDRP$W_STATE-CDRP$L_IOQFL
```

```
00000058 0000 243 IRPSW_SLOT = CDRPSW_SLOT-CDRPSL_IOQFL
0000005A 0000 244 IRPSB_FLAGS = CDRPSB_FLAGS-CDRPSL_IOQFL
000000A8 0000 245 IRPSL_LBN = CDRPSL_LBN-CDRPSL_IOQFL
000000AC 0000 246 IRPSL_SHDW_LNK = CDRPSL_SHDW_LNK-CDRPSL_IOQFL
000000B0 0000 247 IRPSL_FPCX = CDRPSL_FPCX-CDRPSL_IOQFL
000000B4 0000 248 IRPSL_MAINCDRP = CDRPSL_FR3X-CDRPSL_IOQFL
000000B8 0000 249 IRPSL_FR4X = CDRPSL_FR4X-CDRPSL_IOQFL
0000 250 ASSUME <IRPSL_FR4X+4> LE IRPSC_LENGTH
0000 251
00000000 0000 252 CDRPSV_SLAVE = 0 ; This is a slave IRP
00000001 0000 253 CDRPSV_BUFFER = 1 ; This CDRP has a buffer
0000 254 : CDRPSV = 2 ;
0000 255 : CDRPSV = 3 ;
00000004 0000 256 CDRPSV_ABORT = 4 ; This CDRP has been aborted
00000005 0000 257 CDRPSV_NOHSTBUF = 5 ; This CDRP has no host buffer
0000 258
00000000 0000 259 CDRPSK_ST_MEM_WAIT = 0 ; Buffer memory wait
00000001 0000 260 CDRPSK_ST_MAP_WAIT = 1 ; SCS mapping resource wait
00000002 0000 261 CDRPSK_ST_DRVWD_WAIT = 2 ; Disk read wait
00000003 0000 262 CDRPSK_ST_DATA_WAIT = 3 ; SCS data transfer wait
00000004 0000 263 CDRPSK_ST_DRVWRT_WAIT = 4 ; Disk write wait
00000005 0000 264 CDRPSK_ST_MSG_WAIT = 5 ; Response message credit wait
00000006 0000 265 CDRPSK_ST_SHQ_QUE = 6 ; Special holding queue wait
00000007 0000 266 CDRPSK_ST_BLK_QUE = 7 ; Blocked queue wait
00000008 0000 267 CDRPSK_ST_ONBLK_QUE = 8 ; Online-copy blocked queue
00000009 0000 268 CDRPSK_ST_MSG1_WAIT = 9 ; Attention message credit/buffer wait
0000003F 0000 269 CDRPSK_ST_INVACID = 63 ; Invalid state
0000 270
0000 271 :
0000 272 : EXTERNAL SYMBOLS:
0000 273 :
0000 274
0000 275 .CROSS
0000 276 .DEFAULT DISPLACEMENT WORD
0000 277
```

```
0000 279 .SBTTL COMMUNICATIONS AREA
0000 280
00000000 281 .PSECT ASCP$CODE,PAGE
0000 282
0000 283 MSCP_BEGIN:
0000 284 MSCP_COMM_AREA:
0000 285
00000001 0000 286 PRMSW = 1
0000 287
0000 288 SLVTAB END = MSCP_END,-
0000 289 INITRTN = MSCP_START,-
0000 290 SUBTYP = DYN$LC_MSCP,-
0000 291 FACILITY= <MSCP_SERVER>
0024 292
0024 293 LOADVEC TYPE = SLV$K_SDATA,-
0024 294 ENTRY = SC$SGC_MSCP,-
0024 295 SEC_LABEL = MSCP_BEGIN
00000000 002D 296 DEF_RTN = 0
002D 297
FFFFFFFF FFFFFFFF FFFFFFFF 002D 298 .LONG -1,-1,-1
0039 299
0039 300 .ALIGN QUAD
0040 301
00000040 0040 302 .=MSCP_COMM_AREA+MSUSL_STATE
0040 303 STATE:
00000000 0040 304 .LONG 0
0044 305
00000044 0044 306 .=MSCP_COMM_AREA+MSUSW_PACKET
0000 0044 307 PACKET: .WORD 0
0046 308
00000046 0046 309 .=MSCP_COMM_AREA+MSUSW_INI_PKT
0046 310 INI_PKT: ; Number of allocated (free+in-use) packets
0000 0046 311 .WORD 0
0048 312
00000048 0048 313 .=MSCP_COMM_AREA+MSUSW_NUM_PKT
0048 314 NUM_PKT: ; Number of free packets
0000 0048 315 .WORD 0
004A 316
0000004A 004A 317 .=MSCP_COMM_AREA+MSUSW_MAX_PKT
004A 318 MAX_PKT: ; Maximum number of packets to cache
0000 004A 319 .WORD 0
004C 320
0000004C 004C 321 .=MSCP_COMM_AREA+MSUSW_INI_HOST
004C 322 INI_HOST:
0000 004C 323 .WORD 0
004E 324
0000004E 004E 325 .=MSCP_COMM_AREA+MSUSW_NUM_HOST
004E 326 NUM_HOST:
0000 004E 327 .WORD 0
0050 328
00000050 0050 329 .=MSCP_COMM_AREA+MSUSL_BUFF_HEAD
0050 330 BUFF_HEAD:
00000000 0050 331 .LONG 0
0054 332
00000054 0054 333 .=MSCP_COMM_AREA+MSUSL_MAX_BUF
0054 334 MAX_BUF:
00000000 0054 335 .LONG 0
```

```
00000058 0058 336
00000000 0058 337 SMALL: . =MSCP_COMM_AREA+MSUSL_SMALL
00000000 0058 338
00000000 0058 339 .LONG 0
0000005C 005C 340
00000000 005C 341 . =MSCP_COMM_AREA+MSUSL_FRACTION
00000000 005C 342 FRACTION:
00000000 005C 343 .LONG 0
00000060 0060 344
00000060 0060 345 . =MSCP_COMM_AREA+MSUSW_NUM_QUE
0000 0060 346 NUM_QUE:
00000062 0062 347 .WORD 0
00000062 0062 348
00000062 0062 349 . =MSCP_COMM_AREA+MSUSW_MAX_QUE
0000 0062 350 MAX_QUE:
00000000 0064 351 .WORD 0
00000000 0064 352 .LONG 0 ; Reserved
00000068 0068 353
00000068 0068 354 . =MSCP_COMM_AREA+MSUSL_CDSV_SIZE
00000000 0068 355 CDSV_SIZE:
0000006C 006C 356 .LONG 0
0000006C 006C 357
0000006C 006C 358 . =MSCP_COMM_AREA+MSUSL_CDRP_SAVE
00000000 006C 359 CDRP_SAVE:
00000070 0070 360 .LONG 0
00000070 0070 361
00000078 0078 362 . =MSCP_COMM_AREA+MSUSL_CDRP_LIST
00000078 0078 363 QUEUES: QUE HEAD CDRP LIST
00000080 0080 364 . =MSCP_COMM_AREA+MSUSL_MEM_WAIT
00000080 0080 365 QUE HEAD MEM WAIT
00000003 0088 366 . =MSCP_COMM_AREA+MSUSL_UQB_LIST
00000003 0088 367 QUE HEAD UQB LIST
00000088 0088 368 QUE_NUM = <QUEUES>78
00000088 0088 369 ASSUME MSUSC_NUMQUE EQ QUE_NUM
00000088 0088 370
00000088 0088 371 . =MSCP_COMM_AREA+MSUSL_HOST_DSBL
00000000 0088 372 HOST_DSBL:
00000000 0088 373 .LONG 0
0000008C 008C 374
00000000 008C 375 . =MSCP_COMM_AREA+MSUSL_UNIT_VEC-4
00000000 008C 376 .LONG 0 ; *** ENTRY FOR UNIT # -1 !!!!!
00000110 0090 377 UNIT_VEC:
00000110 0090 378 .BLKL 2*MSUSK_MAX_UNIT
00000110 0110 379
00000110 0110 380 . =MSCP_COMM_AREA+MSUSL_HOST_VEC
00000190 0110 381 HOST_VEC:
00000190 0190 382 .BLKL 2*MSUSK_MAX_HOST
00000190 0190 383
00000190 0190 384 . =MSCP_COMM_AREA+MSUSQ_CTRL_ID
00000000 0190 385 CTRL_ID:
0000 0190 386 .LONG 0 ; Controller id
0000 0194 387 .WORD 0
0000 0196 388 .BYTE MSCPSK_CM_EMULA
0000 0197 389 .BYTE MSCPSK_CL_CNTRL
00000198 0198 390
00000198 0198 391 . =MSCP_COMM_AREA+MSUSQ_CTRL_INFO
00000198 0198 392 CTRL_INFO:
```

```
0198 393 VERSION:
0000 0198 394 .WORD 0 ; VERSION
8004 019A 395 .WORD <MSCPSM_CF_REPLC!- ; controller flags
019C 396 - ; disable advertisement of shadowing MSCPSM_CF_SHADW!-
019C 397 MSCPSM_CF_MLTHS>
FFFF 019C 398 .WORD -1 ; Controller timeout
0000 019E 399 .WORD 0 ; Reserved
01A0 400
000001A0 01A0 401 . =MSCP_COMM_AREA+MSUSB_CTRL_DAT
01A0 402 CTRL_DAT:
00000000 00000000 01A0 403 .QUAD 0
00000000 00000000 01A8 404 .QUAD 0
01B0 405
000001B0 01B0 406 . =MSCP_COMM_AREA+MSUSL_OPCOUNT
01B0 407 OPC_COUNT:
000002B0 01B0 408 .BLKL 64
02B0 409
000002B0 02B0 410 . =MSCP_COMM_AREA+MSUSL_BLKCOUNT
02B0 411 BLK_COUNT:
000004B4 02B0 412 .BLKL 129 ; Statistics for block sizes
04B4 413
000004B4 04B4 414 . =MSCP_COMM_AREA+MSUSQ_HQBLIST
04B4 415 HQBLIST:
000004BC 04B4 416 .BLKQ 1 ; List of allocated HQB's
04BC 417
000004CC 04BC 418 . =MSCP_COMM_AREA+MSUSK_LENGTH
04CC 419
000004D0 04CC 420 BUFF_LENGTH:
04CC 421 .BLKL 1 ; Length of allocated buffer
04D0 422 BUFF_ADDR:
000004D4 04D0 423 .BLKL 1 ; Address of allocated buffer
04D4 424 BUFF_END:
000004D8 04D4 425 .BLKL 1 ; Length+Address
04D8 426 IO_PRI:
000004D9 04D8 427 .BLKB 1
04D9 428
```

```
04D9 430 .SBTTL MSU ACTION ROUTINES
04D9 431
04D9 432 :+
04D9 433 :
04D9 434 : The .e routines are called by the MSU program to carry out actions
04D9 435 : that must be done from high IPL code that lives in non-paged pool.
04D9 436 :
04D9 437 : Inputs:
04D9 438 :
04D9 439 :     AP --> #arguments
04D9 440 :             action code
04D9 441 :             parameter(s)
04D9 442 :
04D9 443 : Outputs:
04D9 444 :
04D9 445 :     R0 = status code
04D9 446 :
04D9 447 :-
04D9 448
04D9 449 ASSUME MSUSK_AC_STRT EQ 1
04D9 450 ASSUME MSUSK_AC_STOP EQ 2
04D9 451 ASSUME MSUSK_AC_ADD EQ 3
04D9 452 ASSUME MSUSK_AC_REM EQ 4
04D9 453 ASSUME MSUSK_AC_DISC EQ 5
04D9 454
04D9 455 .ENABLE LSB
04D9 456
04D9 457 MSCP_START:
04D9 458
04D9 459 CASE 4(AP),LIMIT=#1,TYPE=B-
04D9 460 <START,- ; Startup
04D9 461 STOP,- ; Stop
04D9 462 ADD,- ; Add a unit
04D9 463 REMOVE,- ; Remove a unit
04D9 464 DISCON> ; Disconnect a host
04E8 465
50 14 3C 04E8 466 MOVZWL #SS$_BADPARAM, R0 ; Command out of range
05 04E8 467 10$: RSB ; Leave
04EC 468
04EC 469 :
04EC 470 : START the server
04EC 471 :
04EC 472
50 02DC 8F 3C 04EC 473 START: MOVZWL #SS$_ILLSEQOP, R0 ; Set an error code
1F E2 04F1 474 BBSS #MSUSV_START,- ; Check if already started
F4 FB4A CF 04F3 475 STATE,10$
192E 31 04F7 476 BRW START_CONT ; Do it
04FA 477
04FA 478 .DISABLE LSB
04FA 479
04FA 480 :
04FA 481 : ADD a disk unit
04FA 482 :
04FA 483 .ENABLE LSB
04FA 484
50 01 D0 04FA 485 10$: MOVL #1,R0
05 04FD 486 RSB
```

```

      F6 FB3D 1F E1 04FE 487 ADD: BBC #MSUSV_START,- ; Leave if not started yet
      52 08 AC D0 0500 488 STATE,10$
      0504 489 MOVL 8(AP),R2 ; Transfer UQB address
      0508 490
      0508 491 .DISABLE LSB
      0508 492
      0508 493 SEND_ATTN:
      7E 56 7D 0508 494 MOVQ R6,-(SP) ; Unit counter
      56 1F D0 050B 495 MOVL #<MSUSK_MAX_HOST*2>-1,R6 ; Largest host slot index
      57 52 D0 050E 496 MOVL R2,R7 ; UQB address
53 FBFA CF46 D0 0511 497 10$: MOVL HOST_VECC[R6],R3 ; Obtain the HQB pointer
      22 13 0517 498 BEQL 20$ ; None, skip on
      171D 30 0519 499 BSBW GET_STRUCT ; Allocate a packet
      1F 50 E9 051C 500 BLBC R0,30$ ; Branch on failure
      7C A5 53 D0 051F 501 MOVL R3,CDRPSL_HQB(R5) ; Store HQB address
      2C A3 B6 0523 502 INCW HQBSW_REFC(R3) ; Up HQB reference count
      C4 A5 57 D0 0526 503 MOVL R7,CDRPSL_UQB(R5) ; Address of UQB
      20 A5 D4 052A 504 CLRL CDRPSL_RSPID(R5) ; Never send w/RSPID
      54 14 A3 D0 052D 505 MOVL HQBSL_CDT(R3),R4 ; Pick up CDT
      24 A5 54 D0 0531 506 MOVL R4,CDRPSL_CDT(R5) ; Set into CDRP
      54 10 A4 D0 0535 507 MOVL CDTSL_PDT(R4),R4 ; Now grab the PDT address
      07 10 0539 508 BSBW 40$ ; Set up to return synchronously
      D3 56 F4 053B 509 20$: SOBGEQ R6,10$ ; Iterate over all units
      56 8E 7D 053E 510 30$: MOVQ (SP)+,R6 ; Restore registers
      05 0541 511 RSB
      0542 512
      D6 A5 09 B0 0542 513 40$: MOVW #CDRPSK_ST_MSG1_WAIT,- ; Tag this state
      0546 514 CDRPSW_STATE(R5)
      0546 515 ALLOC_MSG_BUF ; Allocate a message buffer
      53 82 D4 0549 516 CLRL (R2)+ ; No command ref number on attn msg
      82 28 A3 D0 054B 517 MOVL CDRPSL_UQB(R5),R3 ; Pick up unit block
      82 40 8F 3C 054F 518 MOVZWL UQBSW_UNIT(R3),(R2)+ ; Copy unit number
      82 10 A3 D0 0553 519 MOVZBL #64,(R2)+ ; Set the attn msg code
      82 82 D4 0557 520 MOVL UQBSW_MULT_UNT(R3),(R2)+ ; Set mult unit & unit_flags
      82 18 A3 7D 055B 521 CLRL (R2)+ ; Reserved field
      62 20 A3 D0 055D 522 MOVQ UQBSQ_UNIT_ID(R3),(R2)+ ; Copy unit identifier
      51 20 D0 0561 523 MOVL UQBSL_MEDIA_ID(R3),(R2) ; Copy media identifier
      0565 524 MOVL #32,R7 ; Size
      0568 525 SEND_CNT_MSG_BUF ; Send it out
      056B 526 ATTN_CLEANUP:
      53 7C A5 D0 056B 527 MOVL CDRPSL_HQB(R5),R3 ; HQB address
      2C A3 B7 056F 528 DECW HQBSW_REFC(R3)
      1723 31 0572 529 BRW FREE_STRUCT ; Release data structure
      0575 530
      0575 531 ; REMOVE a disk unit
      0575 532
      0575 533
      05 0575 534 REMOVE: RSB
      0576 535
      0576 536
      0576 537 ; DISCONNECT a host
      0576 538
      0576 539
      53 08 AC D0 0576 540 DISCON: MOVL 8(AP),R3 ; Pick up CDT
      54 10 A3 D0 057A 541 MOVL CDTSL_PDT(R3),R4 ; Pick up PDT
      00E9 30 057E 542 BSBW VC_ERR ; Do the disconnect
      50 01 D0 0581 543 MOVL #1,R0 ; Always success
```

MSCP  
V04-001

- MSCP Server - Emulator  
MSU ACTION ROUTINES

05 0584 544 RSB

I 13

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 12  
(1)

MS  
V0

```
0585 546 .SBTTL LISTEN ACTION ROUTINE
0585 547 :
0585 548 : When someone wants to connect with us, control will come here
0585 549 : as a fork process initiated by the port driver at IPL 8. If the
0585 550 : connector is the class driver, we will accept it, otherwise we will
0585 551 : reject the connection. Upon a successful acceptance, allocate a
0585 552 : host control block and wait for a message.
0585 553 :
0585 554 : R2 --> Connect message
0585 555 : R3 --> CDT of listen
0585 556 : R4 --> PDT
0585 557 :
0585 558 :
0585 559 LIS_FORK:
0585 560 EXTZV #0,#4,- ; Pick up low 4 bits of node
0585 561 CDT$B_RSTATION(R3),R1
0585 562 BBC R1,HOST_DSBL,20$ ; Have we disabled this host?
0591 563 10$: REJECT #1 ; Yes, don't connect
0597 564 RSB ; Return to Port Driver
0598 565
0598 566 20$: TSTW NUM_HOST ; Any host space left?
059C 567 BEQL 10$ ; No, reject this connect
059E 568 CMPB R1,#MSUSK_MAX_HOST ; Is it a funny one?
05A1 569 BGEQU 30$ ; Yes, bring it in line
05A3 570 TSTL HOST_VEC[R1] ; Any one there now?
05A8 571 BEQL 50$ ; No, okay to use
05AA 572 30$: MOVL #MSUSK_MAX_HOST,R1 ; Starting place for search
05AD 573 40$: TSTL HOST_VEC[R1] ; Any one there now?
05B2 574 BEQL 50$ ; Yes
05B4 575 AOBLS #MSUSK_MAX_HOST*2,R1,40$ ; Try again
05B8 576 BRB 10$ ; No room, reject the connect
05BA 577
05BA 578 50$: PUSHL R1 ; Save vector
05BC 579 PUSHL R3
05BE 580 MOVZWL #HQB$C_LEN, R1 ; Size needed
05C3 581 BSBW ALLOMEM ; Grab some pool
05C6 582 POPL R3
05C9 583 BLBS R0,60$ ; All set
05CC 584 TSTL (SP)+ ; Clean stack
05CE 585 BRB 10$ ; Reject the connect
05D0 586
05D0 587 ASSUME HQB$B_FLAG EQ HQB$B_VEC+1
05D0 588 60$: CVTLW (SP)+,HQB$B_VEC(R2) ; Set the vector number & clear flag
05D4 589 MOVL R2,R5 ; Transfer address
05D7 590 CLRQ (R2)+ ; Clean out trash
05D9 591 MOVW R1,(R2)+ ; Set length
05DC 592 MOVW #<DYN$C_SCS_HQB$B>!,- ; Set type
05E1 593 DYN$C_SCS,(R2)+
05E1 594 :
05E1 595 : ACCEPT the connection and que up the HQB
05E1 596 :
05E1 597 ACCEPT MSGADR=MSG_IN,- ; Message input addr
05E1 598 DGADR=B^DG_ERR,- ; DG input
05E1 599 ERRADR=VC_ERR,- ; Error address
05E1 600 INITCR=PACKET,- ; Initial credit
05E1 601 MINSCR=#1,- ; Minimum send credit
05E1 602 CONDAT=CTRL_DAT,- ; CONNECT/ACCEPT data
```

```

      3D 50 E9 05E1 603
FEA1 CF 65 OE 060B 604
52 OC A5 DE 060E 605
      82 7C 0613 606
      0617 607
      0619 608
      82 53 D0 0619 609
      82 34 061C 610
      061E 611
      04 91 061E 612
      07 A4 12 0620 613
      04 12 0622 614
1F A5 01 90 0624 615
      28 A5 B4 0628 616 70$:
      2A A5 B4 062B 617
      2C A5 01 B0 062E 618
50 30 A5 9E 0632 619
      60 50 D0 0636 620
      60 80 DE 0639 621
      51 1E A5 9A 063C 622
FACA CF41 55 D0 0640 623
      FA04 CF B7 0646 624
      05 064A 625
      064B 626
      50 55 D0 064B 627 DEALRO: MOVL R5,R0
      064E 628
00000000'9F 17 064E 629 DEALMEM: JMP @#EXES$DEANONPAGED
      0654 630
00000000'9F 17 0654 631 ALLOMEM: JMP @#EXES$ALONONPAGED
      065A 632
      065A 633 ; Error
      065A 634
      065A 635 DG_ERR: BUG_CHECK MSCPSERV,FATAL
      065E 636
53 1CE8'CF D0 065E 637 STOP: MOVL LIS_CDT, R3
      0663 638
      0663 639 LIS_ERR:DISCONNECT
      05 0669 640 RSBX: RSB
      066A 641
      066A 642 .DISABLE lsb

AUXSTR=(R5) ; Auxiliary structure
R0,DEALRO ; ERROR
HQB$L_FLINK(R5),HQBLIST ; Link to list of allocated HQB's
12(R5),R2 ; Bump to next
(R2)+
HQB$C_CDT EQ 20
MOV R3,(R2)+ ; Save the CDT address
CLR (R2)+ ; Clear more flags
HQB$W_USE_FRAC EQ 28
#PDT$C_PS=-
PDT$B_PDT_TYPE(R4) ; Is it the passthru driver?
70$ ; No
#1,HQB$B_FLAG(R5) ; Yes, set flag
HQB$W_NUM_QUE(R5)
HQB$W_MAX_QUE(R5)
#1,HQB$W_REFC(R5) ; Reference count = 1 for CDT
HQB$Q_PENDING(R5),R0 ; Cleanup pending queue
R0,(R0) ; Initialize forward pointer
(R0)+,(R0) ; Initialize back pointer
HQB$B_VEC(R5),R1
MOV R5,HOST_VEC[R1] ; Save the host block address
DECV NUM_HOST ; Drop number of hosts left
RSB

R5,R0 ; Set up pointer
@#EXES$DEANONPAGED ; Get rid of HQB and return
@#EXES$ALONONPAGED ; Get memory
; Error
DG_ERR: BUG_CHECK MSCPSERV,FATAL ; ERROR
STOP: MOVL LIS_CDT, R3 ; Pick up the listen cdt
LIS_ERR:DISCONNECT ; Do a disconnect
RSBX: RSB ; Return
.DISABLE lsb
```

```
066A 644 .SBTTL VIRTUAL CIRCUIT ERROR ROUTINE
066A 645
066A 646 :*****
066A 647 :
066A 648 :
066A 649 : 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 650 :
066A 651 :
066A 652 :*****
066A 653 :
066A 654 : Return all data structures for this host and break link
066A 655 :
066A 656 : R3 = CDT addr
066A 657 : R4 = PDT addr
066A 658 :
066A 659 :
066A 660 .ENABLE LSB
066A 661
066A 662 VC_ERR:
066A 663 PUSHHR #*M<R3,R4,R6,R7,R8,R9,R10,R11> ; Save registers
066A 664 MOVL CDT$L_AUXSTRUC(R3),R8 ; Pick up HQB
066A 665 TSTW HQB$W_NUM_QUE(R8) ; Are there any CDRP's outstanding?
066A 666 BEQL 30$ ; No, leave now
066A 667 MOVL #1,R9 ; Init iteration counter
066A 668 10$: MNEGL #1,R10 ; Yes, Start at unit 0
066A 669 20$: INCL R10 ; Next bit position
066A 670 MOVZBL #<2*MSUSK_MAX_UNIT>-1,R1 ; Max bit size
066A 671 SUBL R10,R1 ; Number left
066A 672 FFS R10,R1,-
066A 673 HQB$L_UNIT_ONLN(R8),R10 ; Find any units online
066A 674 BNEQ 40$ ; Got one
066A 675 SOBGEQ R9,10$ ; Finished?
066A 676 30$: BRW FIN_HQB ; Yes, leave now
066A 677 :
066A 678 ; Now run thru the queues on the UQB to find all CDRP's. NOTE: The order
066A 679 ; in which queues are accessed is important and should not be changed.
066A 680 :
066A 681 40$: MOVL UNIT_VEC[R10],R11 ; Get unit que block
066A 682 BEQL 20$ ; None
066A 683 MOVAB UQB$Q_SHQ(R11),R6 ; Point to special holding quehead
066A 684 BSBB 50$ ; Do the work
066A 685 MOVAB UQB$Q_BLOCKQ(R11),R6 ; Point to blocked quehead
066A 686 BSBB 50$ ; Do the work
066A 687 TSTW UQB$W_NUM_QUE(R11) ; Any CDRP's queued?
066A 688 BEQL 20$ ; No
066A 689 MOVAB UQB$L_CDRP_FL(R11),R6 ; Point to quehead
066A 690 BSBB 50$ ; Do the work
066A 691 BRB 20$ ; Try for another unit
066A 692 :
066A 693 ; Subroutine to scan queue and dispatch to appropriate code.
066A 694 :
066A 695 50$: MOVL (R6),R7 ; Get first entry
066A 696 60$: CMPL R7,R6 ; Is there any more?
066A 697 BEQL RSBX ; No
066A 698 MOVAL -CDRP$L_UQB_FL(R7),R5 ; Cover the CDRP portion
066A 699 MOVL (R7),R7 ; Get next entry (later REMQUE zeros flink)
066A 700 CMPL HQB$L_CDT(R8),- ; Belong to this host?
```

58 0FD8 8F BB 066A 663  
5C A3 D0 066E 664  
28 A8 B5 0672 665  
19 13 0675 666  
59 01 D0 0677 667  
5A 01 CE 067A 668  
5A 5A D6 067D 669  
51 1F 9A 067F 670  
51 5A C2 0682 671  
5A 10 A8 51 5A EA 0685 672  
06 12 068B 673  
EA 59 F4 068D 675  
010C 31 0690 676  
0693 677  
0693 678  
0693 679  
0693 680  
5B F9F8 CF4A D0 0693 681  
E2 13 0699 682  
56 70 AB 9E 069B 683  
13 10 069F 684  
56 68 AB 9E 06A1 685  
0D 10 06A5 686  
50 AB B5 06A7 687  
D1 13 06AA 688  
56 60 AB 9E 06AC 689  
02 10 06B0 690  
C9 11 06B2 691  
06B4 692  
06B4 693  
06B4 694  
57 66 D0 06B4 695  
56 57 D1 06B7 696  
AD 13 06BA 697  
55 18 A7 DE 06BC 698  
57 67 D0 06C0 699  
14 A8 D1 06C3 700

```
60 A5 12 06C6 701 CDRP$L_CDTX(R5)
ED 12 06C8 702 BNEQ 60$ ; No
06CA 703
06CA 704 ; Dispatch on the state value. This is changed each time the CDRP goes
06CA 705 ; thru a particular routine. Each state may need different treatment.
06CA 706
06CA 707 BISB #1aCDRP$V_ABORT,- ; Set 'ABORTED' bit
06CC 708 CDRP$B_FLAGS(R5)
53 FA A5 10 88 06CE 709 MOVL HQB$L_CDT(R8),R3 ; Pick up the cdt address in r3
50 D6 A5 9A 06D2 710 MOVZBL CDRP$W_STATE(R5),R0 ; Pick up state
06D6 711 CASE R0,-
06D6 712 <ST_MEM_WAIT,- ; 0-Memory wait
06D6 713 ST_MAP_WAIT,- ; 1-MAP resource wait
06D6 714 ST_DRV_WAIT,- ; 2-Driver que wait
06D6 715 ST_DRV_WAIT,- ; 3-DATA command wait
06D6 716 ST_DRV_WAIT,- ; 4-Driver que wait
06D6 717 ST_MSG_WAIT,- ; 5-MSG send wait
06D6 718 ST_SHQ_QUE,- ; 6-Special holding que
06D6 719 ST_BLK_QUE,- ; 7-Blocked que
06D6 720 ST_ONBCK_QUE> ; 8-Online-copy Blocked que
06EC 721
06EC 722 BUG_CHECK MSCPSERV,FATAL ; ERROR
06F0 723
06F0 724 65$: BUG_CHECK MSCPSERV,FATAL ; ERROR -- connection / CDT mismatch
06F4 725
06F4 726 ;
06F4 727 ; Driver wait: The CDRP has been delivered to the disk driver and is
06F4 728 ; awaiting completion. It has a mapping resource, a buffer, and a
06F4 729 ; message buffer.
06F4 730
06F4 731 ST_DRV_WAIT:
06F4 732 BLBS R9,60$ ; Skip first time
52 B0 A5 18 A3 D0 06F7 733 MOVL CDRP$L_MSGBUFX(R5),R2 ; Pick up message buffer
F8 A2 03 B1 12 06FB 734 CMPL CDT$L_CONID(R3),- ; Check validity of remote conid
0700 735 SC$S$L_DST_CONID(R2)
0700 736 BNEQ 65$ ; Trying to respond to stale connection
B0 A5 12 0702 737 CLRL CDRP$L_MSGBUFX(R5) ; Show it gone
0705 738 DEALLOC_MSG_BUF_REG ; Deallocate it
D6 A5 03 B1 12 0708 739 CMPW #3,CDRP$W_STATE(R5) ; Is it data wait?
070C 740 BNEQ 60$ ; No, allow it to finish
070E 741
070E 742 ;
070E 743 ; Send/Request data wait: The CDRP is somewhere in the process of doing
070E 744 ; a send/request data operation. It has a mapping resource, a buffer, and
070E 745 ; possibly a RSPID and a message buffer.
070E 746
070E 747 ST_DAT_WAIT:
20 A5 06 13 070E 748 TSTL CDRP$L_RSPID(R5) ; No, if there is a RSPID
0711 749 BEQL 70$ ; .. then
0713 750 DEALLOC_RSPID ; Deallocate it
1C A5 03 13 0719 751 70$: TSTL CDRP$L_MSG_BUF(R5) ; If there is something
071C 752 BEQL 80$ ; .. then
071E 753 DEALLOC_MSG_BUF ; Deallocate the message buffer
0721 754 80$:
50 65 D0 0721 755 ; ??? Crock that assumes that this field is initialized!
0721 756 MOVL CDRP$L_FQFL(R5),R0 ; Get possible next element
0724 757 ASSUME CDRP$L_FQFL EQ 0
```

```
55 04 A0 D1 0724 758      CMPL 4(R0),R5      ; Is this block queued to anything?
      03 12 0728 759      BNEQ 90$          ; No
50 65 0F 072A 760      REMQUE CDRP$L_FQFL(R5),R0 ; Remove from wait que
      50 D4 072D 761      : ??? End crock
      53 7D 072D 762 90$: CLRL R0          ; Set for failure
7E 0C B5 16 072F 763      MOVQ R3,-(SP)      ; Save CDT,PDT
      53 8E 7D 0732 764      JSB @CDRP$L_FPC(R5) ; Finish it off
      FF7C 31 0735 765      MOVQ (SP)+,R3    ; Restore
      0738 766      BRW 60$                ; Loop for next CDRP
      073B 767
      073B 768
      073B 769 ; Memory wait: The CDRP is waiting for a data buffer. It has no resources
      073B 770 ; and can be immediately cleaned up after being dequeued from memory wait.
      073B 771
      073B 772 ST_MEM_WAIT:
      073B 773      DECW NUM_QUE
      073F 774
      073F 775
      073F 776 ; Message wait: The CDRP is on a credit wait queue from a RECYCL_MSG_BUF
      073F 777 ; operation. It has a buffer and a message buffer. It can be dequeued
      073F 778 ; from the credit wait queue.
      073F 779
      073F 780 ST_MSG_WAIT:
      073F 781
      073F 782
      073F 783 ; Map wait: The CDRP is waiting for mapping resources. It has a data
      073F 784 ; buffer resource only. It can be immediately cleaned up after being dequeued
      073F 785 ; from the map wait queue.
      073F 786
      073F 787 ST_MAP_WAIT:
50 65 0F 073F 788      REMQUE CDRP$L_FQFL(R5),R0 ; Remove from wait que
      2D 11 0742 789      BRB 150$          ; and finish off
      0744 790
      0744 791
      0744 792 ; Special holding que: The CDRP is waiting for CIP. It has no resources
      0744 793 ; and can be immediately cleaned up after being dequeued.
      0744 794
      0744 795 ST_SHQ_QUE:
50 E8 A5 0F 0744 796      REMQUE CDRP$L_UQB_FL(R5),R0 ; Remove from special holding que
      04 12 0748 797      BNEQ 100$        ; Not the last entry
      08 8A 074A 798      BICB #UQB$M_CMDPEND,- ; Clear the command pending status
      2B AB 074C 799      UQB$B_FLAGS(R11)
      60 D4 074E 800 100$: CLRL (R0)        ; Show not queued
      28 11 0750 801      BRB 155$        ; Done
      0752 802
      0752 803
      0752 804 ; Blocked que: The CDRP is blocked. It has a message buffer and possibly
      0752 805 ; a buffer. It can be immediately cleaned up after being dequeued.
      0752 806
      0752 807 ST_BLK_QUE:
      71'AF 9F 0752 808      PUSHAB B*150$    ; Set return address
50 7C AB B7 0755 809 120$: DECW UQB$W_NUM_BLK(R11) ; Drop the blocked count
      E8 A5 0F 0758 810      REMQUE CDRP$L_UQB_FL(R5),R0 ; Remove from blocked queue
      04 12 075C 811      BNEQ 130$        ; Not the last entry
      02 8A 075E 812      BICB #UQB$M_BLOCKED,- ; None left, clear flag
      2B AB 0760 813      UQB$B_FLAGS(R11)
      60 D4 0762 814 130$: CLRL (R0)        ; Clear link
```

```
05 0764 815 RSB ; Leave now
    0765 816
    0765 817
    0765 818 ; Online-copy blocked que: The CDRP is blocked. It has no resources
    0765 819 ; and can be immediately cleaned up after being dequeued.
    0765 820
    0765 821 ST_ONBLK QUE:
    0765 822 BLBS R9,160$ ; If second time, then
    0768 823 BSBB 120$ ; clean up
    076A 824 MOVQ CDRP$L_FR3(R5),R3 ; Pick up registers
    076E 825 BSBW CLEAN_OP_ONLCPY ; Do the special clean up
    0771 826
    0771 827
    0771 828 ; Clean up the CDRP and deallocate resources.
    0771 829
    09 FB A5 91 0771 830 150$: CMPB CDRP$B_OPCODE(R5), - ; If non-sequential command
    0775 831 #MSCP$K_OP_ONLIN
    0775 832 BLEQU 155$ ; then
    0777 833 UNMAP ; Get rid of possible map resource
    52 B0 A5 D0 077A 834 155$: MOVQ CDRP$L_MSGBUF_X(R5),R2 ; Pick up message buffer
    077E 835 BEQL 157$ ; Just in case!
    FB A2 18 A3 D1 0780 836 CMPL CDT$L_LCONID(R3), - ; Check validity of remote conid
    0785 837 SC$S$L_DST_CONID(R2)
    0785 838 BNEQ 165$ ; Trying to respond to stale connection
    B0 A5 D4 0787 839 CLRL CDRP$L_MSGBUF_X(R5) ; Show it gone
    078A 840 DEALLOC MSG_BUF_REG ; Deallocate it
    53 58 D0 078D 841 157$: MOVQ R8,R3 ; Set HQB address
    54 DD 0790 842 PUSHQ R4 ; Save PDT
    033F 30 0792 843 BSBW SEND_PKT ; Clean up the packet
    54 8ED0 0795 844 POPL R4 ; Restore PDT
    FF1C 31 0798 845 160$: BRW 60$ ; Continue
    079B 846
    079B 847 165$: BUG_CHECK MSCPSERV,FATAL ; ERROR
    079F 848
    079F 849 .DISABLE LSB
    079F 850
    079F 851 ; Clean up of CDRP's all done. Now finish up UNIT and HOST blocks.
    079F 852
    079F 853 FIN_HQB:
    5A 01 CE 079F 854 MNEGL #1,R10 ; Yes, Start at unit 0
    53 58 D0 07A2 855 10$: MOVQ R8,R3 ; Transfer HQB address
    5A D6 07A5 856 INCL R10 ; Next bit position
    51 1F 9A 07A7 857 MOVZBL #<2*MSUSK_MAX_UNIT>-1,R1 ; Max bit size
    51 5A C2 07AA 858 SUBL R10,R1 ; Number left
    5A 10 A8 51 5A EA 07AD 859 FFS R10,R1,-
    07B3 860 HQB$L_UNIT_ONLN(R8),R10 ; Find any units online
    07B3 861 BEQL 40$ ; No units alive
    54 F8D6 CF4A D0 07B5 862 MOVQ UNIT_VEC[R10],R4 ; Get unit que block
    05 12 07BB 863 BNEQ 20$ ; Got one
    E0 10 A3 5A E4 07BD 864 BBSC R10,- ; Clear the bogus bit
    07C2 865 HQB$L_UNIT_ONLN(R3),10$
    1474 30 07C2 866 20$: BSBW GET_STRUCT ; Get data structure
    04 50 E8 07C5 867 BLBS R0,30$ ; Branch on success
    07C8 868 BUG_CHECK MSCPSERV,FATAL ; It failed
    07CC 869
    7C A5 53 D0 07CC 870 30$: MOVQ R3,CDRP$L_HQB(R5) ; Address of HQB
    2C A3 B6 07D0 871 INCW HQB$W_REFCT(R3) ; Upreference count
```

```

      B0 A5 D4 07D3 872 CLRL CDRP$L MSGBUF(R5) ; Prevent spurious unloads
F8 A5 SA D0 07D6 873 MOVL R10,CDRPSW_SLOT(R5) ; Clean out flags and others
      08 90 07DA 874 MOVAB #MSCP$K_OP-AVAIL- ; Set the command
      FB A5 07DC 875 CDRP$B_OPCODE(R5)
78 A5 S4 D0 07DE 876 MOVL R4,CDRPSL_MST_UB(R5) ; Needed by routine
      03BB 30 07E2 877 BSBW SEQ ; Go do the AVAIL
      BB 11 07E5 878 BRB 10$ ; Loop until done
      07E7 879
      07E7 880 40$:
      07E7 881 ; ??? Insert bug trapping code here to find live references to the now stale CDT
53 14 A8 D0 07E7 882 MOVL HQB$L_CDT(R8),R3 ; CDT to look up
54 10 A3 D0 07EB 883 MOVL CDT$L_PDT(R3),R4 ; PDT address
50 0F AF 9E 07EF 884 MOVAB B^100$,R0 ; Action routine
00000000 GF 16 07F3 885 JSB G^SCSS$LKP_RDTCDRP ; Look for matching CDRPs on RDT
50 0F AF 9E 07F9 886 MOVAB B^100$,R0 ; Action routine
00000000 GF 16 07FD 887 JSB G^SCSS$LKP_RDTWAIT ; Look for matching CDRPs on RSPID wait queue
50 0F AF 9E 0803 888 MOVAB B^100$,R0 ; Action routine
00000000 GF 16 0807 889 JSB G^SCSS$LKP_MSGWAIT ; Look for matching CDRPs on message send queue
      04 11 080D 890 BRB 110$
      080F 891
      080F 892 100$: BUG_CHECK MSCPSERV,FATAL ; Found dangling CDT reference
      0813 893
      0813 894 110$:
      0813 895 ; ??? End of bug traps
      55 58 D0 0813 896 MOVL R8,R5 ; Hold HQB address
      OFD8 8F BA 0816 897 POPR #^M<R3,R4,R6,R7,R8,R9,R10,R11>
      081A 898 DISCONNECT
      0820 899 ; ??? more bug traps
14 A5 01 D0 0820 900 MOVL #1,HQB$L_CDT(R5) ; zap CDT address
      0824 901 ; ??? end of more bug traps
51 1E A5 9A 0824 902 MOVZBL HQB$B_VEC(R5),R1 ; Pick up vect number
F8E3 CF41 D4 0828 903 CLRL HOST_VEC[R1] ; Show us gone
F81D CF B6 082D 904 INCW NUM_HOST ; Up number of host slot left
53 55 D0 0831 905 MOVL R5,R3 ; HQB address
      2C A3 B7 0834 906 DECW HQB$W_REFC(R3) ; Decrement reference count of HQB
      06 12 0837 907 BNEQ 120$ ; Not yet 0 -- we never deallocate in this c
50 63 OF 0839 908 REMQUE HQB$L_FLINK(R3),R0 ; Remove from list headed by MSUSQ_HQBLIST
      FE0F 30 083C 909 BSBW DEALMEM ; Get rid of HQB
      05 083F 910 RSB
      0840 911 120$:

```

```
0840 913 .SBTTL WAIT FOR INT
0840 914
0840 915 :+
0840 916 :
0840 917 : The resuming address and R3/R4 are saved in the CDRP. Control
0840 918 : is returned to caller's caller.
0840 919 :
0840 920 : Input:
0840 921 : R3 --> HQB
0840 922 : R4 --> UQB
0840 923 : R5 --> CDRP
0840 924 :-
0840 925
0840 926 .ENABLE LSB
0840 927
0840 928 BLOCKEDH:
68 A4 E8 A5 0E 0840 929 INSQUE CDRP$L_UQB_FL(R5), - ; Insert this packet on blocked que
0845 930 UQB$Q_BLOCKQ(R4) ; at head
0845 931 BRB 10$
0847 932
0847 933 BLOCKEDT:
6C B4 E8 A5 0E 0847 934 INSQUE CDRP$L_UQB_FL(R5), - ; Insert this packet on blocked que
084C 935 @UQB$Q_BLOCKQ+4(R4) ; at tail
D6 A5 07 B0 084C 936 10$: MOVW #7,CDRP$W_STATE(R5) ; Ditto
0850 937 BISB #UQB$M_BLOCKED,- ; Set blocked flag
2B A4 02 88 0852 938 UQB$B_FLAGS(R4)
7C A4 B6 0854 939 INCW UQB$W_NUM_BLK(R4) ; Bump number of packet queue to unit
7C A4 B1 0857 940 CMPW UQB$W_NUM_BLK(R4),- ; Is this number <= max?
7E A4 085A 941 UQB$Q_MAX_BLK(R4)
05 1B 085C 942 BLEQU WAIT_FOR_INT ; Yes, skip
7C A4 B0 085E 943 MOVW UQB$Q_NUM_BLK(R4),- ; No, set new max
7E A4 0861 944 UQB$Q_MAX_BLK(R4)
0863 945 ; And fall thru to wait
0863 946 WAIT_FOR_INT:
10 A5 53 7D 0863 947 MOVQ R3,CDRP$L_FR3(R5) ; Save registers
OC A5 8ED0 0867 948 POPL CDRP$L_FPC(R5) ; Store resume point
05 086B 949 RSB ; Return to caller's caller
086C 950
086C 951 .DISABLE LSB
```

```
086C 953 .SBTTL MAIN LINE ROUTINE
086C 954
086C 955 :+
086C 956 :
086C 957 : This is the main line routine that does all the work.
086C 958 :
086C 959 : Inputs:
086C 960 :
086C 961 : R1 = Length of message
086C 962 : R2 --> Message
086C 963 : R3 --> CDT
086C 964 : R4 --> PDT
086C 965 :
086C 966 :-
086C 967
086C 968 .ENABLE LSB
086C 969
18 BB 086C 970 10$: PUSH R3,R4 ; Save vital registers
086E 971 DEALLOC MSG_BUF_REG ; Deallocate message buffer
18 BA 0871 972 POP R3,R4 ; Restore registers
FDF4 31 0873 973 BRW VC_ERR ; Break connection and return
0876 974
0876 975 MSG_IN:
13C0 30 0876 976 BSBW GET_STRUCT ; Get data structure
FO 50 E9 0879 977 BLBC R0,T0$ ; Failed to get one
F930 CF D6 087C 978 INCL OPC_COUNT ; Grand total of requests
B0 A5 52 D0 0880 979 MOVL R2,CDRPSL_MSGBUF(R5) ; Store the message pointer
1C A5 D4 0884 980 CLRL CDRPSL_MSG_BUF(R5) ; Mark it not in use
24 A5 53 D0 0887 981 MOVL R3,CDRPSL_CDT(R5) ; and the CDT
60 A5 53 D0 088B 982 MOVL R3,CDRPSL_CDTX(R5) ; Store CDT in non-volatile cell
53 5C A3 D0 088F 983 MOVL CDTSL_AUXSTRUC(R3),R3 ; Pick up the HQB address
7C A5 53 D0 0893 984 MOVL R3,CDRPSL_HQB(R5) ; Store HQB address
2C A3 B6 0897 985 INCW HQBSW_REF(R3) ; Up reference count
28 A3 B6 089A 986 INCW HQBSW_NUM_QUE(R3) ; Bump number of packet queue to host
28 A3 B1 089D 987 CMPW HQBSW_NUM_QUE(R3),- ; Is this number <= max?
2A A3 08A0 988 HQBSW_MAX_QUE(R3)
05 1B 08A2 989 BLEQU 30$ ; Yes, skip
28 A3 B0 08A4 990 MOVW HQBSW_NUM_QUE(R3),- ; No, set new max
2A A3 08A7 991 HQBSW_MAX_QUE(R3)
08A9 992
08A9 993 : Check the MSCP command packet for protocol errors
08A9 994
0C 51 B1 08A9 995 30$: CMPW R1,MSCP$K_MIN_SIZE ; Is it the min length?
17 19 08AC 996 BLSS 40$ ; No, error
50 08 A2 9A 08AE 997 MOVZBL MSCP$B_OPCODE(R2),R0 ; Pick up the opcode
75 13 08B2 998 BEQL ERR_1 ; None is error
40 8F 50 91 08B4 999 CMPB R0,MSUSK_MAX_OPC ; Out of valid range?
6F 14 08B8 1000 BGTR ERR_1 ; Yes, error
54 1CFC'CF40 9A 08BA 1001 MOVZBL COM_PKT_LEN[R0],R4 ; Get expected length
67 13 08C0 1002 BEQL ERR_1 ; Invalid command
54 51 B1 08C2 1003 CMPW R1,R4 ; Big enough?
68 19 08C5 1004 40$: BLSS ERR_2 ; No, error
OA A2 1D4C'CF40 B3 08C7 1005 BITW MOD_TBL[R0],-
08CE 1006 MSCP$W_MODIFIER(R2) ; Check allowable
63 12 08CE 1007 BNEQ ERR_3 ; Non-allow bit set, error
F4 A5 02 CE 08D0 1008 MNEGL #2,CDRPSL_CMD_STS(R5) ; Set up initial cmd status
08D4 1009 :
```

```
08D4 1010 ; Grab the unit number and try to find a UQB for it
08D4 1011 ;
51 52 D0 08D4 1012 ; Transfer to a better reg
52 04 A1 3C 08D7 1013 ; Pick up unit number from command
50 52 D0 08DB 1014 ; Copy
54 52 FFFFFFFF 8F CB 08DE 1015 ; R2,R4 ; Slot
52 F7A5 CF44 D0 08E6 1016 ; Look for unit block in slot
07 13 08EC 1017 ; Empty
04 A1 B1 08EE 1018 ;
28 A2 08F1 1019 ; Hit, is it a match?
04 13 08F3 1020 ; Yes
65 10 08F5 1021 50$: BSBB FIND SLOT ; No, try more probing
54 19 08F7 1022 ; Still did not find it
78 A5 52 D0 08F9 1023 60$: MOVL R2,CDRPSL_MST_UQB(R5) ; Save UQB pointer
F8 A5 54 3C 08FD 1024 65$: MOVZWL R4,CDRPSW_SLOT(R5) ; Save slot in CDRP and zero FLAGS
50 08 A1 9A 0901 1025 ; Pick up the opcode
FB A5 50 90 0905 1026 ; Save opcode in CDRP
F8A2 CF40 D6 0909 1027 ; Count the opcode
04 1F A3 E9 090E 1028 ; Is this the passthru driver?
20 88 0912 1029 ; Yes, set flag
FA A5 0914 1030 ; CDRPSB_FLAGS(R5)
0916 1031 ;
0916 1032 ; Split up command into class then dispatch on it.
50 50 FD 8F 78 0916 1033 70$: ASHL #-3,R0,R0 ; Pick up class field
091B 1035 CASE R0,- ; Dispatch on class
091B 1036 <IMMED,- ; 0 - IMMEDIATE
091B 1037 SEQ,- ; 8 - SEQUENTIAL
091B 1038 NSEQ,- ; 16 - NON-SEQUENTIAL
091B 1039 MAINT,- ; 24 - MAINTENANCE
091B 1040 NSEQB> ; 32 - NON-SEQUENTIAL/BUFFER
0929 1041 ;
0929 1042 ; Error paths
0929 1043 ;
50 08 18 9C 0929 1044 ERR_1: ROTL #24,#MSCPSB_OPCODE,R0 ; Opcode error
08 11 092D 1045 BRB ERR_X
092F 1046 ;
50 D4 092F 1047 ERR_2: CLRL R0 ; Illegal length
04 11 0931 1048 BRB ERR_X
0933 1049 ;
50 0A 18 9C 0933 1050 ERR_3: ROTL #24,#MSCPSW_MODIFIER,R0 ; Modifier error
52 B0 A5 D0 0937 1051 ERR_X: MOVL CDRPSL_MSGBUF(R5),R2 ; Recover message buffer
C9 093B 1052 BISL3 #<MSCPSK_ST_ICMD@16- ; Set invalid
093C 1053 !MSCPSK_OP_END>,-
093C 1054 R0,MSCPSB_OPCODE(R2)
0944 1055 CLRL CDRPSL_MST_UQB(R5) ; No master UQB
51 0C D0 0947 1056 MOVL #MSCPSK_MIN_SIZE,R1 ; Length of message
01A0 31 094A 1057 BRW SEND_END
094D 1058 ;
094D 1059 NOT_FOUND:
78 A5 D4 094D 1060 CLRL CDRPSL_MST_UQB(R5) ; Zero out UQB pointer
08 A1 91 0950 1061 CMPB MSCPSB_OPCODE(R1),- ; Command need unit?
08 0953 1062 #MSCPSK_OP_AVAIL
A7 1F 0954 1063 BLSSU 65$ ; No, allow to continue
50 03 D0 0956 1064 MOVL #MSCPSK_ST_OFFLN,R0 ; Yes, set failure status
0178 31 0959 1065 BRW SEND_PKT ; And leave
095C 1066
```

```

095C 1067 .DISABLE LSB
095C 1068 :
095C 1069 : Routine to find slot number from unit number
095C 1070 :
095C 1071 : R0 = Unit number to match against
095C 1072 :
095C 1073 FIND_SLOT:
52 54 10 C0 095C 1074 ADDL #MSUSK_MAX_UNIT,R4 ; Try for second probe
F72C CF44 D0 095F 1075 MOVL UNIT_VEC[R4],R2 ; Look for unit block in slot
06 13 0965 1076 BEQL 60$ ; Empty
28 A2 50 B1 0967 1077 CMPW R0,UQBSW_UNIT(R2) ; Hit, is it a match?
1B 13 0968 1078 BEQL 100$ ; Yes
54 1F D0 096D 1079 60$: MOVL #<2*MSUSK_MAX_UNIT>-1,R4 ; Sequential search
52 F71B CF44 D0 0970 1080 70$: MOVL UNIT_VEC[R4],R2 ; Pick up unit
06 13 0976 1081 BEQL 80$ ; Empty
28 A2 50 B1 0978 1082 CMPW R0,UQBSW_UNIT(R2) ; Hit, is it a match?
0A 13 097C 1083 BEQL 100$ ; Yes
54 D7 097E 1084 80$: DECL R4 ; Loop thru
10 54 D1 0980 1085 CMPL R4,#MSUSK_MAX_UNIT ; Reached end?
EB 1B 0983 1086 BGEQ 70$ ; No, try again
54 01 CE 0985 1087 90$: MNEGL #1,R4 ; Set a failure indicator
05 0988 1088 100$: RSB

```

```

0989 1090 .SBTTL IMMEDIATE class commands
0989 1091 :
0989 1092 :
0989 1093 : R0 = scratch
0989 1094 : R1 = Ptr to message buffer
0989 1095 : R2 = low 3 bits of command
0989 1096 : R3 = Ptr to HQB
0989 1097 : R4 = Unit number
0989 1098 : R5 = Ptr to CDRP
0989 1099 IMMED: CASE CDRP$B_OPCODE(R5),- ; Dispatch on command subtype
0989 1100 TYPE=B,LIMIT=#MSCP$K_OP_ABORT-
0989 1101 <ABORT,- ; 1-ABORT
0989 1102 GET_CMD_STS,- ; 2-GET COMMAND STATUS
0989 1103 GET_UNIT_STS,- ; 3-GET UNIT STATUS
0989 1104 SET_CTRL_CHR> ; 4-SET CONTROLLER CHARACTERISTICS
91 11 0996 1105 BRB ERR_T ; Illegal command

```

```
0998 1107 .SBTTL - 1 - ABORT
0998 1108 *****
0998 1109 *****
0998 1110 # 1 - A B O R T
0998 1111 *****
0998 1112 *****
0998 1113
0998 1114 ABORT:
121D 30 0998 1115 BSBW SEARCH_QUE : Find the entry
73 50 E9 0998 1116 BLBC RO,SS : Failed to find it
099E 1117 CASE CDRP$W_STATE-CDRPSL_UQB_FL(R1),TYPE=W-
099E 1118 <10$,- : 0 - Mem wait
099E 1119 20$,- : 1 - Map wait
099E 1120 40$,- : 2 - Driver que wait
099E 1121 40$,- : 3 - Data command wait
099E 1122 40$,- : 4 - Driver que wait
099E 1123 20$,- : 5 - Msg send wait
099E 1124 50$,- : 6 - Special holding que
099E 1125 70$,- : 7 - Blocked que
099E 1126 70$> : 8 - Online-copy blocked que
09B5 1127
09B5 1128 BUG_CHECK MSCPSERV,FATAL : ERROR
09B9 1129
09B9 1130 : CDRP was on the memory wait queue. Remove it and post it.
09B9 1131
F6A3 CF B7 09B9 1132 10$: DECW NUM_QUE
09BD 1133
09BD 1134 : CDRP was on the map resource wait queue. Remove it and post it.
09BD 1135
50 18 A1 0F 09BD 1136 20$: REMQUE CDRPSL_FQFL- -
09C1 1137 CDRPSL_UQB_FL(R1),RO : Remove from wait queue
18 A1 9F 09C1 1138 30$: PUSHAB -CDRPSL_UQB_FL(R1) : Push addr of CDRP to be aborted
4B 10 09C4 1139 BSBB SS : Return the ABORT end packet
55 BED0 09C6 1140 POPL R5 : Recover aborted CDRP pointer
09 FB A5 91 09C9 1141 CMPB CDRPSB_OPCODE(R5), - : If non-sequential command
03 1B 09CD 1142 #MSCPSK_OP_ONLIN
09CF 1143 BLEQU 35$ : then
50 02 D0 09D2 1145 35$: MOVL #MSCPSK_ST_ABRTD, RO : Get rid of possible map resource
52 B0 A5 D0 09D5 1146 MOVL CDRPSL_MSGBUF(R5),R2 : Set status
E0 A5 D0 09D9 1147 MOVL CDRPSL_ABCNT(R5), - : Recover message pointer
OC A2 09DC 1148 MSCPSL_BYTE_CNT(R2) : Show accumulated byte count
00F3 31 09DE 1149 BRW SEND_PKT : Return the command end packet
09E1 1150
09E1 1151 : CDRP was a activity pending queue. We cannot arbitrarily remove it.
09E1 1152 : Therefore, just set the "ABORT" bit and let the action routine deal with
09E1 1153 : it when it returns.
09E1 1154
12 10 88 09E1 1155 40$: BISB #1@CDRPSV_ABORT,- : Set 'ABORTED' bit
A1 09E3 1156 CDRPSB_FLAGS-CDRPSL_UQB_FL(R1)
2A 11 09E5 1157 BRB SS : Leave with success
09E7 1158
09E7 1159 : CDRP was on the special holding queue. Remove it and post it.
09E7 1160
50 61 0F 09E7 1161 50$: REMQUE (R1),RO : Remove from special holding que
04 12 09EA 1162 BNEQ 60$ : Not the last entry
08 8A 09EC 1163 BICB #UQB$M_CMDPEND,- : Clear the command pending status
```

```
2B A4 09EE 1164 UQBSB_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$: DECW UQBSW_NUM_BLK(R4) ; Drop the blocked count
50 61 0F 09F7 1171 REMQUE (R1),R0 ; Remove from blocked queue
    F4 12 09FA 1172 BNEQ 60$ ; Not the last entry
    02 8A 09FC 1173 BICB #UQBSM_BLOCKED,- ; Clear blocked flag
2B A4 09FE 1174 UQBSB_FLAGS(R4)
EE 11 0A00 1175 BRB 60$ ; And finish up
```

```

      0A02 1177 .SBTTL - 2 - GET COMMAND STATUS
      0A02 1178 :*****
      0A02 1179 :
      0A02 1180 : # 2 - G E T C M D S T S
      0A02 1181 :
      0A02 1182 :*****
      0A02 1183 :
      0A02 1184 GET_CMD_STS:
54 11B3 30 0A02 1185 BSBW SEARCH_QUE ; Find entry
    09 50 E9 0A05 1186 BLBC R0, SS ; None, always return success
    B0 A5 D0 0A08 1187 MOVL CDRP$ MSGBUF(R5), R4 ; Pick up buffer address
    0C A1 D0 0A0C 1188 MOVL CDRP$ CMD_STS-CDRP$ UQB_FL(R1),- ; Found it,
    10 A4 0A0F 1189 MSCP$ CMD_STS(R4) ; copy the status code
    00BE 31 0A11 1190 SS: BRW SEND_SUCCESS
```

```
0A14 1192 .SBTTL - 3 - GET UNIT STATUS
0A14 1193 *****
0A14 1194 :
0A14 1195 # 3 - G E T U N I T S T S
0A14 1196 :
0A14 1197 *****
0A14 1198
0A14 1199 GET_UNIT_STS:
0A14 1200 ASSUME MSCP$V_MD NXUNT EQ 0
1201 BLBC MSCP$W_MODIFIER(R1),50$ ; If not NXTUNT, do staight processing
52 2E 0A A1 E9 0A14 1202 MOVZWL MSCP$W_UNIT(R1),R2 ; Pick up orginal unit number
54 52 04 A1 3C 0A18 1203 MOVAL UQB_LIST,R4 ; Point to head of list
54 F660 CF DE 0A1C 1204 CMPL (R4),R4 ; Queue empty?
54 54 64 D1 0A21 1205 BEQL 20$ ; Yes, return OFFLINE failure, unit=0
50 54 08 13 0A24 1206 MOVL R4,R0 ; No, copy starting point
54 64 D0 0A29 1207 10$: MOVL (R4),R4 ; Get next entry
50 54 D1 0A2C 1208 CMPL R4,R0 ; End of line?
09 12 0A2F 1209 BNEQ 40$ ; No, continue
04 A1 B4 0A31 1210 20$: CLRW MSCP$W_UNIT(R1) ; Yes, give special code (Unit 0)
50 03 D0 0A34 1211 30$: MOVL #MSCP$R_ST_OFFLN,R0 ; Set for OFFLINE
009A 31 0A37 1212 BRW SEND_PKT ; Return end packet
0A3A 1213
A8 A4 52 B1 0A3A 1214 40$: CMPW R2,UQB$W_UNIT- ; Compare unit numbers
0A3E 1215 UQB$Q_UNITQ(R4) ; Not far enough, keep looping
54 80 E9 1A 0A3E 1216 BGTRU 10$ ; Set R4 to head of UQB
54 08 DE 0A40 1217 MOVAL -UQB$Q_UNITQ(R4),R4 ; Done
08 11 0A44 1218 BRB 60$
0A46 1219
54 F645 CF44 D0 0A46 1220 50$: MOVL UNIT_VEC[R4], R4 ; Get pointer to UQB
E6 13 0A4C 1221 BEQL 30$ ; None, return status
0A4E 1222
0A4E 1223 : Found UQB for unit, transfer information
0A4E 1224
0A4E 1225 60$: MOVW UQB$W_UNIT(R4) -
04 A1 80 0A51 1226 MSCP$W_UNIT(R1) ; Return the unit
2F A1 94 0A53 1227 CLRB MSCP$B_RCT CPYS(R1) ; Assume no RCT
50 30 A4 D0 0A56 1228 MOVL UQB$Q_UCB(R4),R0 ; Get device UCB
13 13 0A5A 1229 BEQL 80$ ; None
44 A0 9B 0A5C 1230 MOVZBW UCB$B_SECTORS(R0),- ; Transfer number of sectors
24 A1 0A5F 1231 MSCP$W_TRACK(R1) ; per track.
45 A0 9B 0A61 1232 MOVZBW UCB$B_TRACKS(R0),- ; Number of tracks per group.
26 A1 0A64 1233 MSCP$W_GROUP(R1)
04 38 A0 08 E1 0A66 1234 BBC #DEVS$V_RCT, - ; Branch if no RCT
0A6B 1235 UCB$Q_DEVCHAR(R0),80$
2F A1 01 90 0A6B 1236 MOVW #1,MSCP$B_RCT CPYS(R1) ; Mark RCT present
28 A1 01 D0 0A6F 1237 80$: MOVL #1,MSCP$W_CYLINDER(R1) ; Number of groups per cylinder.
2C A1 B4 0A73 1238 CLRW MSCP$W_RCT_SIZE(R1) ; No disk has these
2E A1 94 0A76 1239 CLRB MSCP$B_RBNS(R1)
50 0E A4 3C 0A79 1240 MOVZWL UQB$W_STATUS(R4),R0 ; Pick up unit status
OC 0C 12 0A7D 1241 BNEQ FIN_UNIT_STS ; Either OFFLINE or AVAIL
52 2A A4 9A 0A7F 1242 MOVZBL UQB$B_SLOT(R4),R2 ; Pick up slot number
10 A3 52 E0 0A83 1243 BBS R2,UQB$Q_UNIT_ONLN(R3),- ; If not already ONLINE to this host
50 03 0A87 1244 FIN_UNIT_STS ; then it is AVAIL to it
04 04 3C 0A88 1245 MOVZWL #MSCP$K_ST_AVLBL,R0
0A8B 1246
0C A1 10 A4 D0 0A8B 1247 FIN_UNIT_STS:
0A8B 1248 MOVL UQB$W_MULT_UNT(R4), -
```

MSCP  
V04-001

- MSCP Server - Emulator  
- 3 - GET UNIT STATUS

M 14

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 29  
(1)

MS  
VO

|    |    |    |    |    |      |      |      |
|----|----|----|----|----|------|------|------|
| 14 | A1 | 18 | A4 | 7D | 0A90 | 1249 |      |
|    |    |    |    |    | 0A90 | 1250 | MOVQ |
|    |    |    |    |    | 0A95 | 1251 |      |
|    |    | 20 | A4 | 7D | 0A95 | 1252 | MOVQ |
|    |    | 1C | A1 |    | 0A98 | 1253 |      |
|    |    | 38 |    | 11 | 0A9A | 1254 | BRB  |

|                      |                                          |
|----------------------|------------------------------------------|
| MSCPSW MULT UNT(R1)  | ; Copy mult_unit & unit_flags            |
| UQBSQ UNIT_ID(R4) -  |                                          |
| MSCPSQ UNIT_ID(R1)   | ; Copy unit identifier                   |
| UQBSL MEDIA_ID(R4) - |                                          |
| MSCPSL MEDIA_ID(R1)  | ; Copy media identifier & shdw unit/flgs |
| SEND_PKT             |                                          |

```
0A9C 1256 .SBTTL - 4 - SET CONTROLLER CHARACTERISTICS
0A9C 1257 :*****
0A9C 1258 :
0A9C 1259 : # 4 - S E T C T R L C H R
0A9C 1260 :
0A9C 1261 :*****
0A9C 1262 :
0A9C 1263 SEND_MISMATCH:
0A9C 1264 BUG_CHECK MSCPSERV,FATAL ; Responding using stale CDT address
0AA0 1265
0AA0 1266 SET_CTRL CHR:
F6F2 CF 0C A1 B1 0AA0 1267 CMPW MSCPSW_VERSION(R1), -
0AA6 1268 VERSION ; Does the version match?
0AA6 1269 BEQL 10$ ; Yes, okay
50 0C 07 13 0AA6 1269 BEQL 10$ ; No, set sub status
FE88 31 0AAC 1270 ROTL #24, MSCPSW_VERSION, R0 ; Send invalid command packet
0AAF 1271 BRW ERR_X
18 A3 0E A1 90 0AAF 1272 10$: MOVW MSCPSW_CNT_FLGS(R1), -
0AB4 1273 HQB$W_CNT_FLGS(R3) ; Transfer Low byte
0AB4 1274 MOVW MSCPSW_HST_TMO(R1), -
0AB7 1275 HQB$W_HST_TMO(R3)
20 A3 1A A3 7D 0AB9 1276 MOVQ MSCPSQ_TIME(R1), -
0ABE 1277 HQB$Q_TIME(R3) ; ... and time
0ABE 1278
0ABE 1279 ; Return controller information to host
0ABE 1280
0ABE 1281
00000000'9F 90 0ABE 1282 MOVW @#CLUSGL_ALLOCLS, - ; Transfer the allocation class
04 A1 04 A1 7D 0AC4 1283 MSCPSB_CNT_ALCS(R1)
OC A1 F6CE CF 7D 0AC6 1284 MOVQ CTRL_INFO, =
0ACC 1285 MSCPSW_VERSION(R1) ; Send back info
14 A1 F6C0 CF 7D 0ACC 1286 MOVQ CTRL_ID, -
0AD2 1287 MSCPSQ_CNT_ID(R1) ; ... and ID
0AD2 1288
0AD2 1289 SEND_SUCCESS:
50 D4 0AD2 1290 CLRL R0 ; Set success
0AD4 1291
0AD4 1292 SEND_PKT:
52 B0 A5 D0 0AD4 1293 MOVL CDRP$L_MSGBUF(X(R5)), R2 ; Pick up message buffer ptr
47 13 0AD8 1294 BEQL CLEAN_OP ; No message buffer
OA A2 50 B0 0ADA 1295 MOVW R0, MSCPSW_STATUS(R2) ; Set the status
0ADE 1296
0ADE 1297 SEND_PKT_STS:
08 A2 08 A2 9A 0ADE 1298 MOVZBL MSCPSB_OPCODE(R2), R1 ; Pick up opcode
80 8F 88 0AE2 1299 BISB2 #MSCPSB_OP_END, -
51 1D24'CF41 9A 0AE7 1300 MSCPSB_OPCODE(R2) ; Make an end packet
0AED 1301 MOVZBL END_PKT_LEN[R1], R1 ; Pick up length
0AED 1302
0AED 1303 SEND_END:
1C A5 52 D0 0AED 1304 MOVL R2, CDRP$L_MSG_BUF(R5) ; Stuff the message buffer address
54 60 A5 D0 0AF1 1305 MOVL CDRP$L_CDTX(R5), R4 ; Pick up the CDT address
24 A5 54 D0 0AF5 1306 MOVL R4, CDRP$L_CDT(R5) ; Store into standard location
F8 A2 18 A4 D1 0AF9 1307 CMPL CDT$L_LCONID(R4), - ; Check validity of remote conid
0AFE 1308 SC$S$L_DST_CONID(R2)
D6 A5 9C 12 0AFE 1309 BNEQ SEND_MISMATCH ; Trying to respond to stale connection
53 5C A4 D0 0B00 1310 MOVW #5, CDRP$W_STATE(R5) ; Message send wait state
54 10 A4 D0 0B04 1311 MOVL CDT$L_AUXSTRUC(R4), R3 ; Pick up HQB address
0B08 1312 MOVL CDT$L_PDT(R4), R4 ; Set up PDT address
```

```
AC A5 51 D0 0B0C 1313      MOVL R1,CDRPSL_PID(R5)      ; Save the length
                                0B10 1314      RECYCL_MSG_BUF      ; Recycle the message
51 AC A5 D0 0B13 1315      MOVL CDRPSL_PID(R5),R1      ; Restore the length
                                0B17 1316      SEND_CNT MSG_BUF      ; Send the message
53 7C A5 D0 0B1A 1317      MOVL CDRPSL_HQB(R5),R3      ; Get HQB address
    28 A3 B7 0B1E 1318      DECW HQBSW_NUM_QUE(R3)      ; Drop count on host
                                0B21 1319
                                0B21 1320      .ENABLE LSB
08 FB A5 91 0B21 1321      CLEAN_UP: CMPB CDRPSB_OPCODE(R5), - ; For immediate commands
                                0B25 1323      #MSCPSK_OP_AVAIL
                                0B25 1324      BLSSU 5$      ; Save CDRP address
                                0B27 1325      PUSHL R5      ; Finish cleaning up
                                0E 10 0B29 1326      BSBW 20$      ; Restore CDRP address
53 7C A5 D0 0B2E 1327      5$: MOVL CDRPSL_HQB(R5),R3      ; Get HQB address
    2C A3 B7 0B32 1329      DECW HQBSW_REFC(R3)      ; Decrement HQB reference count
    1160 30 0B35 1330      BSBW FREE_STRUCT      ; Free data structure
                                05 0B38 1331      10$: RSB      ; Just leave
                                0B39 1332
54 78 A5 D0 0B39 1333      20$: MOVL CDRPSL_MST_UQB(R5),R4      ; Pick up UQB pointer
    F9 13 0B3D 1334      BEQL 10$      ; There is none
    E8 A5 D5 0B3F 1335      TSTL CDRPSL_UQB_FL(R5)      ; Is this CDRP queued on a UQB?
    0A 13 0B42 1336      BEQL 30$      ; No
50 E8 A5 0F 0B44 1337      REMQUE CDRPSL_UQB_FL(R5),R0      ; Remove from unit que
    E8 A5 D4 0B48 1338      CLRL CDRPSL_UQB_FL(R5)      ; Prevent double remque - if this code
    50 A4 B7 0B4B 1340      DECW UQBSW_NUM_QUE(R4)      ; is moved, always get rid of link!!!
                                0B4E 1341      ; Drop count on unit
                                0B4E 1342      ; Check for any buffer resource and return it. If possible start up any
                                0B4E 1343      ; thread waiting for buffer.
                                0B4E 1344
FA A5 02 93 0B4E 1345      30$: BITB #1@CDRPSV_BUFFER, - ; Check for buffer
                                0B52 1346      CDRPSB_F[AGS(R5)
                                0B52 1347      BEQL 40$      ; None, skip next
                                0777 30 0B54 1348      BSBW DEALLOC_BUFF      ; Yes, get rid of it
    F505 CF B5 0B57 1349      TSTW NUM_QUE      ; Anyone waiting?
                                13 0B5B 1350      BEQL 40$      ; No, leave now
55 F517 DF 0F 0B5D 1351      REMQUE @MEM_WAIT,R5      ; Grab first entry
                                0C 1D 0B62 1352      BVS 40$      ; None
                                54 DD 0B64 1353      PUSHL R4      ; Save context
53 10 A5 7D 0B66 1354      MOVQ CDRPSL_FR3(R5),R3      ; Restore old context
    0C B5 16 0B6A 1355      JSB @CDRPSL_FPC(R5)      ; Finish off the pkt
    54 8ED0 0B6D 1356      POPL R4      ; Restore old, old context
                                0B70 1357
                                0B70 1358      ; If this is last command on execute queue, then check for any blocked
                                0B70 1359      ; commands. If there are some, start up the first one. If the SEQ bit
                                0B70 1360      ; is still set, we are in single command mode, so do not fetch any more
                                0B70 1361      ; commands, otherwise drain the blocked queue.
                                0B70 1362
50 A4 B5 0B70 1363      40$: TSTW UQBSW_NUM_QUE(R4)      ; Any entries queued?
    C3 12 0B73 1364      BNEQ 10$      ; Yes, leave
                                0B75 1365
                                0B75 1366      .DISABLE LSB
07C A4 B5 0B75 1367      DRAIN_BLOCKQ:
    25 13 0B75 1368      10$: TSTW UQBSW_NUM_BLK(R4)      ; Any entries queued?
                                0B78 1369      BEQL 30$      ; No, leave
```

|    |    |    |      |      |      |             |                          |   |                         |
|----|----|----|------|------|------|-------------|--------------------------|---|-------------------------|
| 55 | 68 | B4 | 0F   | 0B7A | 1370 | REMQUE      | @UQBSQ_BLOCKQ(R4),R5     | : | Remove from blocked que |
|    |    | 1F | 1D   | 0B7E | 1371 | BVS         | 30\$                     | : | None, leave             |
|    | 7C | A4 | B7   | 0B80 | 1372 | DECW        | UQBSW_NUM_BLK(R4)        | : | Drop count              |
|    |    | 04 | 12   | 0B83 | 1373 | BNEQ        | 20\$                     | : | Is it the last?         |
|    |    | 02 | 8A   | 0B85 | 1374 | BICB        | #UQBSM_BLOCKED,-         | : | Yes, clear the flag     |
|    | 2B | A4 |      | 0B87 | 1375 |             | UQBSB_FLAGS(R4)          |   |                         |
| 55 | 1B | A5 | DE   | 0B89 | 1376 | 20\$: NOVAL | CDRPSL_FQFL-             |   |                         |
|    |    |    |      | 0B8D | 1377 |             | CDRPSL_UQB_FL(R5),R5     | : | Set new CDRP ptr        |
|    |    | 54 | DD   | 0B8D | 1378 | PUSHL       | R4                       | : | Save old UQB            |
| 53 | 10 | A5 | 7D   | 0B8F | 1379 | MOVQ        | CDRPSL_FR3(R5),R3        | : | Restore old context     |
|    | 0C | B5 | 16   | 0B93 | 1380 | JSB         | @CDRPSL_FPC(R5)          | : | Finish off the pkt      |
|    |    | 54 | 8ED0 | 0B96 | 1381 | POPL        | R4                       | : | Restore old UQB         |
|    |    | 30 | 93   | 0B99 | 1382 | BITB        | #<UQBSM_SEQ.UQBSM_RIP>,- | : | Check if in SEQ mode    |
|    | 2B | A4 |      | 0B9B | 1383 |             | UQBSB_FLAGS(R4)          |   |                         |
|    |    | D6 | 13   | 0B9D | 1384 | BEQL        | 10\$                     | : | No, get next command    |
|    |    |    | 05   | 0B9F | 1385 | 30\$: RSB   |                          |   |                         |

```
.SBTTL SEQUENTIAL class commands

OBA0 1387
OBA0 1388
OBA0 1389 : This section of code sets up for sequential command processing. First
OBA0 1390 : we make checks for sequentiality.
OBA0 1391 : a) If commands are already blocked, then this command must be blocked
OBA0 1392 : also to preserve sequentiality. Code at label CLEAN_UP will
OBA0 1393 : unravel all of this
OBA0 1394 : b) If there are any non-sequential commands on the processing queue,
OBA0 1395 : then this command must be blocked and the blocked flag set
OBA0 1396 : in the UQB.
OBA0 1397 : c) EXCEPT that an ONLINE or AVAILABLE to a shadow master in the
OBA0 1398 : copy-in-progress phase will be let thru, since that phase
OBA0 1399 : is not strictly sequential.
OBA0 1400
OBA0 1401 : R0 scratch
OBA0 1402 : R1 = Ptr to message buffer
OBA0 1403 : R2 = low 3 bits of command
OBA0 1404 : R3 = Ptr to HQB
OBA0 1405 : R4 = Unit slot number
OBA0 1406 : R5 = Ptr to CDRP
OBA0 1407

FB A5 91 OBA0 1408 SEQ: CMPB CDRPSB_OPCODE(R5),- : Check opcode now
OB OBA3 1409 : #MSCPSK_CP_DTACP
60 1A OBA4 1410 BGTRU 70$ : Out of range - error
03 1F OBA6 1411 BLSSU 10$ : In range, continue
0794 31 OBA8 1412 BRW DET_ACC_PATH : DAP is a nop
OBA8 1413

54 78 A5 D0 OBA8 1414 10$: MOVL CDRPSL_MST_UQB(R5),R4 : Get UQB pointer
93 OBAF 1415 BITB #<UQBSM_BLOCKED!- : If already blocked,
OBB0 1416 : UQBSM_RIP!- : or in a repair
OBB0 1417 : UQBSM_SEQ>,- : or in sequential mode
2B A4 32 OBB0 1418 : UQBSB_FLAGS(R4) : block this cmd
OF 12 OBB3 1419 BNEQ 30$
09 E1 OBB5 1420 BBC #MSCPSV_UF_SSMST,- : If not shadow master, skip test
05 12 A4 OBB7 1421 : UQBSW_ONT_FLGS(R4),20$
00 E0 OBBA 1422 BBS #UQBSV_CIP,- : If CIP, then allow thru
1A 2B A4 OBBC 1423 : UQBSB_FLAGS(R4),50$
50 A4 B5 OBBF 1424 20$: TSTW UQBSW_NUM_QUE(R4) : Any entries queued?
15 13 OBC2 1425 BEQL 50$ : No, leave now
10 88 OBC4 1426 30$: BISB #UQBSM_SEQ,- : Yes, set for sequential mode
2B A4 OBC6 1427 : UQBSB_FLAGS(R4) : (rundown the IO)
OBC8 1428 BLOCKT : and block the command until done
OBCB 1429
OBCB 1430 : Just in case we get restarted and there are entries left in progress, we
OBCB 1431 : have to wait, but we also have to block any new entries.
OBCB 1432
50 A4 B5 OBCB 1433 40$: TSTW UQBSW_NUM_QUE(R4) : Any entries queued?
09 13 OBCF 1434 BEQL 50$ : No, leave now
10 88 OBD0 1435 BISB #UQBSM_SEQ,- : Yes, set sequential mode while waiting
2B A4 OBD2 1436 : UQBSB_FLAGS(R4)
OBD4 1437 BLOCKH : Block the command
F2 11 OBD7 1438 BRB 40$ : Command is resumed, check again
OBD9 1439
E8 A5 D4 OBD9 1440 50$: CLRL CDRPSL_UQB_FL(R5) : So no spurious remques
10 8A OBDC 1441 : #UQBSM_SEQ,- : Clear the flag in event of an
2B A4 OBDE 1442 : UQBSB_FLAGS(R4) : error exit
50 03 D0 OBE0 1443 MOVL #MSCPSK_ST_OFFLN,R0 : Assume offline status
```

|       |    |    |    |      |      |        |                                |                                   |
|-------|----|----|----|------|------|--------|--------------------------------|-----------------------------------|
| 50    | OE | A4 | 3C | OBE3 | 1444 | MOVZWL | UQBSW_STATUS(R4),R0            | ; Pick up status from unit        |
|       |    | OC | 12 | OBE7 | 1445 | BNEQ   | 60\$                           | ; Either OFFLINE or AVAIL         |
| 51    | 2A | A4 | 9A | OBE9 | 1446 | MOVZBL | UQBSB_SLOT(R4),R1              | ; Recover slot number             |
| 03 10 | A3 | S1 | E0 | OBE0 | 1447 | BBS    | R1,HQBSL_UNIT_ONLN(R3),60\$    | ; If not already ONLINE this host |
|       | 50 | 04 | D0 | 0BF2 | 1448 | MOVL   | #MSCPSK_ST_AVAIL,R0            | ; then it is AVAIL                |
| 51    | B0 | A5 | D0 | 0BF5 | 1449 | MOVL   | CDRPSL_MSGBUF(R5),R1           | ; Pick up message buffer          |
|       |    |    |    | 0BF9 | 1450 | CASE   | CDRPSB_OPCODE(R5),-            | ; Dispatch on command subtype     |
|       |    |    |    | 0BF9 | 1451 |        | TYPE=B,LIMIT=#MSCPSK_OP_AVAIL- |                                   |
|       |    |    |    | 0BF9 | 1452 |        | <AVAIL,-                       | ; 8 - AVAILABLE                   |
|       |    |    |    | 0BF9 | 1453 |        | ONLINE,-                       | ; 9 - ONLINE                      |
|       |    |    |    | 0BF9 | 1454 |        | SET_UNIT_CHR,-                 | ; 10 - SET UNIT CHARACTERISTICS   |
|       |    |    |    | 0BF9 | 1455 |        | DET_ACC_PATH>                  | ; 11 - DETERMINE ACCESS PATH      |
| FD20  |    |    | 31 | OC06 | 1456 | BRW    | ERR_T                          |                                   |

```
0C09 1458 .SBTTL - 8 - AVAILABLE
0C09 1459 :*****
0C09 1460 :
0C09 1461 : # 8 - A V A I L A B L E
0C09 1462 :
0C09 1463 :*****
0C09 1464 :
51 50 B5 0C09 1465 AVAIL: TSTW R0 ; Check unit status
1F 12 0C0B 1466 BNEQ 30$ ; Either offline or available
F8 A5 3C 0C0D 1467 10$: MOVZWL CDRPSW_SLOT(R5),R1 ; Pick up unit number
10 88 0C11 1468 BISB #UQBSM_SEQ,- ; Set for sequential mode
2B A4 0C13 1469 UQBSB_FLAGS(R4) ; no new IO until done
E8 A5 0E 0C15 1470 INSQUE CDRPSL_UQB_FL(R5),- ; Insert this CDRP on unit queue
64 B4 0C18 1471 @UQBS[ CDRP BL(R5)
50 A4 B6 0C1A 1472 INCW UQBSW_NOM_QUE(R4) ; Account for it
1C 10 0C1D 1473 BSBB MAKE_AVAIL ; Make available
54 78 A5 D0 0C1F 1474 MOVL CDRPSL_MST_UQB(R5),R4 ; Recover UQB pointer
10 8A 0C23 1475 BICB #UQBSM_SEQ,- ; Clear the sequential flag -
2B A4 0C25 1476 UQBSB_FLAGS(R4) ; allow any IO now
50 D4 0C27 1477 CLRL R0 ; Send success
FEA8 31 0C29 1478 20$: BRW SEND_PKT ; Do it
0C2C 1479 :
0C2C 1480 ; If unit was OFFLINE, command is illegal in this state, send OFFLINE status
0C2C 1481 :
0C2C 1482 :
FA 50 E8 0C2C 1483 30$: ASSUME <MSCP$K_ST_OFFLN&1> NE 0 ; OFFLINE
54 D5 0C2F 1484 BLBS R0,20$ ; Any UQB?
05 13 0C31 1485 TSTL R4 ; No, skip next
09 E0 0C33 1486 BEQL 40$ ; If this is a shadow master,
D5 12 A4 0C35 1487 BBS #MSCP$V_UF_SSMST,- ; then allow the command else ...
FE97 31 0C38 1488 40$: BRW SEND_SUCCESS ; AVAILABLE, is NOP
0C3B 1489 :
0C3B 1490 ; Subroutine to make unit AVAILABLE. It can be called recursively.
0C3B 1491 :
0C3B 1492 : R1 = Unit number
0C3B 1493 : R3 --> HQB
0C3B 1494 : R4 --> UQB
0C3B 1495 : SP --> Caller of MAKE_AVAIL
0C3B 1496 : 4(SP)> Caller's caller
0C3B 1497 :
0C3B 1498 MAKE_AVAIL:
00 10 A3 51 E5 0C3B 1499 BBCC R1,HQBSL_UNIT_ONLN(R3),10$ ; Show this unit not online this host
51 1E A3 9A 0C40 1500 10$: MOVZBL HQBSB_VEC(R3),R1 ; Pick up host slot number
54 D5 0C44 1501 TSTL R4 ; Any UQB?
2A 13 0C46 1502 BEQL 30$ ; No, skip next
20 34 A4 51 E5 0C48 1503 BBCC R1,UQBSL_HOST_ONLN(R4),20$ ; Show this host not online this unit
OC A4 97 0C4D 1504 DECB UQBSB_HOST_CNT(R4) ; Drop host count on unit
20 18 0C50 1505 BGEQ 30$ ; If any host left, leave now
09 E0 0C52 1506 BBS #MSCP$V_UF_SSMST,- ; If this is a shadow master,
12 A4 0C54 1507 UQBSW_ONT_FLGS(R4),- ; then allow the command else ...
66 0C56 1508 AV CASE3
04 B0 0C57 1509 MOVW #MSCP$K_ST_AVLBL,- ; else set status for unit
OE A4 0C59 1510 MOVL UQBSW_STATUS(R4) ; ONLINE --> AVAILABLE
52 30 A4 D0 0C5B 1511 MOVL UQBSL_UCB(R4),R2 ; Pick up UCB pointer
00AE C2 97 0C5F 1512 DECB UCB$B_ONLCNT(R2) ; Decrement count
OE 15 0C63 1513 BLEQ 40$ ; If zero, do IO
0C65 1514 ;
```

```
0C65 1515 ; This is the last host online to this unit, but there is a local path
0C65 1516 ; to the unit. Thus there is no actual IO needed to the unit. However,
0C65 1517 ; check if any post processing is called for (shadow member only).
0C65 1518 ;
08 12 OE E1 0C65 1519 BBC #MSCP$V UF_SSMEM,- ; If this is a shadow member,
0098 A4 31 0C67 1520 UQBSW_ONT_FLGS(R4),30$ ; then
0C6A 1521 BRW AV_CASE2X ; Continue in special routine
0C6D 1522 ;
0C6D 1523 ; Host was never online to this unit. However make special check for
0C6D 1524 ; a shadow master with no units and in a transistion state.
0C6D 1525 ;
09 E0 0C6D 1526 20$: BBS #MSCP$V UF_SSMST,- ; If this is a shadow master,
12 A4 0C6F 1527 UQBSW_ONT_FLGS(R4),- ; then allow the command else ...
4B 0C71 1528 AV_CASE3
05 0C72 1529 30$: RSB ; Leave without further processing
0C73 1530 ;
0C73 1531 ; This is the last host online to this unit, and there are no local paths
0C73 1532 ; to it either. Thus decide if SPINDOWN is needed, and issue the AVAIL.
0C73 1533 ;
55 D5 0C73 1534 40$: TSTL R5 ; Is there a CDRP?
FB 13 0C75 1535 BEQL 30$ ; No, skip tests
51 B0 A5 D0 0C77 1536 MOVL CDRP$L_MSGBUF(R5),R1 ; Recover MSCP packet pointer
0D 13 0C7B 1537 BEQL 50$ ; None
01 E1 0C7D 1538 BBC #MSCP$V MD_SPNDW,- ; If SPINDOWN modifier...
08 0A A1 0C7F 1539 MSCP$W_MODIFIER(R1),50$
B0 0C82 1540 MOVW #IOS_UNLOAD!- ; Set function to UNLOAD
0C83 1541 IOSM_INHERLOG,-
0C83 1542 CDRP$W_FUNC(R5)
CO A5 0801 8F 0C88 1543 BRB 60$
06 11 0C8A 1544
B0 0C8A 1545 50$: MOVW #IOS_AVAILABLE!- ; Set function to AVAILABLE
0C8B 1546 IOSM_INHERLOG,-
CO A5 0811 8F 0C8B 1547 CDRP$W_FUNC(R5)
OE E1 0C90 1548 60$: BBC #MSCP$V UF_SSMEM,- ; If this is a shadow member,
08 12 A4 0C92 1549 UQBSW_ONT_FLGS(R4),70$ ; then
CC A5 8ED0 0C95 1550 POPL CDRP$L_SVAPTE(R5) ; Save normal return address
OCFE'CF 9F 0C99 1551 PUSHAB AV_CASE2 ; and substitute special routine
0C9D 1552 70$:: BSBB SETX ; Do the IO
0C9D 1553 ;
0C9D 1554 ; Subroutine to setup for disk IO
0C9D 1555 ;
D6 A5 02 B0 0C9D 1556 SETX: MOVW #2,CDRP$W_STATE(R5) ; Show it queued to driver
54 A5 53 D0 OCA1 1557 MOVL R3,CDRP$L_FR3X(R5) ; Save the HQB pointer
C4 A5 54 D0 OCA5 1558 MOVL R4,CDRP$L_UQB(R5) ; Save the UQB pointer
30 A4 D0 OCA9 1559 MOVL UQBSL_UCBTR4) - ; Transfer UCB address
BC A5 0CAC 1560 CDRP$L_UCB(R5) ; to UCB addr in IRP
48 A5 D4 OCAE 1561 CLRL CDRP$L_CBN(R5) ; Just in case
53 A0 A5 DE OCB1 1562 MOVAL CDRP$L_IOQFL(R5),R3 ; Cover IRP w/R3
08 E2 OCB5 1563 BBSS #IRP$V_PHYSIO,- ; Set PHYSICAL IO bypass
00 CA A5 0CB7 1564 CDRP$W_STS(R5),10$
0DB7 31 0CBA 1565 10$: BRW DO_DISKX ; Send it off
0CBD 1566 ;
0CBD 1567 ; AVAILABLE to a shadow set master: Send AVAILABLE to each of the physical
0CBD 1568 ; units in the shadow set. If the shadow set is dissolved, then get rid of
0CBD 1569 ; the shadow master unit.
0CBD 1570 ;
0CBD 1571 AV_CASE3:
```

```
26 A4 33 OCBD 1572 CWTWB LQBSW_SHDW_STS(R4) -  
C2 A5 13 OCC0 1573 CDRPSB_SHDW_CNT(R5) ; If no members  
25 13 OCC2 1574 BEQL 30$ ; then just deallocate shadow UQB  
5C A5 8ED0 OCC4 1575 POPL CDRPSL_SAVD_RTX(R5) ; Save return address  
01 8A OCC8 1576 BICB #1@CDRPSV_SAVE - ; Clear out unneeded flag  
FA A5 OCCA 1577 CDRPSB_FLAGS(R5)  
54 64 D0 OCCC 1578 10$: MOVL UQBSL_SHDW_FL(R4),R4 ; Pick up next shadow member  
OC 13 OCCF 1579 BEQL 20$ ; None left  
51 2A A4 9A OCD1 1580 MOVZBL UQBSB_SLOT(R4),R1 ; Pick up unit slot  
FF63 30 OCD5 1581 BSBW MAKE_AVAIL ; Do the available  
C2 A5 97 OCD8 1582 DECB CDRPSB_SHDW_CNT(R5) ; Drop the count  
EF 1A OCDB 1583 BGTRU 10$ ; Loop if not done  
5C A5 D0 OCDD 1584 20$: PUSHL CDRPSL_SAVD_RTX(R5) ; Return address  
54 78 A5 D0 OCE0 1585 MOVL CDRPSL_MST_QQB(R5),R4 ; Pick up shadow master UQB  
26 A4 B5 OCE4 1586 TSTW UQBSW_SHDW_STS(R4) ; Is this the last physical unit?  
14 12 OCE7 1587 BNEQ 40$ ; No, just return  
51 2A A4 9A OCE9 1588 30$: MOVZBL UQBSB_SLOT(R4),R1 ; Pick up unit slot  
F39E CF41 D4 OCED 1589 CLRL UNIT_VEC[R1] ; Clear shadow unit out of vector  
50 0080 C4 0F OCF2 1590 REMQUE UQBSB_UNITQ(R4),R0 ; Unlink from chain  
50 54 D0 OCF7 1591 MOVL R4,R0 ; Set R0 to point to head  
F951 31 OCFA 1592 BRW ; Get rid of shadow master UQB  
OCFD 1593 ; control returns to MAKE_AVAIL caller  
05 OCFD 1594 40$: RSB ; Return to caller's caller  
OCFE 1595  
OCFE 1596 ; AVAILABLE to a shadow set member: Clear out the shadowing fields, remove  
OCFE 1597 ; this unit from the shadow set ring, and if last unit, set shadow set  
OCFE 1598 ; master unit to available.  
OCFE 1599  
OCFE 1600 AV_CASE2:  
53 54 A5 7D OCFE 1601 MOVQ CDRPSL_FR3X(R5),R3 ; Recover HQB & UQB  
CC A5 DD OD02 1602 PUSHL CDRPSL_SVAPTE(R5) ; Return address  
OD05 1603  
OD05 1604 AV_CASE2X:  
00 12 A4 OE E5 OD05 1605 BBCC #MSCPSV_UF_SSMEM, - ; Clear the shadow set member flag  
24 A4 B4 OD07 1606 UQBSW_CNT_FLGS(R4),10$  
50 78 A4 D0 OD0A 1607 10$: CLRW UQBSW_SHDW_UNT(R4) ; Clear out old shadow master  
78 A4 D4 OD0D 1608 MOVL UQBSL_SHDW_MST(R4),R0 ; Point to shadow master  
26 A0 B7 OD11 1609 CLRL UQBSL_SHDW_MST(R4) ; Remove the pointer  
0B 12 OD14 1610 DECW UQBSW_SHDW_STS(R0) ; Decre shadow member count  
60 D4 OD17 1611 BNEQ 20$ ; At least one left  
78 A0 D4 OD19 1612 CLRL UQBSL_SHDW_FL(R0) ; Clear master pointer  
04 B0 OD1B 1613 CLRL UQBSL_SHDW_LOW(R0) ; and low pointer  
0E A0 OD1E 1614 MOVW #MSCPSK_ST_AVLBL, - ; Set status to AVAIL  
22 11 OD20 1615 UQBSW_STATUS(R0)  
OD22 1616 BRB 40$ ; Skip all else  
OD24 1617  
52 64 D0 OD24 1618 20$: MOVL UQBSL_SHDW_FL(R4),R2 ; Get our successor  
51 64 OF OD27 1619 REMQUE UQBSL_SHDW_FL(R4),R1 ; Remove us from the shadow ring  
54 78 A0 D1 OD2A 1620 CMPL UQBSL_SHDW_LOW(R0),R4 ; Were we the lowest unit?  
0E 12 OD2E 1621 BNEQ 30$ ; No, leave  
28 A2 B0 OD30 1622 MOVW UQBSW_UNIT(R2), - ; Transfer its unit  
24 A0 OD33 1623 UQBSB_SHDW_UNT(R0) ; to shadow master's unit field  
30 A2 D0 OD35 1624 MOVL UQBSL_OCB(R2), - ; Transfer its UCB pointer  
30 A0 OD38 1625 UQBSB_UCB(R0) ; to shadow master  
78 A0 52 D0 OD3A 1626 MOVL R2,UQBSL_SHDW_LOW(R0) ; Update the new lowest pointer  
54 60 D1 OD3E 1627 30$: CMPL UQBSL_SHDW_FL(R0),R4 ; Were we the first in the ring?  
03 12 OD41 1628 BNEQ 40$ ; No, skip on
```

```
60 52 D0 OD43 1629      MOVL    R2,UQB$$_SHDW_FL(R0)    ; Yes, update master with new link
      05 OD46 1630 40$: RSB                          ; Return to MAKE_AVAIL caller
      OD47 1631 :
      OD47 1632 : AVAIL_ALL: Make the unit AVAILABLE to all hosts to which it is currently
      OD47 1633 : ONLINE. Also make it physically AVAILABLE. Cannot be called for
      OD47 1634 : shadow masters.
      OD47 1635 :
      OD47 1636 : R4 --> UQB of unit to be made AVAIL
      OD47 1637 :
      OD47 1638 AVAIL_ALL:
      OD47 1639      POPL    CDRP$$_SAVD_RTX(R5)      ; Save return address
      C4 A5 54 D0 OD48 1640      MOVL    R4,CDRP$$_UQB(R5)      ; Save UQB pointer
      2A A4 98 OD4F 1641      MOVZBW  UQB$$_SLOT(R4),-      ; Pick up the slot index
      FB A5 OD52 1642      CDRP$$_SLOT(R5)      ; and save in CDRP
      01 8A OD54 1643      BICB    #1@CDRP$$_SLAVE,-      ; Clear unneeded flag
      FA A5 OD56 1644      CDRP$$_F[AGS(R5)
      1F B0 OD58 1645      MOVW    #<2*MSUSK_MAX_HOST>-1,- ; Stuff max index into convenient
      D0 A5 OD5A 1646      CDRP$$_BOFF(R5)      ; place
      53 D0 A5 3C OD5C 1647 10$: MOVZWL CDRP$$_BOFF(R5),R3 ; Pick up index
      53 F3AB CF43 D0 OD60 1648      MOVL    HOST_VEC[R3],R3 ; Pick up possible HOB address
      07 13 OD66 1649      BEQL    20$          ; Non, loop for more
      51 FB A5 3C OD68 1650      MOVZWL  CDRP$$_SLOT(R5),R1 ; Pick up the slot index
      FECC 30 OD6C 1651      BSBW    MAKE_AVAIL      ; AVAILABLE the unit
      D0 A5 B7 OD6F 1652 20$: DECW    CDRP$$_BOFF(R5)      ; Count down the index
      E8 18 OD72 1653      BGEQ    10$          ; Loop for more hosts
      5C B5 17 OD74 1654      JMP     @CDRP$$_SAVD_RTX(R5) ; Leave
```

```
OD77 1656 .SBTTL - 9 - ONLINE
OD77 1657 :*****
OD77 1658 :
OD77 1659 : # 9 - O N L I N E
OD77 1660 :
OD77 1661 :*****
OD77 1662 :
OD77 1663 ONLINE:
40 50 E8 OD77 1664 BLBS R0,60$ ; If OFFLINE, error out
E3B8 8F AA OD7A 1665 BICW #^C<HOSTSET>,- ; Clear non-host flags in the packet
OE A1 OD7E 1666 MSCPSW_UNT_FLGS(R1)
13 0A A1 E0 OD80 1667 BBS #MSCPSV_MD_SHDSP,-
OD82 1668 MSCPSW_MODIFIER(R1),10$; Is this a shadowing operation?
OD85 1669 :
OD85 1670 : CASE 1: ONLINE to a single physical unit, no shadowing at all
OD85 1671 :
OD85 1672 :
OD85 1673 :***
OD85 1674 : TEMP UNTIL VMS GETS SHADOWING IN MOUNT
OD85 1675 :
13 12 A4 09 E0 OD85 1676 BBS #MSCPSV_UF_SSMST,UQBSW_UNT_FLGS(R4),20$
OD8A 1677 :***
OD8A 1678 :
B3 OD8A 1679 BITW #<MSCPSM_UF_SSMST!- ; No, check consistency
OD8B 1680 MSCPSM_UF_SSMEM>,-
OD8B 1681 UQBSW_UNT_FLGS(R4) ; Is the unit involved in shadowing?
12 A4 4200 8F OD90 1682 BEQL CHECK ONLINE ; No, go directly to ONLINE test
50 0A 18 9C OD92 1683 ROTL #24,#MSCPSW_MODIFIER,R0 ; Yes, set illegal modifier
15 11 OD96 1684 BRB 40$ ; ... send invalid end message
OD98 1685 :
20 12 A4 09 E1 OD98 1686 10$: BBC #MSCPSV_UF_SSMST,- ; Is this unit a shadow master?
OD9A 1687 UQBSW_UNT_FLGS(R4),70$
OD9D 1688 :
OD9D 1689 : CASE 3: ONLINE to a shadow master unit
OD9D 1690 :
OD9D 1691 :
OD9D 1692 :***
OD9D 1693 : TEMP
OD9D 1694 :
20 A1 04 A1 B0 OD9D 1695 20$: MOVW MSCPSW_UNIT(R1),MSCPSW_SHDW_UNT(R1)
ODA2 1696 :***
ODA2 1697 :
20 A1 B1 ODA2 1698 CMPW MSCPSW_SHDW_UNT(R1),- ; Yes, now does the virtual
28 A4 ODA5 1699 UQBSW_UNIT(R4) ; unit match the physical?
07 13 ODA7 1700 BEQL 50$ ; Yes
50 20 18 9C ODA9 1701 30$: ROTL #24,#MSCPSW_SHDW_UNT,R0 ; No, set illegal shadow unit
FB87 31 ODA0 1702 40$: BRW ERR_X
ODB0 1703 :
26 A4 B0 ODB0 1704 50$: MOVW UQBSW_SHDW_STS(R4),- ; Pick up the number of members
22 A1 ODB3 1705 MSCPSW_SHDW_STS(R1)
3E 12 ODB5 1706 BNEQ CHECK ONLINE ; Non-zero, continue processing
50 24 B0 ODB7 1707 MOVW #<1*MSCPSK_ST_SBCOD>!--
FD17 31 ODBA 1708 MSCPSK_ST_AV[BL,R0 ; AVAILABLE, NO MEMBERS
ODBD 1709 60$: BRW SEND_PKT
ODBD 1710 :
ODBD 1711 : CASE 2: ONLINE to a physical unit to bring it into a shadow set
ODBD 1712 :
```

```
50 20 A1 B0 ODBD 1713 70$: MOVW MSCPSW_SHDW_UNT(R1),R0 ; Pick up shadow unit
      E6 B1 ODC1 1714      BGEQ 30$ ; Error if sign bit not set
28 A4 50 B1 ODC3 1715      CMPW R0,UQBSW_UNIT(R4) ; Does virtual unit match the physical?
      E0 13 ODC7 1716      BEQL 30$ ; Yes, error (only true for master)
      OE E1 ODC9 1717      BBC #MSCPSV UF_SSMEM,- ; No, now is this an existing shadow
      19 12 A4 ODCB 1718      UQBSW_UNT_FLGS(R4),100$ ; set member
24 A4 50 B1 ODCE 1719      CMPW R0,UQBSW_SHDW_UNT(R4) ; Yes, does its shadow master match?
      D5 12 ODD2 1720      BNEQ 30$ ; No, error
      14 A1 C1 ODD4 1721 80$: ADDL3 MSCPSL_EXCL_LBA(R1),- ; Now pick up the excluded area
52 18 A1 ODD7 1722      MSCPSL_EXCL_LBC(R1),R2
48 A4 52 D1 ODDA 1723      CMPL R2,UQBSL_MAXBLOCK(R4) ; Is it beyond the legal limits
      15 1B ODDE 1724      BLEQU CHECK_ONLINE ; No, all is okay
50 14 18 9C ODE0 1725      ROTL #24,#MSCPSL_EXCL_LBA,R0 ; Yes, set status
      FB50 31 ODE4 1726 90$: BRW ERR_X ; ... send invalid end message
      ODE7 1727 ;
      ODE7 1728 ; Physical unit is not already a shadow set member. Now test if it is
      ODE7 1729 ; already ONLINE as a non-shadow unit to anyone. If so, error out.
      ODE7 1730 ;
      OC A4 95 ODE7 1731 100$: TSTB UQBSB_HOST_CNT(R4) ; Is it already online?
      E8 19 ODEA 1732      BLSS 80$ ; No, allow it thru
50 FF000000 8F D0 ODEC 1733      MOVL #*XFF000000,R0 ; Yes, this is illegal
      EF 11 ODF3 1734      BRB 90$ ; Tell host
      ODF5 1735 ;
      ODF5 1736 ; Now check status of unit.
      ODF5 1737 ;
      ODF5 1738 CHECK_ONLINE:
      50 B5 ODF5 1739      TSTW R0 ; Check status
      24 12 ODF7 1740      BNEQ 30$ ; AVAIL
      ODF9 1741 ;
      ODF9 1742 ; Was ONLINE to this host already
      ODF9 1743 ;
50 0100 8F A8 ODF9 1744      BISW2 #<MSCPSK SC_ALONL*- ; Set subcode 'Already online'
      1314'CF 9F ODFE 1745      MSCPSK_ST_SBCOD>,R0
      52 DD OE02 1746      PUSHAB COPY_CHAR ; Where to go next
      E280 8F AB OE04 1747 10$: PUSHL R2 ; Save register
      52 12 A4 OE08 1748      BICW3 #NONHOSTSET,- ; Pick up host settable flags from
      52 OE A1 B1 OE0B 1749      UQBSW_UNT_FLGS(R4),R2 ; the unit
      04 BA OE0F 1750      CMPW MSCPSW_UNT_FLGS(R1),R2 ; Check againsts host settable flags
      09 13 OE11 1751      POPR #*M<R2$ ; Restore, but leave CC set
      8E D5 OE13 1752      BEQL 20$ ; All okay, fill in info
      50 OE 18 9C OE15 1753      TSTL (SP)+ ; Remove return address
      FB1B 31 OE19 1754      ROTL #24,#MSCPSW_UNT_FLGS,R0 ; Set secondary status
      OE1C 1755      BRW ERR_X ; ... send invalid end packed
      05 OE1C 1756
      OE1D 1757 20$: RSB
      OE1D 1758 ;
      OE1D 1759 ; Was AVAILABLE, now turn ONLINE
      OE1D 1760 ;
      OE A4 B5 OE1D 1761 30$: TSTW UQBSW_STATUS(R4) ; Check the actual unit
      02 12 OE20 1762      BNEQ 35$ ; No other host has it online, skip
      DE 10 OE22 1763      BSBB 10$ ; Test unit flags
      09 E0 OE24 1764 35$: BBS #MSCPSV UF_SSMST,- ; If shadow master unit,
      6F 12 A4 OE26 1765      UQBSW_UNT_FLGS(R4),50$ ; skip the physical online stuff
      52 1E A3 9A OE29 1766      MOVZBL HQBSB_VEC(R3),R2 ; Pick up host slot number
14 34 A4 52 E0 OE2D 1767      BBS R2,UQBSL_HOST_ONLN(R4),40$ ; If host online bit set, MOUNTVER
      OC A4 95 OE32 1768      TSTB UQBSB_HOST_CNT(R4) ; Test count of online hosts
      61 18 OE35 1769      BGEQ 50$ ; Skip UCB & packack, if already done
```

```

52 30 A4 D0 OE37 1770      MOVL   UQB$U_UCB(R4),R2      ; Pick up UCB address
01 00AE C2 96 OE3B 1771      INCB   UCB$B_ONLCNT(R2)      ; Increment the online count
    00AE C2 91 OE3F 1772      CMPB   UCB$B_ONLCNT(R2),#1    ; Is it already ONLINE?
    52      1A OE44 1773      BGTRU  50$                ; Yes, do not issue another PACK_ACK
    B0      B0 OE46 1774 40$: MOVW   #IO$_PACKACK!-        ; Set function to PACK_ACK
    OE47 1775      IO$M_INHERLOG,-
    OE47 1776      CDRP$W_FUNC(R5)
CO A5 0808 8F      BISB   #UQB$M_SEQ,-                ; Set the sequential flag
    10      88 OE4C 1777      UQB$B_FLAGS(R4)
    2B A4      OE4E 1778      IN$QUE CDRP$U_UCB_FL(R5),-    ; Insert on unit queue
    E8 A5      OE50 1779      @UQB$C_CDRP_BL(R4)
    64 B4      OE53 1780      INCW   UQB$W_NOM_QUE(R4)      ; Account for it
    50 A4      B6 OE55 1781      BSBW  SETX              ; Do the IO
    FE42      30 OE58 1782      MOVQ  CDRP$U_FR3X(R5),R3    ; Restore HQB & UQB address
53 54 A5      7D OE5B 1783      BICB   #UQB$M_SEQ,-        ; Clear the sequential flag
    10      8A OE5F 1784      UQB$B_FLAGS(R4)
51 2B A4      OE61 1785      MOVL   CDRP$U_M$GBUF$X(R5),R1  ; Restore message buffer address
    B0 A5      D0 OE63 1786      BEQL  47$                ; Bug trap
    2B      13 OE67 1787 ; ??? bug trap
    OE67 1788      OE69 1789 ; ??? end bug trap
    OE69 1790      ASSUME  MSCP$K_ST_SUCC EQ 0
00F4 2C 50      E8 OE69 1791      BLBS  R0,50$            ; Success
    8F      50 B1 OE6C 1792      CMPW  R0,#SS$_ILLIOFUNC    ; Error, was it illegal function
    25      13 OE71 1793      BEQL  50$                ; Yes, okay because drive not support
52 30 A4 D0 OE73 1794      MOVL   UQB$U_UCB(R4),R2      ; Pick up UCB address
    00AE C2 77 OE77 1795      DECB   UCB$B_ONLCNT(R2)      ; Drop the online count
0254 8F      50 B1 OE7B 1796      CMPW  R0,#SS$_VOLINV      ; Is it volume invalid?
    0C      13 OE80 1797      BEQL  45$                ; Yes, change error to mscp error
01A4 8F      50 B1 OE82 1798      CMPW  R0,#SS$_MEDOFL      ; Is it medium offline?
    05      13 OE87 1799      BEQL  45$                ; Yes, change error to mscp error
    50      0B D0 OE89 1800      MOVL  #MSCP$K_ST_DRIVE,R0  ; Make the error fatal drive error
    03      11 OE8C 1801      BRB    46$
    OE8E 1802      50      03 D0 OE8E 1803 45$: MOVL  #MSCP$K_ST_OFFLN,R0 ; Substitute MSCP error status
    FC40      3- OE91 1804 46$: BRW   SEND_PKT            ; and leave now
    OE94 1805      OE94 1806 47$: BUG_CHECK      MSCPSERV,FATAL ; HQB invalid!!!
    OE98 1807      OE98 1808 50$: CLRL  R0                ; Set success status (ONLINE)
03 52 1E A3 9A OE9A 1809      MOVZBL HQB$B_VEC(R3),R2      ; Pick up host slot number
    34 A4      52 E2 OE9E 1810      BBSS  R2,UQB$U_HOST_ONLN(R4),60$ ; Set host online bit in unit
    0C A4      96 OEA3 1811      INCB   UQB$B_HOST_CNT(R4)    ; Bump the host online count
52  F8 A5      3C OEA6 1812 60$: MOVZWL CDRP$U_SLOT(R5),R2  ; Pick up unit slot number
00 10 A3      52 E2 OEAA 1813      BBSS  R2,HQB$U_UNIT_ONLN(R3),70$ ; Set unit-online bit in host
    0E A4      B4 OEAf 1814 70$: CLRW  UQB$W_STATUS(R4)      ; Set ONLINE status in unit
    OE82 1815      ; Physical IO to unit, if any, done. Now check if any post processing
    OE82 1816      ; needed - shadow physical units only.
    OE82 1817      ;
    OE82 1818      ;
    09      E0 OE82 1819      BBS    #MSCP$V_UF_SSMST,-      ; Leave now if shadow master unit
05 12 A4      OE84 1820      BBS    #MSCP$V_MD_SHDSP,-      ; Leave now if not shadowing
    04      E0 OE87 1821      MSCP$W_MODIFIER(R1),CASE2A
03 0A A1      OE89 1822      CCHAR: BRW  COPY_CHAR
    0455      31 OE8C 1823
```

```
0EBF 1825 .SBTTL - Shadow units
0EBF 1826 :
0EBF 1827 : CASE 2: Deal with the shadow unit
0EBF 1828 :
0EBF 1829 CASE2A:
01 0C A4 91 0EBF 1830 CMPB UQBSB_HOST_CNT(R4),#1 ; If this unit is already ONLINE
F7 1E 0EC3 1831 BGEQU CCHAR ; then leave now
38 BB 0EC5 1832 PUSHF #M<R3,R4,R5> ; Save some registers
0EC7 1833 :
0EC7 1834 : Look up the shadow unit. If it exists we will do some checks, if it
0EC7 1835 : does not. We will create a new UQB for it. If there is no space we
0EC7 1836 : will return an error code.
0EC7 1837 :
52 20 A1 3C 0EC7 1838 MOVZWL MSCPSW_SHDW_UNIT(R1),R2 ; Pick up shadow unit from command
50 52 D0 0ECB 1839 MOVL R2,R0 ; Copy
54 52 FFFFFFF0 8F CB 0ECE 1840 BICL3 #C<<1>MSUSK_UNIT_SIZ>-1>,R2,R4 ; Slot
52 F1B5 CF44 D0 0ED6 1841 MOVL UNIT_VEC[R4],R2 ; Look for unit block in slot
07 13 0EDC 1842 BEQL 10$ ; Empty
20 A1 B1 0EDE 1843 CMPW MSCPSW_SHDW_UNIT(R1),-
28 A2 0EE1 1844 UQBSW_UNIT(R2) ; Hit, is it a match?
40 13 0EE3 1845 BEQL OLD_UQB ; Yes
FA74 30 0EE5 1846 10$: BSBW FIND_SLOT ; No, try more probing
38 18 0EE8 1847 BGEQ OLD_UQB ; Found unit
0EEA 1848 :
0EEA 1849 : No match in table. Find an empty slot.
0EEA 1850 :
52 20 A1 3C 0EEA 1851 MOVZWL MSCPSW_SHDW_UNIT(R1),R2 ; Pick up shadow unit from command
53 52 FFFFFFF0 8F CB 0EEE 1852 BICL3 #C<<1>MSUSK_UNIT_SIZ>-1>,R2,R3 ; Slot
55 F195 CF43 D0 0EF6 1853 MOVL UNIT_VEC[R3],R5 ; Look for unit block in slot
14 13 0EFC 1854 BEQL 30$ ; Empty
53 10 C0 0EFE 1855 ADDL #MSUSK_MAX_UNIT,R3 ; Try for second probe
55 F18A CF43 D0 0F01 1856 MOVL UNIT_VEC[R3],R5 ; Look for unit block in slot
77 13 0F07 1857 BEQL NEW_UQB ; Empty
53 1F D0 0F09 1858 MOVL #<2>MSUSK_MAX_UNIT>-1,R3 ; Sequential search
55 F17F CF43 D0 0F0C 1859 20$: MOVL UNIT_VEC[R3],R5 ; Pick up unit
6C 13 0F12 1860 30$: BEQL NEW_UQB ; Empty
53 D7 0F14 1861 DECL R3 ; Loop thru
10 53 D1 0F16 1862 CMPL R3,#MSUSK_MAX_UNIT ; Reached end?
F1 18 0F19 1863 BGEQ 20$ ; No, try again
0F1B 1864 :
0F1B 1865 : No slots left in the unit table. Return an error
0F1B 1866 :
0F1B 1867 CASE2A_ERR:
50 00AA 8F B0 0F1B 1868 MOVW #<5>MSCPSK_ST_SBCOD>!--
0F20 1869 MSCPSK_ST_CNTL,R0 ; Insufficient resources
0F20 1870 :
0F20 1871 CASE2A_ERR_OUT:
38 BA 0F20 1872 POPR #M<R3,R4,R5> ; Restore registers
FBAF 31 0F22 1873 BRW SEND_PKT
0F25 1874 :
0F25 1875 : Existing UQB, check characteristics
0F25 1876 :
0F25 1877 OLD_UQB:
54 04 AE D0 0F25 1878 MOVL 4(SP),R4 ; Recover physical unit UQB
48 A4 D1 0F29 1879 CMPL UQBSL_MAXBLOCK(R4),- ; Check maxblock
48 A2 0F2C 1880 UQBSL_MAXBLOCK(R2)
07 12 0F2E 1881 BNEQ 10$ ; Failed
```

```
20 A4 D1 0F30 1882      CMPL  UQB$ MEDIA_ID(R4),-      ; Check media_id
20 A2      0F33 1883      UQB$C_MEDIA_ID(R2)
50 0105 07 13 0F35 1884      BEQL  20$              ; Okay, continue
      8F B0 0F37 1885 10$:  MOVW  #<8*MSCP$K_ST_SBCOD>!-- ; MEDIA FORMAT ERROR
      E2 11 0F3C 1886      BRB   CASE2A_ERR_OUT
      0F3E 1887
      0F3E 1888      ; Update fields for shadowing
      0F3E 1889
      0F3E 1890
26 A2 B6 0F3E 1891 20$:  INCW  UQB$W_SHDW_STS(R2)      ; Bump the count of units
28 A2 B0 0F41 1892      MOVW  UQB$W_UNIT(R2),-        ; Copy virtual unit number
24 A4      0F44 1893      UQB$W_SHDW_UNT(R4),-        ; to physical unit's shadow unit field
      0E E2 0F46 1894      BBSS  #MSCP$V_UF_SSMEM,-    ; Set shadow set member flag
00 12 A4      0F48 1895      UQB$W_UNT_FLGS(R4),30$
28 A4 B1 0F4B 1896 30$:  CMPW  UQB$W_UNT(R4),-        ; Is this unit number smaller than
24 A2      0F4E 1897      UQB$W_SHDW_UNT(R2),-        ; any other in the shadow chain?
      0E 1E 0F50 1898      BGEQU 40$              ; No, skip on
28 A4 B0 0F52 1899      MOVW  UQB$W_UNIT(R4),-        ; Yes, update the field in shadow
24 A2      0F55 1900      UQB$W_SHDW_UNT(R2),-        ; master unit field
30 A4 D0 0F57 1901      MOVL  UQB$U_UCB(R4),-          ; Update the field in shadow
30 A2      0F5A 1902      UQB$U_UCB(R2),-            ; master UCB field
78 A2 54 D0 0F5C 1903      MOVL  R4,UQB$U_SHDW_LOW(R2) ; Update the lowest unit pointer
78 A4 52 D0 0F60 1904 40$:  MOVL  R2,UQB$U_SHDW_MST(R4) ; Set pointer to shadow master
      0F64 1905
      0F64 1906      ; Link this physical UQB into shadowing ring
      0F64 1907
50 62 D0 0F64 1908      MOVL  UQB$U_SHDW_FL(R2),R0      ; Start with R0-->first entry
52 50 D0 0F67 1909      MOVL  R0,R2                    ; Copy
52 04 A2 D0 0F6A 1910 50$:  MOVL  UQB$U_SHDW_BL(R2),R2 ; Get next entry
52 50 D1 0F6E 1911      CMPL  R0,R2                    ; End of the line?
      07 13 0F71 1912      BEQL  60$              ; Yes
28 A4 B1 0F73 1913      CMPW  UQB$W_UNIT(R4),-        ; Compare unit numbers
28 A2      0F76 1914      UQB$W_UNIT-UQB$U_SHDW_FL(R2)
      F0 1F 0F78 1915      BLSSU 50$              ; Current is still higher than target
62 64 0E 0F7A 1916 60$:  INSQUE UQB$U_SHDW_FL(R4),(R2) ; Insert ahead of this unit
      00D9 31 0F7D 1917      BRW   CASE2A_DONE          ; and leave now
      0F80 1918
      0F80 1919      ; No existing UQB, create one
      0F80 1920
      0F80 1921      R1 --> MSCP packet
      0F80 1922      R2 = Shadow unit number
      0F80 1923      R3 = Shadow slot
      0F80 1924      R4 --> Physical unit UQB
      0F80 1925      (SP)--> HQB
      0F80 1926
      0F80 1927 NEW_UQB:
      0E BB 0F80 1928      PUSHR #^M<R1,R2,R3>        ; Save registers
51 22 A1 B4 0F82 1929      CLRW  MSCP$W_COPY_SPD(R1)    ; Ignore copy speed field
      9C 8F 9A 0F85 1930      MOVZBL #UQB$C_LEN,R1      ; Size of unit block
      F6C8 30 0F89 1931      BSBW  ALLOMEM            ; Go get it
      06 50 E8 0F8C 1932      BLBS  R0,110$          ; Okay
5E 0C C0 0F8F 1933      ADDL  #3*4,SP                ; Clear off registers
      FF86 31 0F92 1934      BRW   CASE2A_ERR          ; Error out
      0F95 1935
      55 52 D0 0F95 1936 110$:  MOVL  R2,R5            ; New UQB pointer
50 08 AE D0 0F98 1937      MOVL  8(SP),R0            ; Restore slot number
53 0C AE 7D 0F9C 1938      MOVQ  12(SP),R3           ; Pick up HQB & UQB
```

|                                  |        |                                                         |
|----------------------------------|--------|---------------------------------------------------------|
| 00 10 A3 50 E2 OFA0 1939         | BBSS   | R0, HQBSL UNIT ONLN(R3), 111\$ ; Set a 'online' bit     |
| FOE5 CF40 55 D0 OFA5 1940 111\$: | MOVL   | R5, UNIT_VEC(R0) ; Stuff it where it belongs            |
| 82 82 54 D0 OFAB 1941            | MOVL   | R4, (R2)+ ; Set link to first UQB                       |
| 82 82 82 D4 OFAE 1942            | CLRL   | (R2)+ ; Clear out unused                                |
| 82 82 51 B0 OFB0 1943            | MOVW   | R1, (R2)+ ; Set size field                              |
| 82 0A60 8F B0 OFB3 1944          | MOVW   | #<DYN\$C_SCS_UQB@8>:- ; Set type                        |
| 82 82 01 AE OFB8 1945            |        | DYN\$C_SCS, (R2)+                                       |
| 82 82 04 B0 OFB8 1946            | ASSUME | UQBSB_HOST_CNT EQ 12                                    |
| 82 82 82 B4 OFBE 1947            | MNEGW  | #1, (R2)+ ; Set initial HOST CNT, AVL                   |
| 82 82 82 B0 OFBB 1948            | MOVW   | #MSCP\$K_ST_AVLBL, (R2)+ ; Set STATUS to AVAILABLE      |
| 82 82 82 B0 OFBE 1949            | ASSUME | UQBSW_MULT_UNIT EQ UQBSB_HOST_CNT+4                     |
| 82 82 82 B0 OFC0 1950            | CLRW   | (R2)+ ; Set unit                                        |
| 82 82 82 B0 OFC1 1951            | MOVW   | #<MSCP\$M_UF_REPLC!- ; Set flags - Controller Bad Block |
| 82 82 82 B0 OFC1 1952            |        | MSCP\$M_UF_SSMST>, - ; - Shadow set master              |
| 82 82 82 B0 OFC1 1953            |        | (R2)+ ; ... in the unit block                           |
| 82 82 82 B0 OFC1 1954            | CLRL   | (R2)+ ; Clear out reserved                              |
| 82 82 82 B0 OFC1 1955            | ASSUME | UQBSQ_UNIT_ID EQ UQBSW_MULT_UNIT+8                      |
| 82 82 82 B0 OFC1 1956            | CLRW   | (R2)+ ; Clear low 32 bits of unit id                    |
| 82 82 82 B0 OFC1 1957            | MOVB   | #4, -1(R2) ; Set MSCP class code                        |
| 82 82 82 B0 OFC1 1958            | ASSUME | UQBSL_MEDIA_ID EQ UQBSQ_UNIT_ID+8                       |
| 82 82 82 B0 OFC1 1959            | MOVL   | UQBSL_MEDIA_ID(R4), (R2)+ ; Media ID                    |
| 82 82 82 B0 OFC1 1960            | ASSUME | UQBSW_SHDW_UNIT EQ UQBSL_MEDIA_ID+4                     |
| 82 82 82 B0 OFC1 1961            | MOVW   | UQBSW_UNIT(R4), (R2)+ ; Shadow unit (lowest phys unit)  |
| 82 82 82 B0 OFC1 1962            | ASSUME | UQBSW_SHDW_STS EQ UQBSW_SHDW_UNIT+2                     |
| 82 82 82 B0 OFC1 1963            | MOVW   | #1, (R2)+ ; Shadow status (number of phys units)        |
| 82 82 82 B0 OFC1 1964            | ASSUME | UQBSW_UNIT EQ UQBSW_SHDW_STS+2                          |
| 82 82 82 B0 OFC1 1965            | MOVW   | 4(SP), (R2)+ ; Set in unit number                       |
| 82 82 82 B0 OFC1 1966            | ASSUME | UQBSB_SLOT EQ UQBSW_UNIT+2                              |
| 82 82 82 B0 OFC1 1967            | MOVB   | 8(SP), (R2)+ ; Set in slot number                       |
| 82 82 82 B0 OFC1 1968            | CLRB   | (R2)+ ; Shadow count                                    |
| 82 82 82 B0 OFC1 1969            | ASSUME | UQBSL_VOL_SER EQ UQBSB_SLOT+2                           |
| 82 82 82 B0 OFC1 1970            | CLRL   | (R2)+ ; Clear volume serial                             |
| 82 82 82 B0 OFC1 1971            | ASSUME | UQBSL_UCB EQ UQBSL_VOL_SER+4                            |
| 82 82 82 B0 OFC1 1972            | MOVL   | UQBSL_UCB(R4), (R2)+ ; Set UCB pointer                  |
| 82 82 82 B0 OFC1 1973            | CLRL   | (R2)+ ; Hosts online                                    |
| 82 82 82 B0 OFC1 1974            | MOVW   | #X5306, (R2)+ ; Length                                  |
| 82 82 82 B0 OFC1 1975            | MOVL   | #A/HADO/, (R2)+                                         |
| 82 82 82 B0 OFC1 1976            | MOVL   | #A/W /, (R2)+                                           |
| 82 82 82 B0 OFC1 1977            | CLRW   | (R2)+                                                   |
| 82 82 82 B0 OFC1 1978            | CLRL   | (R2)+                                                   |
| 82 82 82 B0 OFC1 1979            | ASSUME | UQBSL_MAXBLOCK EQ UQBSL_NAME+16                         |
| 82 82 82 B0 OFC1 1980            | MOVL   | UQBSL_MAXBLOCK(R4), (R2)+ ; Maxblocks                   |
| 82 82 82 B0 OFC1 1981            | ASSUME | UQBSL_START_LBN EQ UQBSL_MAXBLOCK+4                     |
| 82 82 82 B0 OFC1 1982            | CLRL   | (R2)+ ; Start LBN                                       |
| 82 82 82 B0 OFC1 1983            | ASSUME | UQBSW_NUM_QUE EQ UQBSL_START_LBN+4                      |
| 82 82 82 B0 OFC1 1984            | CLRW   | (R2)+ ; Counts                                          |
| 82 82 82 B0 OFC1 1985            | ASSUME | UQBSL_FENCEL EQ UQBSW_NUM_QUE+8                         |
| 82 82 82 B0 OFC1 1986            | CLRW   | (R2)+ ; Fence low and high                              |
| 82 82 82 B0 OFC1 1987            | ASSUME | UQBSL_CDRP_FL EQ UQBSL_FENCEL+8                         |
| 82 82 82 B0 OFC1 1988            | MOVAL  | (R2), (R2) ; Executing CDRP que head                    |
| 82 82 82 B0 OFC1 1989            | MOVAL  | (R2)+, (R2)+                                            |
| 82 82 82 B0 OFC1 1990            | ASSUME | UQBSQ_BLOCKQ EQ UQBSL_CDRP_FL+8                         |
| 82 82 82 B0 OFC1 1991            | MOVAL  | (R2), (R2) ; Blocked CDRP que head                      |
| 82 82 82 B0 OFC1 1992            | MOVAL  | (R2)+, (R2)+                                            |
| 82 82 82 B0 OFC1 1993            | ASSUME | UQBSQ_SHQ EQ UQBSQ_BLOCKQ+8                             |
| 82 82 82 B0 OFC1 1994            | MOVAL  | (R2), (R2) ; Special holding que head                   |
| 82 82 82 B0 OFC1 1995            | MOVAL  | (R2)+, (R2)+                                            |

```

      82  54  D0  101D  1996      ASSUME  UQBSL_SHDW_LOW EQ UQBSQ_SHQ+8
      82  54  D0  101D  1997      MOVL   R4,(R2)+      ; Set lowest phys unit
      82  54  D0  1020  1998      ASSUME  UQBSW_NUM_BLK EQ UQBSL_SHDW_LOW+4
      82  54  D0  1020  1999      CLRL   (R2)+      ; Blocked counts
      82  54  D0  1022  2000      ASSUME  UQBSQ_UNITQ EQ UQBSW_NUM_BLK+4
      82  54  D0  1022  2001      CLRL   (R2)+      ; Unit que head
      82  54  D0  1024  2002      CLRL   (R2)+
24 A4  04  AE  B0  1026  2003      MOVW   4(SP),UQBSW_SHDW_UNT(R4); Set phys unit shadow unit field
78 A4  55  D0  1028  2004      MOVL   R5,UQBSL_SHDW_MST(R4); Set pointer to shadow master
      0E  E2  102F  2005      BBSS    #MSCPSV_OF_SS MEM,- ; Set shadow set member flag
      00 12 A4  1031  2006      UQBSW_UNT_FLGS(R4),120$
      1034  2007      ;
      1034  2008      ; Now Link shadow UQB into unit chain
      1034  2009      ;
51 F048 CF  DE  1034  2010 120$: MOVAL  UQB_LIST,R1      ; UQB list head
51 50 51 D0  1039  2011      MOVL   R1,R0      ; Copy
51 04 A1 D0  103C  2012 130$: MOVL   4(R1),R1      ; Get next entry, backwards
51 51 50 D1  1040  2013      CMPL   R0,R1      ; End of the line?
      07 13 1043  2014      BEQL   140$      ; Yes
A8 A1 04 AE B1  1045  2015      CMPW   4(SP),UQBSW_UNIT- -
      F0 1F 104A  2016      UQBSQ_UNITQ(R1)      ; Compare unit numbers
61 0080 C5 0E 104A  2017      BLSSU  130$      ; Current is still higher than target
      0E 104C  2018 140$: INSQUE  UQBSQ_UNITQ(R5),(R1) ; Insert ahead of this unit
      1051  2019      ;
      1051  2020      ; Send AVAIL attention message announcing the new shadow unit
      1051  2021      ;
52 55 D0  1051  2022      MOVL   R5,R2      ; Transfer UQB
      F4B1 30 1054  2023      BSBW   SEND_ATTN      ; Send the attn message
      0E BA 1057  2024      POPR    #*M<R1,R2,R3>      ; Restore registers
      1059  2025      ;
      1059  2026 CASE2A_DONE:
      38 BA 1059  2027      POPR    #*M<R3,R4,R5>
22 A1 B5 105B  2028      TSTW   MSCPSW_COPY_SPD(R1) ; Is this a shadow copy
      05 12 105E  2029      BNEQ   ONL_COPY      ; Yes, go do it
      50 D4 1060  2030 CCX: CLRL   R0      ; Set success status
      02AF 31 1062  2031      BRW    COPY_CHAR      ; Transfer the ONLINE information
```

```
1065 2033 .SBTTL - ONLINE shadow copy
1065 2034 :
1065 2035 : R1 --> MSCP packet
1065 2036 : R4 --> UQB of physical unit
1065 2037 : R5 --> CDRP for ONL command
1065 2038 :
1065 2039 :
00000024 1065 2040 WRT_UQB = ^X24
00000028 1065 2041 RED_UQB = ^X28
0000002C 1065 2042 MST_UQB = ^X2C
1065 2043 :
1065 2044 ONL_COPY:
1065 2045 BICB #1@CDRPSV NOWSTBUF,- ; Clear special flag
1067 2046 CDRPSB_FLAGS(R5)
1069 2047 TSTB UQBSB_HOST_CNT(R4) ; Is it already online and copied?
106C 2048 BGTR CCX ; Yes, leave with success
106E 2049 BISB #UQBSM_SEQ,- ; Set seq mode on write (physical)
1070 2050 UQBSB_FLAGS(R4) ; unit - lock IO until this is done
1072 2051 MOVL R1,R2 ; Transfer MSCP packet pointer
1075 2052 MOVL R4,WRT_UQB(R2) ; Store the write UQB
1079 2053 REMQUE CDRPSL_UQB_FL(R5),R0 ; Pull CDRP off physical unit que
107D 2054 DECB UQBSW_NUM_QUE(R4) ; and show it
1080 2055 MOVL R5,UQBSL_COPY_CDRP(R4) ; Save a pointer to this cdrp
1085 2056 MOVL UQBSL_SHADOW_MST(R4),R4 ; Get pointer to shadow master
1089 2057 MOVL R4,MST_UQB(R2) ; Store the shadow master UQB
108D 2058 :
108D 2059 : Handle the excluded block area
108D 2060 :
108D 2061 MOVL MSCPSW_EXCL_LBC(R2),R1 ; Pick up count
1091 2062 MOVL MSCPSL_EXCL_LBA(R2),R0 ; Pick up exclude address
1095 2063 BNEQ 10$ ; Non-zero
1097 2064 MOVL R1,MSCPSL_LBN(R2) ; Set as starting address
109B 2065 BSBW GET_MAX_BLOCK ; Find the maximum block
109E 2066 BRB 20$ ; Go set the byte count
10A0 2067 :
10A0 2068 10$: CLRL MSCPSL_LBN(R2) ; Set to start at beginning
10A3 2069 ADDL R0,R1 ; Find end of exclude area
10A6 2070 CMPL R1,UQBSL_MAXBLOCK(R4) ; Is it the end of the disk?
10AA 2071 BGEQU 20$ ; Yes
10AC 2072 BISB #UQBSM_2PASS,- ; No, set a flag for later
10AE 2073 UQBSB_FLAGS(R4) ; Convert to number of bytes
10B0 2074 20$: ASHL #9,R0,MSCPSL_BYTE_CNT(R2)
10B5 2075 :
10B5 2076 : Now check that there are no IO's in progress on the shadow unit.
10B5 2077 : If there are we must block the copy CDRP, set sequential mode (rundown)
10B5 2078 : and wait for the last IO to wake the CDRP.
10B5 2079 :
10B5 2080 30$: TSTW UQBSW_NUM_QUE(R4) ; Check the shadow master for IO
10B8 2081 BEQL 40$ ; None, all okay
10BA 2082 BISB #UQBSM_SEQ,- ; Set sequential mode while waiting
10BC 2083 UQBSB_FLAGS(R4)
10BE 2084 BLOCKH ; IO in progress, rundown and wait
10C1 2085 BRB 30$
10C3 2086 :
10C3 2087 40$: BISB #UQBSM_CIP,- ; Set the copy in progress flag
10C5 2088 UQBSB_FLAGS(R4)
10C7 2089 MOVL WRT_UQB(R2),R1 ; Write UQB pointer
```

```
01 88 10CB 2090      BISB  #UQBSM_CIP,-      ; Set the copy in progress flag
2B A1 10CD 2091      UQBSB_FLAGS(R1)
      10CF 2092      ;
      10CF 2093      ; Start of read/write loop
      10CF 2094      ;
      10CF 2095      ONL_LOOP:
      10CF 2096      BBS  #CDRPSV_ABORT,-      ; Leave if aborted
      10CF 2097      CDRPSB_FLAGS(R5),90$
      10CF 2098      BICB  #UQBSM_SEQ,-      ; Turn off sequential mode
      10D1 2099      UQBSB_FLAGS(R4)
      10D3 2100      MOVW  #IOS_READPBLK,R0      ; Set up for read
      10D6 2101      BSBW  SETUP      ; Common set up
      10D9 2102      MOVL  CDRPSL_MSGBUF(R5),R2      ; Refresh MSCP packet pointer
      10DD 2103      MOVL  CDRPSL_UQB(R5),-      ; Pick up read UQB
      10E0 2104      RED_UQB(R2)      ; and refresh the copy
      10E2 2105      ;
      10E2 2106      ONL_READ:
      10E2 2107      MOVW  #IOS_READPBLK,-      ; Set up for read
      10E4 2108      CDRPSW_FUNC(R5)
      10E6 2109      MOVL  CDRPSL_MSGBUF(R5),R2      ; Refresh MSCP packet pointer
      10EA 2110      MOVL  RED_UQB(R2),R4      ; Pick up read UQB
      10EE 2111      MOVL  R4,CDRPSL_UQB(R5)      ; Stuff it in CDRP
      10F2 2112      MOVL  UQBSL_UCB(R4),-      ; Refresh the read UCB
      10F5 2113      CDRPSL_UCB(R5)
      10F7 2114      MOVL  MST_UQB(R2),R4      ; Pick up shadow master UQB
      10FB 2115      MOVL  CDRPSL_LBN(R5),-      ; Stuff the starting fence lbn
      10FE 2116      UQBSL_FENCEL(R4)
      1100 2117      ASHL  #-9,CDRPSL_BCNT(R5),R0      ; Pick up the byte count
      1106 2118      ADDL3 R0,UQBSL_FENCEL(R4),-      ; Calculate the high fence
      110A 2119      UQBSL_FENCEH(R4)
      110C 2120      MOVW  #2,CDRPSW_STATE(R5)      ; Set the state
      1110 2121      BSBW  DO_DISK      ; Set up for disk transfer
      1113 2122      BBS  #CDRPSV_ABORT,-      ; Leave if aborted
      1115 2123      CDRPSB_FLAGS(R5),-
      1117 2124      ONL_LOOP_CHK
      1118 2125      CLRW  (CDRPSW_FUNC(R5)      ; Allow modifier in error routine
      111B 2126      BLBS  R0,ONL_WRITE      ; Success
      111E 2127      BRW  ONL_READ_ERR      ; Attempt to recover the error
      1121 2128      ;
      1121 2129      ; Read has finished, now switch some parameters for writes
      1121 2130      ;
      1121 2131      ONL_WRITE:
      1121 2132      BISW  #IOS_WRITEPBLK,-      ; Set up for write
      1123 2133      CDRPSW_FUNC(R5)
      1125 2134      MOVL  CDRPSL_MSGBUF(R5),R2      ; Grab the MSCP packet pointer
      1129 2135      MOVL  WRT_UQB(R2),R4      ; Now get the write UQB
      112D 2136      MOVL  R4,CDRPSL_UQB(R5)      ; Stuff it in CDRP
      1131 2137      MOVL  UQBSL_UCB(R4),-      ; Transfer the write UCB
      1134 2138      CDRPSL_UCB(R5)
      1136 2139      MOVW  #4,CDRPSW_STATE(R5)      ; Bump the state
      113A 2140      BSBW  DO_DISK      ; Do the disk transfer
      113D 2141      BLBS  R0,ONL_LOOP_CHK      ; Success
      1140 2142      BRW  ONL_WRITE_ERR      ; Error out
      1143 2143      ;
      1143 2144      ; Write has finished, now check if need to loop again or possibly
      1143 2145      ; wait for a pending write.
      1143 2146      ;
```

```
1143 2147 ONL_LOOP CHK:
52 B0 A5 D0 1143 2148 MOVL CDRP$L_MSGBUF(R5),R2 ; Grab pointer to MSCP packet
54 2C A2 D0 1147 2149 MOVL MST_UQB(R2),R4 ; Pick up master UQB
E0 A5 D2 A5 C0 1148 2150 ADDL CDRP$L_BCNT(R5),- ; Calc accumulated bytes xfer
1150 2151 CDRP$L_ABCNT(R5)
03 FA A5 E1 1150 2152 BBC #CDRP$V_ABORT,- ; Leave if aborted
0111 31 1152 2153 CDRP$B_FLAGS(R5),5$
1155 2154 ONL_ABORT_ERR
1158 2155
50 E4 A5 E0 A5 C3 1158 2156 5$: SUBL3 CDRP$L_ABCNT(R5),-
115E 2157 CDRP$L_OBCNT(R5),R0 ; Calc how much left to do
115E 2158 BLEQ 30$ ; None, we are finished
51 D2 A5 F7 8F 78 1160 2159 ASHL #9,CDRP$L_BCNT(R5),R1 ; Select the correct disk block
51 48 A5 C0 1166 2160 ADDL CDRP$L_LBN(R5),R1 ; Compute next start LBN
48 A5 51 D0 116A 2161 MOVL R1,CDRP$L_LBN(R5) ; Store it away
D2 A5 D1 116E 2162 CMPL CDRP$L_BCNT(R5),- ; If the bytecount has been reduced
B4 A5 1171 2163 CDRP$L_BUFLN(R5) ; as a result of error recovery...
05 1E 1173 2164 BGEQU 10$ ; Then
B4 A5 D0 1175 2165 MOVL CDRP$L_BUFLN(R5),- ; Set it back to original
D2 A5 50 D1 1178 2166 CDRP$L_BCNT(R5)
D2 A5 04 18 117E 2168 10$: CMPL R0,CDRP$L_BCNT(R5) ; If piece left less than BCNT
D2 A5 50 D0 1180 2169 BGEQ 20$ ; .. then ..
21 11 1184 2170 20$: MOVL R0,CDRP$L_BCNT(R5) ; Use it
BRB 60$
1186 2171 ;
1186 2172 ; Finished all the copy specified in the packet, now check if a possible
1186 2173 ; second pass is needed because the exclude area is in the middle of the disk.
1186 2174 ;
03 02 E4 1186 2175 30$: BBSC #UQB$V_2PASS,- ; Skip if no second pass needed
2B A4 1188 2176 UQB$B_FLAGS(R4),50$
0053 31 118B 2177 40$: BRW ONL_FINI
118E 2178
50 14 A2 7D 118E 2179 50$: MOVQ MSCP$L_EXCL_LBA(R2),R0 ; Pick the exclude parameters
51 50 C0 1192 2180 ADDL R0,R1 ; Starting LBN after exclude area
48 A5 51 D0 1195 2181 MOVL R1,CDRP$L_LBN(R5) ; Put it in CDRP
0101 30 1199 2182 BSBW GET_MAX_BLOCK ; Find the maximum blocks to copy
50 50 09 78 119C 2183 ASHL #9,R0,R0 ; Convert to bytes for next copy loop
E4 A5 50 D0 11A0 2184 MOVL R0,CDRP$L_OBCNT(R5) ; Store in IRP
E0 A5 D4 11A4 2185 CLRL CDRP$L_ABCNT(R5) ; Reset accumulated count
11A7 2186 ;
11A7 2187 ; Another round is needed. Check if there is a write pending, if so
11A7 2188 ; stall the copy and do the write. At the end of the write, the copy
11A7 2189 ; will be restarted.
11A7 2190 ;
11A7 2191 ; R0 = #bytes left to copy
11A7 2192 ; R1 = LBN to start copy at
11A7 2193 ; R4 --> Master UQB
11A7 2194 ;
03 03 E4 11A7 2195 60$: BBSC #UQB$V_CMDPEND,- ; Go around again if no write pend
2B A4 11A9 2196 UQB$B_FLAGS(R4),70$
FF33 31 11AC 2197 BRW ONL_READ
11AF 2198
0C A2 50 D0 11AF 2199 70$: MOVL R0,MSCP$L_BYTE_CNT(R2) ; Update copy in packet
1C A2 51 D0 11B3 2200 MOVL R1,MSCP$L_LBN(R2) ; Ditto
58 A4 01 CE 11B7 2201 MNEGL #1,UQB$L_FENCEL(R4) ; Set special value for tests in SETUP
OC A5 FF10 CF 9E 11BB 2202 MOVAB ONL_LOOP,CDRP$L_FPC(R5) ; Store resume point
10 8E 11C1 2203 BISB #UQB$M_SEQ,- ; Set sequential mode while waiting
```

```
2B A4 11C3 2204 UQBSB_FLAGS(R4)
00F0 30 11C5 2205 BSBW BLOCK_CDRP ; Block the CDRP
11C8 2206
11C8 2207 : Now empty the special holding queue.
11C8 2208
11C8 2209 EMPTY_SHQ:
50 70 B4 0F 11C8 2210 REMQUE @UQBSQ_SHQ(R4),R0 ; Remove from special que
12 1D 11CC 2211 BVS 10$ ; None, leave
55 18 A0 DE 11CE 2212 MOVAL CDRP$L_FQFL- -
11D2 2213 CDRP$L_UQB_FL(R0),R5 ; Set new CDRP ptr
11D2 2214 PUSHL R4 ; Save old UQB
53 10 A5 7D 11D4 2215 MOVQ CDRP$L_FR3(R5),R3 ; Restore old context
OC B5 16 11D8 2216 JSB @CDRP$L_FPC(R5) ; Finish off the pkt
54 8ED0 11DB 2217 POPL R4 ; Restore old UQB
E8 11 11DE 2218 BRB EMPTY_SHQ ; No, get next command
11E0 2219
05 11E0 2220 10$: RSB ; Finish this thread
11E1 2221
11E1 2222 : All copying is done. Clean up status flags and do the copy of parameters.
11E1 2223 : At this point the CDRP is still queued to shadow master UQB. We must
11E1 2224 : remque it, and adjust other fields so this CDRP looks like it was on
11E1 2225 : the physical unit que.
11E1 2226
50 D4 11E1 2227 ONL_FINI:
11E1 2228 CLRL R0 ; Set good status
11E3 2229
11E3 2230 ONL_FINIX:
50 50 DD 11E3 2231 PUSHL R0 ; Save status
E8 A5 0F 11E5 2232 REMQUE CDRP$L_UQB_FL(R5),R0 ; Strip it from que
60 D4 11E9 2233 CLRL (R0) ; show it not queued
50 A4 B7 11EB 2234 DECB UQBSW_NUM_QUE(R4) ; and account for it
009B 30 11EE 2235 BSBW CLEAN_UP_ONLCPY ; Do a clean up
D5 10 11F1 2236 BSBB EMPTY_SHQ ; Finish off and held commands
50 8ED0 11F3 2237 POPL R0 ; Recover status
51 B0 A5 D0 11F6 2238 MOVQ CDRP$L_MSGBUF_X(R5),R1 ; Recover MSCP pointer
54 24 A1 D0 11FA 2239 MOVQ WRT_UQB(R1),R4 ; and write unit UQB pointer
78 A5 54 D0 11FE 2240 MOVQ R4,CDRP$L_MST_UQB(R5) ; Stuff it CDRP
0088 C4 D4 1202 2241 CLRL UQBS$L_CPY_CDRP(R4) ; Clear out pointer
010B 31 1206 2242 BRW COPY_CHAR ; and leave
```

```
1209 2244 .SBTTL - ERROR HANDLING
1209 2245
1209 2246 :++
1209 2247 :
1209 2248 : Read error recovery for bad blocks. First we attempt an ordinary read
1209 2249 : read repair operation. That operation can return with four cases:
1209 2250 :
1209 2251 : 1) RO = $$$_NORMAL: Read partially succeeded, up to a parity or
1209 2252 : forcederror, but all blocks before the one in error have
1209 2253 : been successfully read into our buffer. Continue the loop
1209 2254 : and write out the good blocks to the target unit. The next
1209 2255 : read will either succeed or will encounter a forcederror/
1209 2256 : parity error immediately on the first block. If the latter
1209 2257 : then we retry until success or no more units. If no more
1209 2258 : units we will return here with $$$_PARITY or $$$_FORCEDERROR.
1209 2259 :
1209 2260 : 2) RO = $$$_SHACHASTA: An unrepairable, fatal error occurred on one
1209 2261 : of the source units or the target unit. We abort the copy
1209 2262 : operation, failout the offending unit, and report to host.
1209 2263 :
1209 2264 : 3) RO = $$$_PARITY: This error means we have a parity error in the
1209 2265 : same LBN of all the source units. We then just write out
1209 2266 : one block of whatever is in the buffer to the target unit.
1209 2267 : Hopefully, the host's file structure will be able to deal
1209 2268 : with this crazy block. After the one block write, we continue
1209 2269 : the normal loop. The shadow set remains intact.
1209 2270 :
1209 2271 : 4) RO = $$$_FORCEDERROR: As case 3 above, only the block is written
1209 2272 : out with a FORCERR modifier. The host's file structure will
1209 2273 : eventually catch this.
1209 2274 :
1209 2275 :--
1209 2276 :
1209 2277 .ENABLE LSB
1209 2278
1209 2279 ONL_READ_ERR:
1209 2280 MOVAB ONL_READ,CDRPSL_FPCX(R5); Where to send blocked reads
1209 2281 BSBW READ_REPAIR ; Attempt read repair
1209 2282 BLBC RO,10$ ; A failure of some sort
1209 2283 MOVL CDRPSL_MSGBUFX(R5),R2 ; Refresh MSCP packet pointer
1209 2284 MOVL CDRPSL_UQB(R5),- ; Pick up new read UQB
1209 2285 RED_UQB(R2) ; and refresh the copy
1209 2286 PUSHL R5 ; Save R5
1209 2287 MOVL MST_UQB(R2),R4 ; Pick up master UQB
1209 2288 BSBW DRAIN_BLOCK0 ; See if anything blocked while repair
1209 2289 POPL R5 ; Restore R5
1209 2290 BRB 30$ ; Repair succeeded, continue
1209 2291 :
1209 2292 : Either PARITY or FORCEDERROR. Set up for a one block write.
1209 2293 :
1209 2294 10$: CMPW RO, $$$_PARITY ; Is it parity error?
1209 2295 BEQL 20$ ; Yes, adjust buffer
1209 2296 CMPW RO, $$$_FORCEDERROR ; No, is it forced error?
1209 2297 BNEQ 40$ ; No, real hard error
1209 2298 MOVW #IOSM_FORCERR,- ; Set for writing forced error
1209 2299 CDRPSW_FUNC(R5)
1209 2300 20$: MOVZWL #512,CDRPSL_BCNT(R5) ; Adjust byte counts
```

```
50 A5 FED5 CF 9E 1209 2280
      039D 30 120F 2281
      17 50 E9 1212 2282
52 B0 A5 D0 1215 2283
      C4 A5 D0 1219 2284
      28 A2 121C 2285
      55 DD 121E 2286
54 2C A2 D0 1220 2287
      F94E 30 1224 2288
      55 8ED0 1227 2289
      1A 11 122A 2290
      122C 2291
      122C 2292
      122C 2293
01F4 8F 50 B1 122C 2294
      0D 13 1231 2295
2144 8F 50 B1 1233 2296
      26 12 1238 2297
      0100 8F B0 123A 2298
      C0 A5 123E 2299
D2 A5 0200 8F 3C 1240 2300
```

```
FED8 31 1246 2301 30$: BRW ONL_WRITE ; Return inline
1249 2302
1249 2303
1249 2304 : Write error recovery for bad blocks
1249 2305
1249 2306 ONL_WRITE_ERR:
01F4 8F 50 B1 1249 2307 CMPW R0,#SS$ _PARITY ; Is it parity error?
10 12 124E 2308 BNEQ 40$ ; No, hard error
51 DA A5 F7 8F 78 1250 2309 ASHL #-9,CDRPSL_IOST1+2(R5),R1 ; Number of blocks transfered
51 D6 1256 2310 INCL R1 ; Skip over bad block
D2 A5 51 09 78 1258 2311 ASHL #9,R1,CDRPSL_BCNT(R5) ; Update number of bytes to copy
FEE3 31 125D 2312 BRW ONL_LOOP_CHK ; Return inline
1260 2313
1260 2314 : Fatal error - take care of shadow set housekeeping
1260 2315
54 C4 A5 D0 1260 2316 40$: MOVL CDRPSL_UQB(R5),R4 ; Pick up pointer to phy unit
FAE0 30 1264 2317 BSBW AVAIL_ALL ; Throw it out of shadow set
02 11 1267 2318 BRB 50$ ; Finish up
1269 2319
1269 2320 ONL_ABORT_ERR:
52 54 D4 1269 2321 CRL R4 ; Always want the write unit
B0 A5 D0 126B 2322 50$: MOVL CDRPSL_MSGBUFX(R5),R2 ; Recover MSCP pointer
50 24 A2 D0 126F 2323 MOVL WRT_UQB(R2),R0 ; Pick up write (target) UQB
54 50 D1 1273 2324 CMPL R0,R4 ; Was the error on the write?
06 13 1276 2325 BEQL 60$ ; Yes, then just fail it out
54 50 D0 1278 2326 MOVL R0,R4 ; No, we must also fail write out
FAC9 30 127B 2327 BSBW AVAIL_ALL ; Throw it out of shadow set
50 50 3C 127E 2328 60$: MOVZWL #MSCP$K_ST_SHST,R0 ; Set error code
B0 A5 D0 1281 2329 MOVL CDRPSL_MSGBUFX(R5),R2 ; Recover MSCP pointer
54 2C A2 D0 1285 2330 MOVL MST_UQB(R2),R4 ; Recover master UQB pointer
FF57 31 1289 2331 BRW ONL_FINISH ; Leave
128C 2332
128C 2333 .DISABLE LSB
128C 2334
128C 2335 : CLEAN_UP_ONLCOPY
128C 2336
128C 2337 CLEAN_UP_ONLCOPY:
52 B0 A5 D0 128C 2338 MOVL CDRPSL_MSGBUFX(R5),R2 ; Recover MSCP pointer
11 8A 1290 2339 BICB #<UQBSM_CIP!UQBSM_SEQ>,- ; Clear flags on shadow unit
2B A4 1292 2340 UQBSB_FLAGS(R4)
51 24 A2 D0 1294 2341 MOVL WRT_UQB(R2),R1 ; Grab write unit UQB pointer
8A 1298 2342 BICB #<UQBSM_CIP!UQBSM_SEQ!- ; Clear flags on physical unit
1299 2343 UQBSM_BLOCKED>,-
2B A1 13 1299 2344 UQBSB_FLAGS(R1)
05 129C 2345 RSB
129D 2346
129D 2347 : GET_MAX_BLOCK
129D 2348
129D 2349 :
129D 2350 .ENABLE LSB
129D 2351
129D 2352 GET_MAX_BLOCK:
50 48 A4 52 DD 129D 2353 PUSH R2 ; Get a register
52 30 A4 D0 129F 2354 SUBL3 R1,UQBSL_MAXBLOCK(R4),R0 ; Calculate number of blocks to copy
08 E0 12A4 2355 MOVL UQBSL_UCB(R4),R2 ; Pick up UCB pointer
07 38 A2 12A8 2356 BBS #DEV$V_RCT,- ; Skip next if no last track device
12AA 2357 UCB$ _DEV$CHAR(R2),10$
```

```
52 44 A2 9A 12AD 2358 MOVZBL UCBSB_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,CDRPSL_FR3(R5) ; Save R3 & R4
50 E8 A5 0F 12B8 2367 REMQUE CDRPSL_QUE_FL(R5),R0 ; Remove from head of execute queue and
50 A4 B7 12C0 2368 DECW UQBSW_NUM_QUE(R4) ; (make count agree)
68 A4 60 0E 12C3 2369 INSQUE (R0),UQBSW_BLOCKQ(R4) ; insert at head of block queue
7C A4 B6 12C7 2370 INCW UQBSW_NUM_BLK(R4) ; (make count agree)
D6 A5 08 B0 12CA 2371 MOVW #B,CDRPSW_STATE(R5) ; Show proper state
12CE 2372
12CE 2373 DEALLOC_BUFF:
E4 FA A5 E5 12CE 2374 BBCC #CDRPSV_BUFFER,- ; Clear the buffer flag
12D0 2375 CDRPSB_FLAGS(R5),20$
51 B4 A5 D0 12D3 2376 MOVL CDRPSL_BUFLLEN(R5),R1 ; Pick up the buffer
50 B8 A5 D0 12D7 2377 MOVL CDRPSL_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 @#EXES$DEALLOCATE ; Get rid of buffer
12F8 2388
12F8 2389 70$: BUG_CHECK MSCPSERV,FATAL ; Invalid buffer deallocation
12FC 2390
12FC 2391 .DISABLE LSB
```

```
12FC 2393 .SBTTL - 10 - SET UNIT CHARACTERISTICS
12FC 2394 :*****
12FC 2395 :
12FC 2396 # 10 - S E T U N I T C H R
12FC 2397 :
12FC 2398 :*****
12FC 2399 :
12FC 2400 .ENABLE LSB
12FC 2401
12FC 2402 SET_UNIT CHR:
12FC 2403 TSTW R0 ; Check status
12FE 2404 BNEQ 50$ ; Either AVAIL or OFFLINE
1300 2405 :
1300 2406 : Unit ONLINE to this host - set the unit characteristics
1300 2407 :
1300 2408 10$: BBS #MSCPSV_MD STWRP,- ; Check for correct modifier
05 0A A1 1302 2409 MSCPSV_MODIFIER(R1),20$
1305 2410 BBCC #MSCPSV_UF WRTPS,- ; Not allowed, remove it
00 0E A1 1307 2411 MSCPSV_UNT_FLGS(R1),20$
12 A4 20 A1 130A 2412 20$: BISW MSCPSV_UNT_FLGS(R1),-
130D 2413 UQBSW_UNT_FLGS(R4) ; Host settable flags
24 A4 20 A1 B0 130F 2414 MOVW MSCPSV_SHDW_UNT(R1), -
1314 2415 UQBSW_SHDW_UNT(R4) ; Shadow unit
1314 2416 :
1314 2417 : Move in returned info to end packet
1314 2418 :
1314 2419 COPY_CHAR:
24 A1 48 A4 D0 1314 2420 MOVL UQBSL_MAXBLOCK(R4), - ; Size
1319 2421 MSCPSL_UNT_SIZE(R1)
1319 2422 BNEQ 40$ ; All okay
52 30 A4 D0 131B 2423 MOVL UQBSL_UCB(R4),R2 ; Pick up UCB pointer
131F 2424 BNEQ 30$ ; There is one
52 78 A4 D0 1321 2425 MOVL UQBSL_SHDW_LOW(R4),R2 ; For shadow master, use first phys unit
52 30 A2 D0 1325 2426 MOVL UQBSL_UCB(R2),R2 ; Pick up UCB pointer
00B0 C2 D0 1329 2427 30$: MOVL UCB$L_MAXBLOCK(R2), - ; Get it from the UCB
24 A1 48 A4 D0 132D 2428 UQBSL_MAXBLOCK(R4)
132F 2429 MOVL UQBSL_MAXBLOCK(R4), - ; Size (try again)
28 A1 2C A4 D0 1334 2430 MSCPSL_UNT_SIZE(R1)
1339 2431 40$: MOVL UQBSL_VOL_SER(R4), -
1339 2432 MSCPSL_VOL_SER(R1) ; VOL_ID
F74F 31 1339 2433 BRW FIN_UNIT_STS ; Finish packet elsewhere
133C 2434
F795 31 133C 2435 50$: BRW SEND_PKT ; Return AVAIL/OFFLINE error
133F 2436
133F 2437 .DISABLE lsb
```

MSCP  
V04-001

- MSCP Server - Emulator  
- 11 - DETERMINE ACCESS PATHS

L 16

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 54  
(1)

```
133F 2439 .SBTTL - 11 - DETERMINE ACCESS PATHS
133F 2440 :*****
133F 2441 :
133F 2442 : # 11 - D E T A C C P A T H
133F 2443 :
133F 2444 :*****
133F 2445 :
133F 2446 DET_ACC_PATH:
F790 31 133F 2447 BRW SEND_SUCCESS ; NOP
```

```
1342 2449 .SBTTL NON-SEQUENTIAL NON-BUFFER class commands
1342 2450 :
1342 2451 : R0 = scratch
1342 2452 : R1 = Ptr to message buffer
1342 2453 : R2 = low 3 bits of command
1342 2454 : R3 = Ptr to HQB
1342 2455 : R4 = Unit number
1342 2456 : R5 = Ptr to CDRP
1342 2457 :
54 78 A5 D0 1342 2458 NSEQ: MOVL CDRP$L_MST UQB(R5),R4 ; Recover UQB pointer
50 0E A4 3C 1346 2459 MOVZWL UQB$W_STATOS(R4),R0 ; Pick up status
      03 13 134A 2460 BEQL 10$ ; Access - ONLINE
      F785 31 134C 2461 BRW SEND_PKT ; Return error code
134F 2462
50 30 A4 D0 134F 2463 10$: MOVL UQB$L_UCB(R4),R0 ; Ucb address
06 64 A0 0E E1 1353 2464 BBC #UCB$V_MNTVERIP,UCB$L_STS(R0),15$ ; Branch if mount verification is not
1358 2465 ; progress
1358 2466 ; mark unit as available
50 04 D0 1358 2467 MOVL #MSCP$K_ST_AVLBL,R0 ; return unit available
      F776 31 135B 2468 BRW SEND_PKT
135E 2469
      93 135E 2470 15$: BITB #<UQB$M_SEQ!- ; Check if in SEQ mode
135F 2471 ; or in repair
135F 2472 ; or blocked
2B A4 32 135F 2473 ; or blocked
      03 13 1362 2474 BEQL 20$ ; If so ...
1364 2475 BLOCKT ; then block the command
1367 2476
1367 2477 20$: CASE CDRP$B_OPCODE(R5),- ; Dispatch on command subtype
1367 2478 TYPE=B,LIMIT=#MSCP$K_OP_ACCES-
1367 2479 <ACCESS,- ; 16-ACCESS
1367 2480 COMP_CTRL_DAT,- ; 17-COMPARE CONTROLLER DATA
1367 2481 ERASE,- ; 18-ERASE
1367 2482 FLUSH,- ; 19-FLUSH
1367 2483 REPLACE> ; 20-REPLACE
      F580 31 1376 2484 BRW ERR_1
```

```

1379 2486 .SBTTL - 16 - ACCESS
1379 2487 :*****
1379 2488 :
1379 2489 : # 16 - A C C E S S
1379 2490 :
1379 2491 :*****
1379 2492 :
1379 2493 ACCESS:
20 88 1379 2494 BISB #1@CDRPSV,NOHSTBUF,- ; Set for no host buffer
FA A5 137B 2495 CDRPSB_F[AGS(R5)
005D 31 137D 2496 BUG_CHECK -MSCPSERV,FATAL ; ?? Tied off here and in opcode table
1381 2497 BRW- READ ; And join read code

```

MSCP  
V04-001

C 1  
- MSCP Server - Emulator  
- 17 - COMPARE CONTROLLER DATA

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 57  
(1)

MS  
V0

```
1384 2499 .SBTTL - 17 - COMPARE CONTROLLER DATA
1384 2500 :*****
1384 2501 :
1384 2502 : # 17 - C O M P C T R L D A T
1384 2503 :
1384 2504 :*****
1384 2505 :
1384 2506 COMP_CTRL_DAT:
F74B 31 1384 2507 BRW SEND_SUCCESS ; NOP
```

```

1387 2509 .SBTTL - 18 - ERASE
1387 2510 :*****
1387 2511 :
1387 2512 : # 18 - E R A S E
1387 2513 :
1387 2514 :*****
1387 2515 :
1387 2516 ERASE:
      B3 1387 2517 BITW #<MSCP$M-UF-WRTPS!-
1388 2518 MSCP$M-UF-WRTPH>-
1388 2519 UQBSW_UNT_FLGS(R4) ; If WRITE_PROTECT
12 A4 3000 8F 13 138D 2520 BEQL 10$
      50 06 D0 138F 2521 MOVL #MSCP$K-ST-WRTPR, R0 ; Set error status
      F73F 31 1392 2522 BRW SEND_PKT ; ... and exit
      20 88 1395 2523
      FA A5 1397 2524 10$: BISB #1@CDRPSV NOHSTBUF,- ; Set for no host buffer
      50 15 B0 1399 2525 CDRPSB F[AGS(R5)
      0360 31 139C 2526 MOVW #IOS_DSE,R0 ; Indicate write
      DO_WRT ; Join WRITE command

```

```
139F 2529 .SBTTL - 19 - FLUSH
139F 2530 :*****
139F 2531 :
139F 2532 : # 19 - F L U S H
139F 2533 :
139F 2534 :*****
139F 2535 :
139F 2536 FLUSH:
F730 31 139F 2537 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 2554 .SBTTL NON-SEQUENTIAL BUFFER class commands
13A8 2555 :
13A8 2556 : R0 = scratch
13A8 2557 : R1 = Ptr to message buffer
13A8 2558 : R2 = low 3 bits of command
13A8 2559 : R3 = Ptr to HQB
13A8 2560 : R4 = Unit number
13A8 2561 : R5 = Ptr to CDRP
13A8 2562 :
54 78 A5 D0 13A8 2563 NSEQB: MOVL CDRP$L_MST UQB(R5),R4 ; Recover UQB pointer
50 0E A4 3C 13AC 2564 MOVZWL UQB$W_STATUS(R4),R0 ; Pick up status
      03 13 13B0 2565 BEQL 10$ ; Success - ONLINE
      F71F 31 13B2 2566 BRW SEND_PKT ; Return error code
      13B5 2567
50 30 A4 D0 13B5 2568 10$: MOVL UQB$L_UCB(R4),R0 ; Ucb address
06 64 A0 0E E1 13B9 2569 BBC #UCB$V_MNTVERIP,UCB$L_STS(R0),15$ ; Branch if mount verification is not
      13BE 2570 ; progress
      13BE 2571 ; mark unit as available
50 04 D0 13BE 2572 MOVL #MSCP$K_ST_AVLBL,R0 ; return unit available
      F710 31 13C1 2573 BRW SEND_PKT
      13C4 2574
      93 13C4 2575 15$: BITB #<UQB$M_SEQ!- ; Check if in SEQ mode
      13C5 2576 ; or in repair
      13C5 2577 ; or blocked
      13C5 2578 ; or blocked
2B A4 32 13C5 2578 BEQL 20$ ; If so
      03 13 13C8 2579 BLOCKT ; then block the command
      13CA 2580
      13CD 2581
      13CD 2582 20$: CASE CDRP$B_OPCODE(R5),- ; Dispatch on command subtype
      13CD 2583 TYPE=B,LIMIT=#MSCP$K_OP_COMP-
      13CD 2584 <COMP_HOST_DAT,- ; 32-COMPARE HOST DATA
      13CD 2585 READ,- ; 33-READ
      13CD 2586 WRITE> ; 34-WRITE
      F54E 31 13D8 2587 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 #IOS_WRITECHECK,R0 ; Set function code
031E 31 13DE 2598 BRW DO_WRT ; and go do it
```

```
13E1 2600 .SBTTL - 33 - READ
13E1 2601 *****
13E1 2602 *****
13E1 2603 # 33 - R E A D
13E1 2604 *****
13E1 2605 *****
13E1 2606 *****
13E1 2607 READ:
50 0C B0 13E1 2608 MOVW #IOS_READPBLK,R0 ; Set up for read
03D8 30 13E4 2609 BSBW SETUP ; Common set up
20 93 13E7 2610 BITB #1@CDRPSV_NOHSTBUF,- ; Test flags
FA A5 13E9 2611 CDRPSB_FLAGS(R5)
07 12 13EB 2612 BNEQ READ_LOOP ; Skip if no host buffer
D6 A 01 B0 13ED 2613 MOVW #1,CDRPSW_STATE(R5) ; Show it in mapping resource wait
13F1 2614 MAP_IRP ; Map the buffer
13F4 2615 ;
13F4 2616 ; Read the data from the disk
13F4 2617 ;
13F4 2618 READ_LOOP:
D6 A5 02 B0 13F4 2619 MOVW #2,CDRPSW_STATE(R5) ; Set the state
0662 30 13F8 2620 BSBW DO_DISK ; Set up for disk transfer
10 93 13FB 2621 BITB #1@CDRPSV_ABORT,- ; Check if aborted
FA A5 13FD 2622 CDRPSB_FLAGS(R5)
38 12 13FF 2623 BNEQ FINI ; Leave if so
07 50 E8 1401 2624 BLBS R0,READ_CONT ; Success
0168 31 1404 2625 BRW READ_ERR ; Leave now for error
1407 2626
1407 2627 READ_INVALID_CDT:
1407 2628 BUG_CHECK MSCPSERV,FATAL ; Stale CDT
1408 2629 ;
1408 2630 ; Send the data to the host buffer
1408 2631 ;
1408 2632 READ_CONT:
20 93 140B 2633 BITB #1@CDRPSV_NOHSTBUF,- ; Test flags
FA A5 140D 2634 CDRPSB_FLAGS(R5)
28 12 140F 2635 BNEQ FINI ; Skip if no host buffer
D6 A5 03 B0 1411 2636 MOVW #3,CDRPSW_STATE(R5) ; Show it queued to SCS block transfer
1415 2637 ; ??? Begin bug trap
52 B0 A5 D0 1415 2638 MOVL CDRPSL_MSGBUF(R5),R2 ; Get message buffer address for validity ch
51 24 A5 D0 1419 2639 MOVL CDRPSL_CDT(R5),R1 ; Get CDT address
F8 A2 18 A1 D1 141D 2640 CMPL CDTSL_CONID(R1),- ; Check validity of remote conid
1422 2641 SCSSL_DST_CONID(R2)
E3 12 1422 2642 BNEQ READ_INVALID_CDT ; Trying to use stale connection
1424 2643 ; ??? End bug trap
1C A5 D4 1424 2644 CLRL CDRPSL_MSG_BUF(R5) ; Mark it not in use
1427 2645 SEND_DATA ; Send to host buffer
46 50 E9 1436 2646 BLBC R0,ABX_ER ; Aborted
F4 A5 D7 1439 2647 DECL CDRPSL_CMD_STS(R5) ; Set the command status
B5 AF 9F 143C 2648 FINI: PUSHAB READ_LOOP ; Loop again
143F 2649
143F 2650 ;
143F 2651 ; TST_END: Update the accumulated byte count and compare to original to
143F 2652 ; determine if another round is needed. Also check for errors and translate
143F 2653 ; VMS errors to MSCP errors. Finally, clean up resources and branch off
143F 2654 ; to return end packet with status.
143F 2655 ;
143F 2656 .ENABLE LSB
```

```

143F 2657
53 DA A5 D0 143F 2658      MOVL  CDRP$L_IOST1+2(R5),R3  ; Pick up transfered bytes
      04 11 1443 2659      BRB    10$                ; Join common code
      1445 2660
      1445 2661 TST_END:
51 53 D2 A5 D0 1445 2662      MOVL  CDRP$L_BCNT(R5),R3      ; Pick up transfered bytes
      F7 8F 78 1449 2663 10$: ASHL  #-9,R3,R1          ; Make it blocks
      E0 A5 53 C0 144E 2664      ADDL  R3,CDRP$L_ABCNT(R5)    ; Calc accumulated bytes xfer
      10 93 1452 2665      BITB   #1,CDRP$V_ABORT,-        ; Check if aborted
      FA A5 25 12 1454 2666      CDRP$B_FLAGS(R5)
      29 50 E9 1456 2667      BNEQ   ABO_ER                ; Leave if so
      50 D4 1458 2668      BLBC    R0,OUT_ER                ; Leave if error
      50 D4 1458 2669      CLRL    R0                      ; Assume success
52 E4 A5 E0 A5 C3 145D 2670      SUBL3  CDRP$L_ABCNT(R5),-    ; Calc how much left to do
      1463 2671      CDRP$L_OBCNT(R5),R2
      D2 A5 52 D1 1465 2672      BLEQ   OUT                ; None, we are finished
      08 18 1469 2673      CMPL    R2,CDRP$L_BCNT(R5)        ; If piece left less than BCNT
      D2 A5 52 D0 146B 2674      BGEQ   20$                ; .. then ..
      3C A5 52 D0 146F 2675      MOVL  R2,CDRP$L_BCNT(R5)    ; Use it
      38 A5 E0 A5 D0 1473 2677 20$: MOVL  R2,CDRP$L_XCT_LEN(R5) ; Ditto
      48 A5 51 C0 1478 2678      MOVL  CDRP$L_ABCNT(R5),-    ; Set the remote byte offset
      05 147C 2680      ADDL  R1,CDRP$L_LBN(R5)              ; to be accumulated xfer size
      147D 2681      RSB                    ; Update LBN
      147D 2682      .DISABLE LSB
      147D 2683      ; Command was aborted while on disk queue
      147D 2684
      147D 2685
      8E D5 147D 2686 ABO_ER: TSTL  (SP)+                ; Clean stack
      147F 2687
      147F 2688      ; Command was aborted during SEND/RECEIVE DATA to host
      147F 2689
      50 02 D0 147F 2690 ABX_ER: MOVL  #MSCP$K_ST_ABRTD,R0    ; Set status
      2A 11 1482 2691      BRB    OUT1
      1484 2692
      1484 2693      ; Command received a disk error - translate from VMS to MSCP status.
      1484 2694
      8E D5 1484 2695 OUT_ER: TSTL  (SP)+                ; Get rid of return address
      04 E0 1486 2696      BBS     #CDRP$V_ABORT,-          ; Check if aborted
      F4 FA A5 1488 2697      CDRP$B_FLAGS(R5),ABX_ER      ; Leave if so (has priority over errors)
      14F6 CF 3E 148B 2698      MOVAW  ERR_TBL,R2          ; Pick up error table address
      51 82 3C 1490 2699 10$: MOVZWL (R2)+,R1              ; Grab VMS error entry
      0A 13 1493 2700      BEQL    20$                    ; End of table, quit
      50 51 B1 1495 2701      CMPW   R1,R0                  ; Try for a match
      05 13 1498 2702      BEQL    20$                    ; Success
      52 04 C0 149A 2703      ADDL    #4,R2                 ; Step to next entry
      F1 11 149D 2704      BRB     10$                    ; Loop
      50 82 3C 149F 2705      20$: MOVZWL (R2)+,R0          ; Pick up MSCP error
      51 62 32 14A2 2707      CVTWL  (R2),R1              ; Get possible routine
      07 13 14A5 2708      BEQL    OUT1                    ; None
      6142 16 14A7 2709      JSB     (R1)(R2)              ; Invoke the routine
      02 11 14AA 2710      BRB     OUT1                    ; Continue on
      14AC 2711
      14AC 2712      ; Final clean up for transfer commands. Set status, release resources.
      14AC 2713
```

```
52  B0 A5 D0 14AC 2714 OUT: TSTL (SP)+ ; Get rid of return
      09 13 14AE 2715 OUT1: MOVL CDRP$L_MSGBUF(X(R5),R2 ; Pick up old message buffer ptr
      E0 A5 D0 14B2 2716 BEQL 10$ ; None, probably an abort
      0C A2 D0 14B4 2717 MOVL CDRP$L_ABCNT(R5),- ; Set byte count
      50 B0 14B7 2718 MSCP$C_BYTE_CNT(R2)
      A5 05 B0 14B9 2719 MOVW R0,MSCP$W_STATUS(R2) ; Store the status
      20 B0 14BD 2720 10$: MOVW #5,CDRP$W_STATE(R5) ; Set state
      FA A5 93 14C1 2721 BITB #1@CDRP$V_NOHSTBUF,-
      0B 12 14C3 2722 CDRP$B_FLAGS(R5)
      60 A5 D0 14C5 2723 BNEQ 20$ ; Skip if no host buffer
      10 A3 D0 14C7 2724 MOVL CDRP$L_CDTX(R5),R3 ; Pick up the CDT address
      4C A5 D0 14CB 2725 MOVL CDT$L_PDT(R3),R4 ; Set up PDT address
      50 54 D0 14CF 2726 UNMAP ; Get rid of mapping resources
      0A 13 14D2 2727 20$: MOVL CDRP$L_SHDW_LNK(R5),R4 ; Check possible IRP link
      F169 30 14D6 2728 BEQL 40$ ; None, continue on
      F1 11 14D8 2729 30$: MOVL R4,R0 ; Transfer IRP pointer
      54 00AC C4 D0 14DB 2730 BEQL 40$ ; Done
      30 14DD 2731 MOVL IRP$L_SHDW_LNK(R4),R4 ; Transfer next IRP pointer
      11 14E2 2732 BSBW DEALMEM ; Get rid of it
      52 B0 A5 D0 14E5 2733 BRB 30$ ; Loop for more IRP's
      06 13 14E7 2734 40$: MOVL CDRP$L_MSGBUF(X(R5),R2 ; Pick up old message buffer ptr
      1C A2 D0 14EB 2735 BEQL 50$ ; None, probably an abort
      F5EB 31 14ED 2736 CLRL MSCP$L_FRST_BAD(R2) ; Clear unused field
      31 14F0 2737 BRW SEND_ORT_STS ; Return the end packet
      F62B 31 14F3 2739 50$: BRW CLEAN_UP ; Finish up
      14F6 2741
      14F6 2742 ERR_TBL:
      14FC 2743 ACTION SSS_MEDOFL,MSCP$K_ST_AVLBL,ERR_AVAIL
      1502 2744 ACTION SSS_VOLINV,MSCP$K_ST_AVLBL,ERR_AVAIL
      1508 2745 ACTION SSS_WRITLCK,MSCP$K_ST_WRTPR,ERR_WRITLCK
      150E 2746 ACTION SSS_DATAHECK,MSCP$K_ST_COMP
      1514 2747 ACTION SSS_CTRLERR,MSCP$K_ST_CNTRLR
      151A 2748 ACTION SSS_FORMAT,MSCP$K_ST_MFMT
      1520 2749 ACTION SSS_FORCEDERROR,MSCP$K_ST_DATA
      1526 2750 ACTION SSS_PARITY,<<MSCP$K_ST_DATA>!<1*MSCP$K_ST_SBCOD>>
      152C 2751 ACTION SSS_IVBUFLN,MSCP$K_ST_HSTBF
      1532 2752 ACTION SSS_SHACHASTA,MSCP$K_ST_SHST,ERR_SHACHA
      1538 2753 ACTION SSS_TIMEOUT,<MSCP$K_ST_OFFLN!MSCP$K_SC_UNKNO>,ERR_OFFLINE
      153E 2754 ACTION 0,MSCP$K_ST_DRIVE ; PATCH SPACE
      1544 2755 ACTION 0,MSCP$K_ST_DRIVE ; End of table - default error
      1544 2756
      54 C4 A5 D0 1544 2757 ERR_AVAIL:
      04 B0 1544 2758 MOVL CDRP$L_UQB(R5),R4 ; Pick up pointer
      0E A4 B0 1548 2759 MOVW #MSCP$K_ST_AVLBL,- ; Set new status in unit
      05 154A 2760 UQB$W_STATUS(R4)
      154C 2761 RSB
      154D 2762
      54 C4 A5 D0 154D 2763 ERR_OFFLINE:
      03 B0 154D 2764 MOVL CDRP$L_UQB(R5),R4 ; Pick up pointer
      0E A4 B0 1551 2765 MOVW #MSCP$K_ST_OFFLN,- ; Set new status in unit
      05 1553 2766 UQB$W_STATUS(R4)
      1555 2767 RSB
      1556 2768
      51 C4 A5 D0 1556 2769 ERR_WRITLCK:
      1556 2770 MOVL CDRP$L_UQB(R5),R1 ; Pick up pointer
```

```
00 12 A1 0D E2 155A 2771 BBSS #MSCPSV UF WRTPH, -
                                155F 2772 UQBSW_UNT_FLGS(R1),20$ ; Set in unit flags
                                05 155F 2773 20$: RSB
                                1560 2774
                                1560 2775 ERR_SHACHA:
                                1560 2776 TSTL (SP)+ ; Remove return address
54 C4 A5 D5 1560 2776 MOVL CDRPSL_UQB(R5),R4 ; Pick up pointer
    F7DE 30 1566 2777 BSBW AVAIL ALL ; Kick unit out of shadow set
    50 OC D0 1569 2779 MOVL #MSCPSK_ST_SHST,R0 ; Restore status
    FF3F 31 156C 2780 BRW OUT1 ; Continue
                                156F 2781
                                156F 2782 ;
                                156F 2783 ; Error recovery routine for shadow reads. First, verify that this is
                                156F 2784 ; really a shadow read. If so, attempt the read repair. There are only
                                156F 2785 ; two cases to consider:
                                156F 2786 ;
                                156F 2787 ; 1) R0 = SSS_NORMAL: Read partially succeeded, up to a parity or
                                156F 2788 ; forced error. Continue the SEND_DATA part of the loop
                                156F 2789 ; and retry the read at the failed block. The read will
                                156F 2790 ; either succeed or fail back here with an error.
                                156F 2791 ;
                                156F 2792 ; 2) R0 = failure: The repair/retry logic failed. Terminate the
                                156F 2793 ; read and report the error to host.
                                156F 2794 ;
                                156F 2795 READ_ERR:
51 C4 A5 D0 156F 2796 MOVL CDRPSL_UQB(R5),R1 ; Pick up UQB pointer
52 78 A5 D0 1573 2797 MOVL CDRPSL_MST_UQB(R5),R2 ; Pick up master UQB pointer
    OE E1 1577 2798 BBC #MSCPSV UF_SSMEM, - ; If this unit is shadow member
    30 12 A1 1579 2799 UQBSW_UNT_FLGS(R1),20$ ; and
    52 51 D1 157C 2800 CMPL R1,R2 ; If IO directed to shadow master
    2B 13 157F 2801 BEQL 20$ ; then
50 A5 FE6F CF 9E 1581 2802 MOVAB READ_LOOP,CDRPSL_FPCX(R5) ; Where to send blocked reads
    26 10 1587 2803 BSBB READ_REPAIR ; Attempt read repair
    20 50 E9 1589 2804 BLBC R0,20$ ; Failed to find good block
51 DA A5 F7 8F 78 158C 2805 ASHL #-9,CDRPSL_IOST1+2(R5),R1 ; Find how much was transferred
3C A5 51 09 78 1592 2806 ASHL #9,R1,CDRPSL_XCT_LEN(R5) ; Back to bytes
    B4 A5 D0 1597 2807 MOVL CDRPSL_BUFLER(R5),- ; Restore the old bytecnt
    D2 A5 159A 2808 CDRPSL_BCNT(R5)
    05 E1 159C 2809 BBC #CDRPSV_NOHSTBUF, - ; If this is a passthru,
    08 FA A5 159E 2810 CDRPSB_FLAGS(R5),10$
51 51 02 78 15A1 2811 ASHL #2,R1,RT ; Make it number of longwords
    CC A5 51 C0 15A5 2812 ADDL R1,CDRPSL_SVAPTE(R5) ; Update PTE pointer
    FE5F 31 15A9 2813 10$: BRW READ_CONT ; Succeeded in repair, continue SEND
    15AC 2814
    FE8D 31 15AC 2815 20$: BRW FINI ; Leave for good
```

```
15AF 2817 .SBTTL - READ REPAIR routine
15AF 2818
15AF 2819 :++
15AF 2820 :
15AF 2821 : READ_REPAIR
15AF 2822 :
15AF 2823 : Read error recovery for bad blocks. This operation has two phases:
15AF 2824 :
15AF 2825 : Phase 1: If error is not SSS_PARITY or SSS_FORCEDERROR, then
15AF 2826 : leave immediately with R0 = SSS_SHACHASTA. Next we check
15AF 2827 : the bytecnt. If at least one block has been successfully
15AF 2828 : read, then we leave with R0 = SSS_NORMAL after updating
15AF 2829 : transfer parameters to reflect a transfer of up to but not
15AF 2830 : including the bad block. The next read will then cause us
15AF 2831 : to return here for phase 2. Otherwise we drop immediately
15AF 2832 : into phase 2.
15AF 2833 :
15AF 2834 : Phase 2: We enter with an immediate error in the first block. We
15AF 2835 : can return with four cases:
15AF 2836 :
15AF 2837 : 1) R0 = SSS_NORMAL: The block in error on all source units has been
15AF 2838 : successfully repaired if a FORCEDERROR or a good block on an
15AF 2839 : alternate unit has been found for PARITY error. We leave
15AF 2840 : with transfer parameters indicating a good one block read.
15AF 2841 :
15AF 2842 : 2) R0 = SSS_SHACHASTA: An unrepairable, fatal error occurred on one
15AF 2843 : of the source units or the target unit. The calling routine
15AF 2844 : must fail the offending unit out of the shadow set.
15AF 2845 :
15AF 2846 : 3) R0 = SSS_PARITY: We have a parity error in the same LBN of all
15AF 2847 : the source units. No repair is possible. The shadow set
15AF 2848 : remains intact.
15AF 2849 :
15AF 2850 : 4) R0 = SSS_FORCEDERROR: As case 3 above, only a forcederror.
15AF 2851 :
15AF 2852 : Any error that occurs while a repair is pending is put on the blocked queue
15AF 2853 : and will be restarted from the read when the repair operation is finished.
15AF 2854 : This routine is non-abortable.
15AF 2855 :
15AF 2856 : Inputs:
15AF 2857 : R0 = Disk failure code
15AF 2858 : R1 --> Failing UQB
15AF 2859 : R2 --> Master UQB
15AF 2860 : R3/R4 saved
15AF 2861 : R5 --> CDRP
15AF 2862 : CDRP$L_UQB --> Failing unit UQB
15AF 2863 : CDRP$L_FPCX --> Where to restart blocked reads
15AF 2864 :
15AF 2865 : Outputs:
15AF 2866 : R0 = Status code
15AF 2867 : CDRP$L_IOST1 = upated
15AF 2868 :
15AF 2869 : --
15AF 2870 :
15AF 2871 : READ_REPAIR:
51 C4 A5 D0 15AF 2872 : MOVL CDRP$L_UQB(R5),R1 : Pick up UQB pointer
52 78 A5 D0 15B3 2873 : MOVL CDRP$L_MST_UQB(R5),R2 : Pick up master UQB pointer
```

```
2144 8F 50 B1 15B7 2874 CMPW R0,#SS$_FORCEDERROR ; Is this a repairable error?
10 13 15BC 2875 BEQL 10$ ; Yes, go off to repair logic
01F4 8F 50 B1 15BE 2876 CMPW R0,#SS$_PARITY ; Is it parity error
09 13 15C3 2877 BEQL 10$ ; Yes, go off to repair logic
DA A5 D4 15C5 2878 CLRL CDRP$L_IOST1+2(R5) ; No bytes transfered to host
50 2284 8F 3C 15C8 2879 MOVZWL #SS$_SHACHASTA,R0 ; Set fatal error
05 05 15CD 2880 RSB ; Leave
15CE 2881 ;
15CE 2882 ; Set RIP mode (rundown) for the shadow unit. If this is not the first
15CE 2883 ; error for this unit, hold it on the blocked queue until the first
15CE 2884 ; repair is finished. ALL REPAIR OPERATIONS MUST BE SEQUENTIAL!!!
15CE 2885 ; When the repair is finished, these operations will be restarted.
15CE 2886 ;
5C A5 8ED0 15CE 2887 10$: POPL CDRP$L_SAVD_RTX(R5) ; Save return address
05 E1 15D2 2888 BBC #UQBS$_RIP,- ; If already a RIP
1C 2B A2 15D4 2889 UQBS$_FLAGS(R2),20$
54 52 D0 15D7 2890 MOVL R2,R4 ; Transfer to proper register
50 E8 A5 0F 15DA 2891 REMQUE CDRP$L_UQB_FL(R5),R0 ; Remove from execute queue
50 A4 B7 15DE 2892 DECW UQBS$_NUM_QUE(R4) ; Account for it
15E1 2893 BLOCKT ; Block until first RIP over
E8 A5 0E 15E4 2894 INSQUE CDRP$L_UQB_FL(R5),- ; Insert packet on unit que
64 B4 15E7 2895 @UQBS$_CDRP_BL(R4)
50 A4 B6 15E9 2896 INCW UQBS$_NUM_QUE(R4) ; Account for it
54 58 A5 D0 15EC 2897 MOVL CDRP$L_FR3X(R5),R4 ; Restore R4 (PDT)
50 B5 17 15F0 2898 JMP @CDRP$_FPCX(R5) ; Attempt the read again
15F3 2899 ;
15F3 2900 ; First repairable error on this volume. Check for any I/O. If so
15F3 2901 ; block the RIP CDRP until all other I/O done, saving current state.
15F3 2902 ;
C0 A5 B0 15F3 2903 20$: MOVW CDRP$_FUNC(R5),- ; Save old function code
56 A5 15F6 2904 CDRP$_FR3X+2(R5)
50 DA A5 F7 8F 78 15F8 2905 ASHL #-9,CDRP$_IOST1+2(R5),R0 ; Number of blocks transfered
0D 13 15FE 2906 BEQL 30$ ; None, we are in real repair
D2 A5 50 09 78 1600 2907 ASHL #9,R0,CDRP$_BCNT(R5) ; Update transfer parameter
D2 A5 D0 1605 2908 MOVL CDRP$_BCNT(R5),- ; Also update status
DA A5 1608 2909 CDRP$_IOST1+2(R5)
00C6 31 160A 2910 BRW 140$ ; Leave now, first part done
160D 2911 ;
20 88 160D 2912 30$: BISB #UQBS$_RIP,- ; Set the repair in progress flag
2B A2 160F 2913 UQBS$_FLAGS(R2) ; on shadow master unit
20 88 1611 2914 BISB #UQBS$_RIP,- ; Set the repair in progress flag
2B A1 1613 2915 UQBS$_FLAGS(R1) ; on physical unit
D8 A5 B0 1615 2916 MOVW CDRP$_IOST1(R5),- ; Save old status
54 A5 1618 2917 CDRP$_FR3X(R5)
01 50 A2 B1 161A 2918 CMPW UQBS$_NUM_QUE(R2),#1 ; Check the shadow master for 10
1D 1B 161E 2919 BLEQU 40$ ; None, all okay
54 52 D0 1620 2920 MOVL R2,R4 ; Transfer to proper register
50 E8 A5 0F 1623 2921 REMQUE CDRP$L_UQB_FL(R5),R0 ; Remove from execute queue
50 A4 B7 1627 2922 DECW UQBS$_NUM_QUE(R4) ; Account for it
162A 2923 BLOCKT ; Block until all I/O rundown
E8 A5 0E 162D 2924 INSQUE CDRP$L_UQB_FL(R5),- ; Insert packet on unit que
64 B4 1630 2925 @UQBS$_CDRP_BL(R4)
50 A4 B6 1632 2926 INCW UQBS$_NUM_QUE(R4) ; Account for it
51 C4 A5 D0 1635 2927 MOVL CDRP$_UQB(R5),R1 ; Pick up failed UQB pointer
52 78 A5 D0 1639 2928 MOVL CDRP$_MST_UQB(R5),R2 ; Pick up master UQB pointer
D2 A5 0200 8F 3C 163D 2929 40$: MOVZWL #512,CDRP$_BCNT(R5) ; Only do one block
1643 2930 MOVL R1,CDRP$_STRT_UQB(R5) ; Set a end condition
```

```
1643 2931 :  
1643 2932 : Attempt to find a good copy of the block, reading all units if necessary.  
1643 2933 : Keep a list of units that are unreadable. If no units are readable, we  
1643 2934 : will return the IO with the FORCEDERROR status. If any unit has an  
1643 2935 : unrepairable error, we will fail out with Shadow Set Status has changed.  
1643 2936 :  
50 54 A5 3C 1643 2937 50$: MOVZWL CDRP$L_FR3X(R5),R0 ; Pick up true status code  
03EA 30 1647 2938 BSBW SELECT_NEXT_UNIT ; Get the next unit  
56 13 164A 2939 BEQL 90$ ; None, left error out  
B0 164C 2940 MOVW #<IOS_READPBLK!- ; Set for READ/COMPARE  
164D 2941 IOSM_DATACHECK>,-  
164D 2942 CDRP$W_FUNC(R5)  
CO A5 400C 8F 164D 2943 MOVW #2,CDRP$W_STATE(R5) ; Set the state  
D6 A5 02 B0 1652 2943 DO DISK ; Do the read  
0404 30 1656 2944 BSBW ; Pick up UQB pointer  
51 C4 A5 D0 1659 2945 CDRP$L_UQB(R5),R1 ; Success now go to write  
OC 50 E8 165D 2946 R0,60$ ; Was the error repairable?  
54 A5 50 B1 1660 2947 CMPW R0,CDRP$L_FR3X(R5) ; No, leave with failure  
37 12 1664 2948 BNEQ 80$ ; Set the repair in progress flag  
20 88 1666 2949 BISB #UQB$M_RIP,-  
2B A1 1668 2950 UQB$B_FLAGS(R1)  
D7 11 166A 2951 BRB 50$ ; Try for another unit  
166C 2952 :  
166C 2953 : Read phase succeeded in locating a good copy of the block. Now run thru  
166C 2954 : the list of units with a bad copy and repair them.  
166C 2955 :  
74 A5 51 D0 166C 2956 60$: MOVL R1,CDRP$L_STRT_UQB(R5) ; Set end condition & save good unit UQB  
B0 1670 2957 MOVW #<IOS_WRITEPBLK!- ; Now change to write  
1671 2958 IOSM_DATACHECK>,-  
1671 2959 CDRP$W_FUNC(R5)  
CO A5 400B 8F 1671 2959 70$: BSBW SELECT_NEXT_UNIT ; Get the next unit  
03BB 30 1676 2960 BEQL 130$ ; None left, leave  
41 13 1679 2961 BBCC #UQB$V_RIP,- ; Check for units needing repair  
05 E5 167B 2962 UQB$B_FLAGS(R1),70$  
F6 2B A1 167D 2963 CMPW CDRP$L_FR3X(R5),- ; If parity error,  
54 A5 B1 1680 2964 #SS$_PARITY ; then skip write  
01F4 8F 1683 2965 BEQL 70$ ; since it cannot be written  
EE 13 1686 2966 MOVW #4,CDRP$W_STATE(R5) ; Set the state  
D6 A5 04 B0 1688 2967 BSBW DO DISK ; Do the write  
03CE 30 168C 2968 MOVL CDRP$L_UQB(R5),R1 ; Pick up UQB pointer  
51 C4 A5 D0 168F 2969 BLBS R0,70$ ; Success, look for next unit  
E0 50 E8 1693 2970 CMPW R0,#SS$_WRITLCK ; Was the error writelock?  
025C 8F 50 B1 1696 2971 BEQL 70$ ; If so, ignore it  
D9 13 169B 2972 :  
169D 2973 :  
169D 2974 : A hard, unrepairable error has occurred. Clean up any status and  
169D 2975 : then fail this unit out of shadow set and report status to host.  
169D 2976 :  
50 2284 8F B0 169D 2977 80$: MOVW #SS$_SHACHASTA,R0 ; Set the fatal error code  
51 74 A5 D0 16A2 2978 90$: MOVL CDRP$L_STRT_UQB(R5),R1 ; Get starting point  
09 11 16A6 2979 BRB 110$ ; Do it  
16A8 2980 :  
51 61 D0 16A8 2981 100$: MOVL UQB$L_SHDW_FL(R1),R1 ; Advance to next unit  
74 A5 51 D1 16AB 2982 CMPL R1,CDRP$L_STRT_UQB(R5) ; End?  
06 13 16AF 2983 BEQL 120$ ; Yes  
20 8A 16B1 2984 110$: BICB #UQB$M_RIP,- ; Clear flags  
2B A1 16B3 2985 UQB$B_FLAGS(R1)  
F1 11 16B5 2986 BRB 100$ ; Loop for more  
16B7 2987
```

```
DA A5 D4 16B7 2988 120$: CLRL CDRP$L_I0ST1+2(R5) ; Show nothing transfered
1A 11 16BA 2989 BRB 150$ ; Leave
16BC 2990 ;
16BC 2991 ; Repair is finished. Clean up, restore original parameters and
16BC 2992 ; restart the read from the beginning.
16BC 2993 ;
20 8A 16BC 2994 130$: BICB #UQB$M_RIP,- ; Clear status bit
2B A1 16BE 2995 UQB$B_FLAGS(R1)
51 74 A5 D0 16C0 2996 MOVL CDRP$L_STRT_UQB(R5),R1 ; Recover original UQB
C4 A5 51 D0 16C4 2997 MOVL R1,CDRP$L_UQB(R5) ; Restore it
30 A1 D0 16C8 2998 MOVL UQB$L_UCB(R1),- ; Transfer UCB pointer
BC A5 16CB 2999 CDRP$L_UCB(R5)
DA A5 0200 8F 3C 16CD 3000 MOVZWL #512,CDRP$L_I0ST1+2(R5) ; Show one block read
50 01 3C 16D3 3001 140$: MOVZWL #SS$ NORMAL,R0 ; Set success status
56 A5 B0 16D6 3002 150$: MOVW CDRP$L_FR3X+2(R5),- ; Restore function code
C0 A5 16D9 3003 CDRP$L_FUNC(R5)
52 78 A5 D0 16DB 3004 MOVL CDRP$L_MST_UQB(R5),R2 ; Grab shadow master UQB
20 8A 16DF 3005 BICB #UQB$M_RIP,- ; Clear status bit
2B A2 16E1 3006 UQB$B_FLAGS(R2)
54 60 A5 D0 16E3 3007 MOVL CDRP$L_CDTX(R5),R4 ; Pick up CDT address
54 10 A4 D0 16E7 3008 MOVL CDT$L_PDT(R4),R4 ; Pick up PDT address
5C B5 17 16EB 3009 JMP @CDRP$L_SAVD_RTX(R5) ; Return
```

```
16EE 3011 .SBTTL - 34 - WRITE
16EE 3012 :*****
16EE 3013 :
16EE 3014 : # 34 - W R I T E
16EE 3015 :
16EE 3016 :*****
16EE 3017 :
16EE 3018 .ENABLE LSB
16EE 3019
16EE 3020 WRITE:
16EE 3021 BITW #<MSCP$M-UF-WRTPS!-
16EF 3022 MSCP$M-UF-WRTPH>-
16EF 3023 UQBSW_UNT_FLGS(R4) ; If WRITE_PROTECT
16F4 3024 BEQL 10$
16F6 3025 MOVL #MSCP$K_ST-WRTPR, R0 ; Set error status
16F9 3026 BRW SEND_PKT ; ... and exit
16FC 3027
16FC 3028 10$: MOVW #10$ WRITEPBLK, R0 ; Set up for write
16FF 3029 DO_WRT: BSBW SETUP
1702 3030 BITB #1@CDRPSV_NOHSTBUF,- ; Test flags
1704 3031 CDRP$B_F[AGS(R5)
1706 3032 BNEQ 20$ ; Skip if no host buffer
1708 3033 MOVW #1, CDRP$W_STATE(R5) ; Mapping resource wait state
170C 3034 MAP_IRP ; Map the data
170F 3035 :
170F 3036 : Request data from host buffer
170F 3037 :
170F 3038 20$: MOVW #3, CDRP$W_STATE(R5) ; Bump the state
1713 3039 BITB #1@CDRPSV_NOHSTBUF,- ; Test flags
1715 3040 CDRP$B_F[AGS(R5)
1717 3041 BNEQ 30$ ; Skip if no host buffer
1719 3042 : ??? Begin bug trap
1719 3043 MOVL CDRP$L_MSGBUF(R5), R2 ; Get message buffer address for validity ch
171D 3044 MOVL CDRP$L_CDT(R5), R1 ; Get CDT address
1721 3045 CMPL CDT$L_[CONID(R1), - ; Check validity of remote conid
1726 3046 SCSSL_DST_CONID(R2)
1726 3047 BNEQ 25$ ; Trying to use stale connection
1728 3048 : ??? End bug trap
1728 3049 CLRL CDRP$L_MSG_BUF(R5) ; Mark it not in use
172B 3050 REQUEST_DATA ; Get the data from host
173A 3051 BLBS -R0, 30$ ; Success
173D 3052 BRW ABX_ER ; Aborted
1740 3053
1740 3054 25$: BUG_CHECK MSCPSERV, FATAL ; Trying to use stale CDT
1744 3055 :
1744 3056 : Write the data to the disk
1744 3057 :
1744 3058 30$: DECL CDRP$L_CMD_STS(R5) ; Set the command status
1747 3059 INCW CDRP$W_STATE(R5) ; Set state
174A 3060 MOVL CDRP$L_SHDW_LNK(R5), R3 ; Pick up the first IRP
174E 3061 BEQL 60$ ; No shadowing, skip
1750 3062 MOVL #1, CDRP$L_IOST1(R5) ; Set default success
1754 3063 40$: PUSHL IRP$L_SHDW_LNK(R3) ; Save pointer to next IRP
1758 3064 MOVQ R4, -(SP) ; Save registers
175B 3065 BSBW 70$ ; Do it
175D 3066 MOVQ (SP)+, R4 ; Restore registers
1760 3067 POPL R3 ; Restore pointer
```

```
EF 12 1763 3068 BNEQ 40$ ; and loop
05 1765 3069 50$: RSB ; None left, leave now
1766 3070 ;
1766 3071 ; Non-shadowed writes
1766 3072 ;
02F4 30 1766 3073 60$: BSBW DO_DISK ; Do the disk transfer
A3 AF 9F 1769 3074 PUSHAB 20$ ; Loop return
FCD6 31 176C 3075 BRW TST_END ; Check if finished
176F 3076 ;
176F 3077 ; Shadowed writes
176F 3078 ;
C2 A5 96 176F 3079 70$: INCB CDRP$B_SHDW_CNT(R5) ; Count the number of copies
02EC 30 1772 3080 80$: BSBW DO_DISK ; Do the disk transfer
0F 50 E9 1775 3081 BLBC R0,WRITE_ERR ; Error
C2 A5 97 1778 3082 90$: DECB CDRP$B_SHDW_CNT(R5) ; Drop the count
E8 12 1778 3083 BNEQ 50$ ; Not finished, go back for more
8F AF 9F 177D 3084 PUSHAB 20$ ; Loop return
50 DB A5 D0 1780 3085 MOVL CDRP$L_IOST1(R5),R0 ; Pick up composite status
FCBE 31 1784 3086 BRW TST_END ; Check if finished
1787 3087 ;
1787 3088 ; Shadowed write error recovery.
1787 3089 ;
1787 3090 WRITE_ERR:
04 E0 1787 3091 BBS #CDRP$V_ABORT,- ; Leave if abort
EC FA A5 1789 3092 CDRP$B_FLAGS(R5),90$
01F4 8F 50 B1 178C 3093 CMPW R0,#SS$_PARITY ; If not parity
08 13 1791 3094 BEQL 110$ ; then
2284 8F 3C 1793 3095 MOVZWL #SS$_SHACHASTA,- ; Update composite status
DB A5 1797 3096 CDRP$L_IOST1(R5)
DD 11 1799 3097 BRB 90$ ; Continue
1798 3098
51 3A A3 50 01 D0 1798 3099 110$: MOVL #1,R0 ; Set for success
F7 8F 78 179E 3100 ASHL #-9,IRP$L_IOST1+2(R3),R1 ; Number of blocks transfered
51 D6 17A4 3101 INCL R1 ; Skip this block
00A8 C3 51 C0 17A6 3102 ADDL R1,IRP$L_LBN(R3) ; Show it
51 51 02 78 17AB 3103 ASHL #2,R1,R1 ; Number of bytes of svapte
2C A3 51 C0 17AF 3104 ADDL R1,IRP$L_SVAPTE(R3) ; Update buffer address
51 51 07 78 17B3 3105 ASHL #7,R1,R1 ; Number of bytes "transfered"
32 A3 51 C2 17B7 3106 SUBL R1,IRP$L_BCNT(R3) ; Left to go
B5 12 17BB 3107 BNEQ 80$ ; Redo the IO
B9 11 17BD 3108 BRB 90$ ; Nothing left, fake success
17BF 3109
17BF 3110 .DISABLE LSB
```

```
17BF 3112 .SBTTL SETUP - Setup the CDRP for transfers
17BF 3113
17BF 3114 :++
17BF 3115 :
17BF 3116 : SETUP - Set up the all the entries for both disk and CI transfers
17BF 3117 :
17BF 3118 : Inputs:
17BF 3119 : R0 = Function code
17BF 3120 : R2 --> MSCP packet
17BF 3121 : R4 --> UQB
17BF 3122 : R5 --> CDRP
17BF 3123 :
17BF 3124 : Output:
17BF 3125 : R4 --> PDT
17BF 3126 : R5 --> CDRP
17BF 3127 :
17BF 3128 :--
17BF 3129 :
52 B0 A5 D0 17BF 3130 SETUP: MOVL CDRP$L MSGBUF(R5),R2 ; Refresh R2 => MSCP message.
C4 A5 54 D0 17C3 3131 MOVL R4,CDRP$L_UQB(R5) ; Save the UQB pointer
78 A5 54 D0 17C7 3132 MOVL R4,CDRP$L_MST_UQB(R5) ; Save master UQB pointer
64 B4 E8 A5 0E 17CB 3133 INSQUE CDRP$L_UQB_FL(R5),- ; Insert this packet on unit que
17D0 3134 2UQB$L_CDRP_BL(R4)
50 A4 B6 17D0 3135 INCW UQB$L_NUM_QUE(R4) ; Bump number of packet queue to unit
50 A4 B1 17D3 3136 CMPW UQB$L_NUM_QUE(R4),- ; Is this number <= max?
52 A4 17D6 3137 UQB$L_MAX_QUE(R4)
05 1B 17D8 3138 BLEQU 10$ ; Yes, skip
50 A4 B0 17DA 3139 MOVW UQB$L_NUM_QUE(R4),- ; No, set new max
52 A4 17DD 3140 UQB$L_MAX_QUE(R4)
D6 A5 B4 17DF 3141 10$: CLRW CDRP$L_STATE(R5) ; Set state
4000 8F B3 17E2 3142 BITW #10MSCP$L_MD_COMP,-
0A A2 17E6 3143 MSCP$L_MODIFIER(R2)
05 13 17E8 3144 BEQL 20$ ; If COMPARE modifier
50 4000 8F A8 17EA 3145 BISW #10SM_DATACHECK,R0 ; ... set DATACHECK modifier
0100 8F B3 17EF 3146 20$: BITW #10MSCP$L_MD_SEREC,-
0A A2 17F3 3147 MSCP$L_MODIFIER(R2)
05 13 17F5 3148 BEQL 30$ ; If SUPPRESS ERROR CORRECT modifier
50 8000 8F A8 17F7 3149 BISW #10SM_INHRETRY,R0 ; ... set INHIBIT RETRY modifier
C0 A5 50 B0 17FC 3150 30$: MOVW R0,CDRP$L_FUNC(R5) ; Set completed code in packet
51 0C A2 D0 1800 3151 MOVL MSCP$L_BYTE_CNT(R2),R1 ; Pick up size of request
5A 13 1804 3152 BEQL 60$ ; Special case zero length
30 A4 D0 1806 3153 MOVL UQB$L_UCB(R4),- ; Transfer UCB address
BC A5 1809 3154 CDRP$L_UCB(R5) ; to UCB addr in IRP
E0 A5 D4 180B 3155 CLRL CDRP$L_ABCNT(R5) ; Clear accumulated byte count
E4 A5 51 D0 180E 3156 MOVL R1,CDRP$L_OBCNT(R5) ; Set original byte count
53 51 D0 1812 3157 MOVL R1,R3 ; Copy to another register
53 53 D7 1815 3158 DECL R3 ; Round byte count down
53 53 F7 8F 78 1817 3159 ASHL #-VASS_BYTE,R3,R3 ; Convert to block count
50 50 53 D0 181C 3160 MOVL R3,R0 ; Block count
0000007F 8F 50 D1 181F 3161 CMPL R0,#127 ; If less than, skip on
04 1B 1826 3162 BLEQU 40$
50 7F 8F 9A 1828 3163 MOVZBL #127,R0 ; Else make 127
EA83 CF40 D6 182C 3164 40$: INCL BLK_COUNT+4[R0] ; Keep statistics
50 1C A2 D0 1831 3165 MOVL MSCP$L_LBN(R2),R0 ; Transfer the logical block number
53 50 C0 1835 3166 ADDL R0,R3 ; Calculate ending block number
48 A4 53 D1 1838 3167 CMPL R3,UQB$L_MAXBLOCK(R4) ; Legal block number
18 1A 183C 3168 BGTRU 50$ ; If GTRU no
```

```
4C A4 50 C1 183E 3169 ADDL3 R0,UQBSL START_LBN(R4),-
      48 A5 1842 3170 CDRP$L [BN(R5) ; Set 'real' starting LBN
      20 93 1844 3171 BITB #1@CDRP$V NOHSTBUF,- ; Is this a no host buffer IO?
      FA A5 1846 3172 CDRP$B_Flags(R5)
      1F 13 1848 3173 BEQL 70$ ; No, continue with setup
B4 A5 51 D0 184A 3174 MOVL R1,CDRP$L BUFLen(R5) ; Yes, save length
      10 A2 7D 184E 3175 MSCP$B BUFFER(R2) - ; X'fer SVAPTE and BOFF
      CC A5 1851 3176 CDRP$L_SVAPTE(R5)
      00CD 31 1853 3177 BRW 130$ ; And finish
      1856 3178
      1856 3179 ; Out of line error routine
      1856 3180
      1856 3181
      1856 3182
      1856 3183
50 0C01 8E D5 1856 3184 50$: TSTL (SP)+ ; Get rid of return address
      8F 3C 1858 3184 MOVZWL #MSCP$K ST_ICMD -
      F274 31 185D 3185 !<MSCP$L_BYTE_CNT@8>,R0 ; Set error status
      EA4C CF D6 1860 3186 BRW SEND_PKT ; Leave in disgrace
      8E D5 1864 3187 60$: INCL BLK_COUNT ; Count zero block transfer
      F269 31 1866 3189 TSTL (SP)+ ; Get rid of return address
      1869 3190 BRW SEND_SUCCESS
      1869 3191
      1869 3192 ; Not a passthru so grab an internal buffer, and map it
      1869 3193
      1869 3194
      1869 3195
34 A5 10 A2 DE 1869 3196 70$: MOVAL MSCP$B BUFFER(R2), - ; Calculate offset of remote buf hdl
      38 A5 D4 186E 3197 CDRP$L_RBUFH_AD(R5)
      E7E6 CF 51 D1 1871 3198 CLRL CDRP$L_RBOFF(R5) ; Clear remote offset
      05 1B 1876 3199 CMPL R1,FRACTION ; Greater than the fraction?
      51 E7E0 CF D0 1878 3200 BLEQU 80$ ; No, continue
      51 01FF C1 9E 187D 3201 MOVL FRACTION,R1 ; Yes, use the fraction
51 000001FF 8F CA 1882 3202 80$: MOVAB 511(R1),R1 ; Round up
      21 13 1889 3203 BICL2 #511,R1
      53 E7C1 CF DE 188B 3204 BEQL 90$ ; Less than a page
      00000000'9F 16 1892 3205 MOVAL BUFF_HEAD,R3 ; Set up
      42 50 E8 1898 3206 PUSHL R1 ; Save request size
      51 8ED0 189B 3207 JSB @#EXES$ALLOCATE ; Grab the block
      E7B5 CF 51 D1 189E 3208 BLBS R0,110$ ; Space
      07 1B 18A3 3209 POPL R1 ; Retrieve request size
      51 51 FF 8F 78 18A5 3210 CMPL R1,SMALL ; Less than smallest fragment?
      D1 11 18AA 3211 BLEQU 90$ ; Yes, wait for pool
      18AC 3212 ASHL #-1,R1,R1 ; Try for half
      18AC 3213 BRB 80$ ; And go again
      18AC 3214
      18 A5 8ED0 18AC 3215 90$: POPL CDRP$L_SAVD_RTN(R5) ; Store return address
      65 0E 18B0 3216 INSQUE CDRP$L_FQFLT(R5),- ; Insert on the waiting que
      E7C7 DF 18B2 3217 @MEM_WAIT+4
      E7A7 CF B6 18B5 3218 INCW NUM_QUE ; Bump memory wait counter
      E7A2 CF E7A3 CF B1 18B9 3219 CMPW NUM_QUE,MAX_QUE ; Maximize with
      07 1B 18C0 3220 BLEQU 100$ ; current
      E799 CF E79A CF B0 18C2 3221 MCVW NUM_QUE,MAX_QUE ; maximum
      18C9 3222 100$: WFINT ; Stall until resource avail
      18CC 3223 ; (Will be activated at CLEAN_UP)
      E790 CF B7 18CC 3224 DECW NUM_QUE ; Decr memory wait counter
      18 A5 DD 18D0 3225 PUSHL CDRP$L_SAVD_RTN(R5) ; Set return address
```

```
51 E4 A5 D0 18D3 3226 MOVL CDRP$L_OBCNT(R5),R1 ; Pick up needed length
      A4 11 18D7 3227 BRB 80$ ; Try again
      18D9 3228
      18D9 3229 105$: BUG_CHECK MSCPSERV,FATAL ; Allocated region out of bounds
      18DD 3230
      SE 04 C0 18DD 3231 110$: ADDL2 #4,SP ; Flush request size
      18E0 3232 : ??? BUG TRAP
EBEB CF 52 D1 18E0 3233 CMPL R2,BUFF_ADDR ; Does address precede allocation region
      F2 1F 18E5 3234 BLSSU 105$ ; Branch if yes
50 51 52 C1 18E7 3235 ADDL3 R2,R1,R0 ; End of region+1
EBE4 CF 50 D1 18EB 3236 CMPL R0,BUFF_END ; Is it beyond+1
      E7 1A 18F0 3237 BGTRU 105$
      18F2 3238 : ??? END BUG TRAP
      D6 A5 66 18F2 3239 INCW CDRP$W_STATE(R5) ; Bump the state
      B4 A5 51 7D 18F5 3240 MOVQ R1,CDRP$L_BUFLN(R5) ; Save length and address
      E4 A5 51 D1 18F9 3241 CMPL R1,CDRP$L_OBCNT(R5) ; If the length has been rounded
      04 15 18FD 3242 BLEQ 120$ ; then
      51 E4 A5 D0 18FF 3243 MOVL CDRP$L_OBCNT(R5),R1 ; Restore the original byte count
      3C A5 51 D0 1903 3244 120$: MOVL R1,CDRP$L_XCT_LEN(R5) ; Set length in place for S/R DATA
      FA A5 02 88 1907 3245 BISB #1@CDRP$V_BUFFER, -
      190B 3246 CDRP$B_FLAGS(R5) ; Set buffer flag
DO A5 52 FE00 8F AB 190B 3247 BICW3 #XFE00,R2, -
      1912 3248 CDRP$W_BOFF(R5) ; Set the byte offset
      15 09 EF 1912 3249 EXTZV #VASV_VPN,#VASS_VPN,-
      52 52 1915 3250 R2,R2 ; Virtual page number
50 00000000'9F D0 1917 3251 MOVL @#MMG$GSL_SPTBASE, R0 ; Get address of sys page table
CC A5 6042 DE 191E 3252 MOVAL (R0)[R2],CDRP$L_SVAPTE(R5) ; Set in CDRP
      D2 A5 51 D0 1923 3253 130$: MOVL R1,CDRP$L_BCNT(R5) ; Set the length = byte count
      1927 3254
      1927 3255 ; Almost done, now check for shadowing and allocate multiple IRP's
      1927 3256
      4C A5 D4 1927 3257 CLRL CDRP$L_SHDW_LNK(R5) ; Clear out shadow irp link
51 A0 A5 DE 192A 3258 MOVAL CDRP$L_IOQF(R5),R1 ; Cover IRP portion w/R1
      0200 8F B3 192E 3259 BITW #1@MSCP$V_UF_SSMST,- ; If shadow unit
      12 A4 1932 3260 UQBSW UNT_F[GS(R4)
      09 12 1934 3261 BNEQ SETUP_SHADOW ; then do shadow specific
      1936 3262
      1936 3263 SETUP_OUT:
54 60 A5 D0 1936 3264 MOVL CDRP$L_CDTX(R5),R4 ; Pick up the CDT address
54 10 A4 D0 193A 3265 MOVL CDT$L_PDT(R4),R4 ; Set up PDT address
      05 193E 3266 RSB
      193F 3267
      193F 3268
      193F 3269 ; Shadowing enabled, grab extra IRP's for the multiple operations.
      193F 3270
      193F 3271
      193F 3272 SETUP_SHADOW:
      22 91 193F 3273 CMPB #MSCP$K_OP_WRITE,-
      FB A5 1941 3274 CDRP$B_OPCODE(R5) ; Leave if not write
      03 13 1943 3275 BEQL SETUP_SHADOW_WRITE
      00D7 31 1945 3276 BRW SETUP_SHADOW_READ
      1948 3277
      1948 3278 ; Shadow writes. Check for CIP. If so may need to hold write. If not
      1948 3279 ; spawn extra IRP's for the multiple write operations.
      1948 3280
      1948 3281 SETUP_SHADOW_WRITE:
      01 B3 1948 3282 BITW #UQBSM_CIP,-
```

```

      2B A4      194A 3283      UQBSB_FLAGS(R4)      : If CIP not in progress
      45 13      194C 3284      20$      : just continue on
      58 A5      04 194E 3285      BEQL      : Clear a flag
      50 48 A5      D0 1951 3286      CLRL      : Pick up starting block
      51 E4 A5      F7 1955 3287      MOVL      : Get block range
      51 50      C0 1958 3288      ASHL      : Now ending block
      5C A4      50 D1 195E 3289      ADDL      : Check
      2F 1E 1962 3290      CMPL      : Outside (over)
      58 A5      D6 1964 3291      BGEQU      : Set a flag
      58 A4      51 D1 1967 3292      INCL      : Check block range
      26 1B 1968 3293      CMPL      : Outside (under)
      08 88 196D 3294      BLEQU      : Set a write pending
      2B A4      196F 3295      10$: BISB      : UQBSM_CMDPEND, -
      18 A5      8ED0 1971 3296      UQBSB_FLAGS(R4)
      F956 30 1975 3297      POPL      : CDRPSL_SAVD RTN(R5)
      50 E8 A5      OF 1978 3298      BSBW      : Hold return address
      50 A4      B7 197C 3299      REMQUE     : Get rid of buffer
      74 B4      60 OE 197F 3300      DECU      : Remove from executing queue
      D6 A5      06 B0 1983 3301      DECW      : (make count agree)
      18 A5      DD 198A 3302      INSQUE     : Insert on special queue
      50 0B      B0 198D 3303      MOVW      : Set state as special queued
      FE2C 31 1990 3304      WFIN      : Block and wait for copy to complete
      1993 3305      PUSHL      : Restore
      1993 3306      MOVW      : Set for write
      1993 3307      BRW      : Now try again
      1993 3308      : Now either we had no copy-in-progress on the shadow unit or we did but
      1993 3309      : the IO was outside the fence. If the latter, then we write to all
      1993 3310      : units if under the fence; and to all units except the CIP unit if over.
      01 26 A4      B1 1993 3311      20$: CMPW      : UQBSW_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
      7E 26 A4      3C 199C 3314      :
      51 A0 A5      DE 19A0 3315      25$: MOVZWL     : UQBSW_SHDW_STS(R4),-(SP) : Set up unit counter
      53 64      D0 19A4 3316      MOVAL      : CDRPSL_IOQFL(R5),R1 : Cover IRP portion w/R1
      01 93 19A7 3317      30$: MOVL      : UQBSL_SHDW_FL(R4),R3 : Pick up possible shadow UQB's
      2B A3      19A9 3318      BITB      : #UQBSM_CIP,- : Check for copy in progress
      09 13 19AB 3319      UQBSB_FLAGS(R3)
      05 58 A5      E8 19AD 3320      BEQL      : 40$      : None, all is okay
      54 53      D0 19B1 3321      BLBS      : CDRPSL_FR4X(R5),40$ : CIP, but IO is under the fence
      60 11 19B4 3322      MOVL      : R3,R4 : Oops, try another
      19B6 3323      BRB      : 50$      : Check for end of line
      19B6 3324      :
      19B6 3325      : Grab an IRP for each active shadow set member
      19B6 3326      :
      51 00C4 8F      DD 19B6 3327      40$: PUSHL      : R1 : Save R1
      EC94 30 19B8 3328      MOVZWL     : #IRPSC_LENGTH,R1 : Size for an IRP
      51 8ED0 19C0 3329      BSBW      : ALLOMEM : Grab one
      03 50      E8 19C3 3330      POPL      : R1 : Restore
      0089 31 19C6 3331      BLBS      : R0,45$ : Success
      19C9 3332      BRW      : SETUP_ERR : No space left
      19C9 3333      :
      19C9 3334      : Fill in the IRP with the essential transfer fields
      19C9 3335      :
      08 A2      000A00D0 8F      D0 19C9 3336      45$: MOVL      : #<DYN$C_IRP@16>!- : Set size and DYN code
      01 88 19CA 3337      : <<IRPSC_LENGTH+15>&-16>,- :
      19CA 3338      : IRP$W_SIZE(R2) :
      19D1 3339      BISB      : #1@CDRPSV_SLAVE,- : Set the slave IRP flag
```

```
00B4 FA A5 19D3 3340 CDRP$B_FLAGS(R5)
24 C2 55 D0 19D5 3341 MOVL R5,IRP$C_MAINCDRP(R2) ; Set pointer to CDRP
00AC A2 53 D0 19DA 3342 MOVL R3,IRP$L_UQB(R2) ; Set pointer to UQB
C1 C2 D4 19DE 3343 CLRL IRP$L_SHDW_LNK(R2) ; Clear shadowing pointer
CA A5 D0 19E2 3344 MOVL R2,IRP$L_SHDW_LNK(R1) ; Set back pointer
2A A5 7D 19E7 3345 MOVQ CDRP$W_STS(R5),- ; Copy transfer parameters
D2 A5 D0 19EA 3346 IRP$W_STS(R2)
32 A2 D0 19EC 3347 MOVL CDRP$L_BCNT(R5),- ; Ditto
48 A5 D0 19EF 3348 IRP$L_BCNT(R2)
00A8 C2 D0 19F1 3349 MOVL CDRP$L_LBN(R5),- ; Ditto
54 53 D0 19F4 3350 IRP$L_LBN(R2)
C0 A5 D0 19F7 3351 MOVL R3,R4 ; Copy UQB pointer
20 A2 D0 19FA 3352 MOVL CDRP$W_FUNC(R5),- ; Ditto
30 A4 D0 19FD 3353 IRP$W_FUNC(R2)
1C A2 D0 19FF 3354 MOVL UQB$L_OCB(R4),- ; Ditto
53 52 D0 1A02 3355 IRP$L_UCB(R2)
F8 A5 D0 1A04 3356 MOVL R2,R3 ; Cover IRP
58 A2 D0 1A07 3357 MOVL CDRP$W_SLOT(R5),- ; Copy flags and unit
51 52 D0 1A0A 3358 IRP$W_SLOT(R2)
36 A2 B4 1A0C 3359 MOVL R2,R1 ; Copy IRP pointer
01 88 1A0F 3360 CLRW IPP$W_STATE(R2) ; Clear out indicator
5A A1 88 1A12 3361 BISB #1@CDRP$V_SLAVE,- ; Set the slave IRP flag
6E D7 1A14 3362 IRP$B_FLAGS(R1)
8A 1A 1A16 3363 50$: DECL (SP) ; Count down
8E D5 1A18 3364 BGTRU 30$ ; More to go
FF17 31 1A1A 3365 TSTL (SP)+ ; Clean stack
1A1C 3366 BRW SETUP_OUT ; End of the line
1A1F 3367 ;
1A1F 3368 ; Shadow reads. Pick the next unit to read from. Advance the ring pointer.
1A1F 3369 ;
1A1F 3370 SETUP_SHADOW_READ:
51 64 D0 1A1F 3371 MOVL UQB$L_SHDW_FL(R4),R1 ; Pick up first unit
74 A5 51 D0 1A22 3372 MOVL R1,CDRP$L_STRT_UQB(R5) ; Set end condition
15 10 1A26 3373 BSB0 SELECT_FIRST_UNIT ; Pick up first useable reader
06 13 1A28 3374 BEQL 999$ ; ERROR
61 D0 1A2A 3375 MOVL UQB$L_SHDW_FL(R1),- ; Update master UQB pointer
64 1A2C 3376 UQB$L_SHDW_FL(R4)
FF06 31 1A2D 3377 BRW SETUP_OUT ; Leave
1A30 3378
1A30 3379 999$: BUG_CHECK MSCPSERV,FATAL ; ERROR
1A34 3380 ;
1A34 3381 ; SELECT_NEXT_UNIT - Loop thru the shadow members
1A34 3382 ;
1A34 3383 ; R1 --> Base UQB to start from
1A34 3384 ; **** USE CIP IF BELOW FENCE?
1A34 3385 ;
1A34 3386 ; .ENABLE LSB
1A34 3387
1A34 3388 SELECT_NEXT_UNIT:
51 61 D0 1A34 3389 10$: MOVL UQB$L_SHDW_FL(R1),R1 ; Pick up next candidate
74 A5 51 D1 1A37 3390 CMPL R1,CDRP$L_STRT_UQB(R5) ; Have we gone around the ring
14 13 1A38 3391 BEQL 20$ ; Yes, just error out
1A3D 3392
1A3D 3393 SELECT_FIRST_UNIT:
2B 01 93 1A3D 3394 BITB #UQB$M_CIP,- ; Check for possible copy in progress
A1 A1 1A3F 3395 UQB$B_FLAGS(R1)
F1 12 1A41 3396 BNEQ 10$ ; Yes, disqualify
```

- MSCP Server - Emulator
- SETUP - Setup the CDRP for transfers

```
16-SEP-1984 02:00:  VAX/VMS Macro V04-00
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR:2
```

Page 79  
(1)

|    |      |    |    |      |      |                  |                        |                                     |
|----|------|----|----|------|------|------------------|------------------------|-------------------------------------|
|    | OE   | 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,CDRPSL_UQB(R5)      | ; Transfer new UQB pointer          |
|    | 30   | A1 | D0 | 1A4C | 3400 | MOVL             | UQB\$SL_UCB(R1) -      | ; Transfer UCB pointer              |
|    | BC   | A5 |    | 1A4F | 3401 |                  | CDRPSL_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 |
| 5C | 00AA | 8F | 3C | 1A55 | 3410 | MOVZWL           | #<5*MSCPSK_ST_SBCOD! - |                                     |
|    |      |    |    | 1A5A | 3411 |                  | MSCPSK_ST_CNTLR>,R0    | ; Set error status ;*** TEMP        |
|    | F077 |    | 31 | 1A5A | 3412 | BRW              | SEND_PKT               | ; Leave in disgrace                 |

[illegible]

```
1A5D 3414 .SBTTL DO_DISK & BACK - DISK IO utility routines
1A5D 3415
1A5D 3416 :++
1A5D 3417 :
1A5D 3418 : DO_DISK - Routine to set up IRP for disk I/O
1A5D 3419 :
1A5D 3420 : Inputs:
1A5D 3421 :
1A5D 3422 : R3 - Current IRP address
1A5D 3423 : R4 - PDI address
1A5D 3424 : R5 - (Master) CDRP address
1A5D 3425 :
1A5D 3426 :--
1A5D 3427
1A5D 3428 .ENABLE LSB
1A5D 3429
1A5D 3430 DO_DISK:
1A5D 3431 MOVAB CDRP$L_IOQFL(R5),R3 ; Cover IRP portion
1A61 3432
1A61 3433 DO_DISKA:
1A61 3434 MOVL CDRP$L_UQB(R5),R1 ; Pick up UQB ptr
1A65 3435 TSTW UQB$W_STATUS(R1) ; Has it gone AVAIL while we waited?
1A68 3436 BEQL DO_DISKX ; No, finish IO
1A6A 3437 MOVZWL #$$$_MEDOFL,R0 ; Yes, give error code
1A6F 3438 RSB ; and leave
1A70 3439
1A70 3440 INV_FUNC:
1A70 3441 BUG_CHECK MSCPSERV,FATAL ; Invalid function code
1A74 3442
1A74 3443 DO_DISKX:
1A74 3444 POPL IRP$L_FPCX(R3) ; Where to return after IOPOST
1A79 3445 MOVL R4,IRP$L_FR4X(R3) ; Save R4
1A7E 3446 ASSUME CDRP$V_SAVE EQ 0
1A7E 3447 BLBS IRP$B_FLAGS(R3),5$ ; If this is a slave IRP, then
1A82 3448 MOVL CDRP$C_XCT_LEN(R5),- ; Save length in a safe place
1A85 3449 CDRP$C_XCT_LENX(R5)
1A87 3450 5$: MOVAB W^BACK,IRP$C_PID(R3) ; Addr to which IOPOST will return IRP
1A8D 3451 CLRL CDRP$L_MSG_BUF(R5) ; Mark it not in use
1A90 3452 ; ??? begin bug trap
1A90 3453 EXTZV #IOSV_FCODE, - ; Isolate function code
1A96 3454 #IOSS_FCODE, -
1A96 3455 IRP$W_FUNC(R3),R0
1A96 3456 CMPW R0,#IOS_READPBLK ; Read?
1A99 3457 BEQL 7$ ; Branch if read
1A98 3458 CMPW R0,#IOS_WRITEPBLK ; Write?
1A9E 3459 BEQL 7$ ; Branch if write
1AA0 3460 PUSHL R5 ; Save CDRP address
1AA2 3461 MOVL IRP$L_UCB(R3),R5 ; Get UCB address
1AA6 3462 MOVL UCB$L_DDT(R5),R5 ; DDT address
1AAB 3463 BBC R0,@DDT$L_FDT(R5),INV_FUNC ; Branch if function code is invalid
1AB0 3464 POPL R5 ; Restore CDRP address
1AB3 3465 CMPW R0,#IOS_WRITECHECK ; Writecheck?
1AB6 3466 BNEQ 9$
1AB8 3467 7$: CMPL IRP$L_BCNT(R3),CDRP$L_BUFLN(R5)
1ABD 3468 BGTRU 8$ ; Tilt!
1ABF 3469 8$: SUBL3 @MMG$GL_SPTBASE, - ; Get address of buffer page table
1AC8 3470 IRP$L_SVAPTE(R3),R0
```

```
50 50 07 78 1AC8 3471 ASHL #9-2,R0,R0 ; Convert to address in segment
00 50 1F E2 1ACC 3472 BBSS #31,R0,R1 ; Set 80000000 bit
50 D0 A5 A0 1AD0 3473 71$: ADDW2 CDRPSW_BOFF(R5),R0
BB A5 50 D1 1AD4 3474 CMPL R0,CDRPSL_BUFADR(R5) ; Is buffer address consistent?
12 12 1AD8 3475 BNEQ 8$
E9F1 CF 50 D1 1ADA 3476 CMPL R0,BUFF_ADDR
08 1F 1ADF 3477 BLSSU 8$ ; Start of buffer out of bounds
50 32 A3 C0 1AE1 3478 ADDL2 IRPSL_BCNT(R3),R0
E9EA CF 50 D1 1AE5 3479 CMPL R0,BUFF_END ; Is buffer contained within range?
04 1B 1AEA 3480 BLEQU 9$ ; Branch if yes
1AEC 3481 8$: BUG_CHECK MSCPSERV,FATAL ; Invalid disk buffer address
1AF0 3482 9$:
1AF0 3483 ; ??? end bug trap
50 00A8 C3 D0 1AF0 3485 MOVL IRPSL_LBN(R3),R0 ; Pick up starting logical block
55 24 A3 D0 1AF5 3486 MOVL IRPSL_UB(R3),R5 ; Pick up UQB ptr
54 A5 B6 1AF9 3487 INCW UQBSW_NUM_OPS(R5) ; Bump number of packets queued to unit
54 A5 B1 1AFC 3488 CMPW UQBSW_NUM_OPS(R5),- ; Greater than current max?
56 A5 1AFF 3489 UQBSW_MAX_OPS(R5)
05 1B 1B01 3490 BLEQU 10$ ; No, skip on
54 A5 B0 1B03 3491 MOVW UQBSW_NUM_OPS(R5),- ; Yes, replace current max
56 A5 1B06 3492 UQBSW_MAX_OPS(R5)
55 1C A3 D0 1B08 3493 10$: MOVL IRPSL_UCB(R3),R5 ; and now UCB
00000000'9F D5 1B0C 3494 TSTL @#PMSSGL_IOPFMPDB ; I/O performance monitoring enabled ?
23 12 1B12 3495 BNEQ 30$ ; BR if yes
00000000'9F D0 1B14 3496 DSBINT #15 ; Protect I/O sequence # manipulation
50 A3 1B20 3497 MOVL @#PMSSGL_IOPFMSEQ,- ; Insert I/O sequence # into IRP
00000000'9F D6 1B22 3498 IRPSL_SEQNUM(R3)
1B28 3500 INCL @#PMSSGL_IOPFMSEQ ; and bump global counter
00000000'9F 16 1B2B 3501 20$: JSB @#IOC$CVTLOGPHY ; Restore previous IPL
00000000'9F 17 1B31 3502 JMP @#EXES$INSIOQ ; Convert logical to physical block
1B37 3503 ; Queue the IRP to the driver
00000000'9F 16 1B37 3504 30$: JSB @#PMSS$START_RQ ; Gather Start I/O Request statistics
EC 11 1B3D 3505 BRB 20$
1B3F 3506
1B3F 3507 .DISABLE LSB
1B3F 3508
1B3F 3509 ; Routine will RSB back to our caller's caller which is in the PORT
1B3F 3510 ; DRIVER. Thread is now suspended and will resume when IOPOST calls us
1B3F 3511 ; at entry BACK.
1B3F 3512
1B3F 3513 R5 --> IRP
1B3F 3514
1B3F 3515 BACK: DSBINT #IPL$ SCS ; Set IPL to FORK
0100 8F AA 1B45 3516 BICW #IRPSM_PHYSIO,- ; Clear the physical IO flag
2A A5 1B49 3517 IRPSW_STS(R5) ; only needed for ONLINE
00000000'9F D5 1B4B 3518 TSTL @#PMSSGL_IOPFMPDB ; I/O performance monitoring enabled ?
06 13 1B51 3519 BEQL 10$ ; BR if no
00000000'9F 16 1B53 3520 JSB @#PMSS$END_RQ ; Gather End I/O Request statistics
53 55 D0 1B59 3521 10$: MOVL R5,R3 ; R3-->head of IRP
54 00B8 C3 D0 1B5C 3522 MOVL IRPSL_FR4X(R3),R4 ; Restore PDT
51 24 A3 D0 1B61 3523 MOVL IRPSL_UB(R3),R1 ; Pick up UQB pointer
54 A1 B7 1B65 3524 DECW UQBSW_NUM_OPS(R1) ; Drop count of operations
51 1C A3 D0 1B68 3525 MOVL IRPSL_UCB(R3),R1 ; Pick up UCB pointer
6A A1 B7 1B6C 3526 DECW UCB$W_QLEN(R1) ; Drop queue length counter
1B6F 3527 ASSUME CDRPSV_SLAVE EQ 0
```

```
1888 3555 .SBTTL UTILITY ROUTINES
1888 3556
1888 3557 .....
1888 3558 .....
1888 3559 .....
1888 3560 SEARCH_QUEUE
1888 3561 .....
1888 3562 .....
1888 3563 .....
1888 3564
1888 3565 Input:
1888 3566 R3 = Address of HQB possible having a CDRP with that command
1888 3567 R4 = Unit number
1888 3568 R5 = Address of CDRP containing command reference # to search for
1888 3569
1888 3570 Output:
1888 3571 R0 = Status
1888 3572 Low bit clear if not found
1888 3573 Low bit set if found
1888 3574 R1 --> CDRP$UQB_FL of the CDRP found
1888 3575 :-
1888 3576
1888 3577 SEARCH_QUEUE:
54 E4D1 CF 50 D4 1888 3578 CLRL R0 ; Set for failure
56 B0 A5 D0 1888 3579 MOVL UNIT_VEC[R4],R4 ; Grab UQB
56 0C A6 D0 1888 3580 BEQL 40$ ; None there
56 50 A4 B5 1888 3581 PUSHL R6 ; Grab a register
56 09 13 1888 3582 MOVL CDRP$MSGBUF[R5],R6 ; Pick up buffer
51 60 A4 9E 1888 3583 MOVL MSCP$OUT_REF[R6],R6 ; Id to search for
51 40 10 1888 3584 TSTW UQB$NUM_QUEUE(R4) ; Any entries on normal que?
51 39 50 E8 1888 3585 BEQL 10$ ; No, leave now
51 7C A4 B5 1888 3586 MOVAB UQB$_CDRP_FL(R4),R1 ; Set to head of que
51 09 13 1888 3587 BSBB 50$ ; Look for the packet
51 68 A4 9E 1888 3588 BLBS R0,30$ ; Found it
51 32 10 1888 3589 10$: TSTW UQB$NUM_BLK(R4) ; Any entries on blocked que?
51 2B 50 E8 1888 3590 BEQL 20$ ; No, leave now
51 70 A4 9E 1888 3591 MOVAB UQB$_BLOCKQ(R4),R1 ; Set to head of que
51 29 10 1888 3592 BSBB 50$ ; Look for the packet
51 22 50 E8 1888 3593 BLBS R0,30$ ; Found it
51 00 00 E1 1888 3594 20$: MOVAB UQB$_SHQ(R4),R1 ; Set to head of special hold que
51 1D 2B A4 1888 3595 BSBB 50$ ; Look for the packet
51 00 00 E1 1888 3596 BLBS R0,30$ ; Found it
51 00 00 E1 1888 3597 BBC #UQB$V_CIP,- ; If unit is part of shadow copy
51 00 00 E1 1888 3598 MOVL UQB$B_FLAGS(R4),30$ ; then
51 00 00 E1 1888 3599 MOVL UQB$_CPY_CDRP(R4),R1 ; Pick up pointer to its CDRP
51 00 00 E1 1888 3600 MOVAB CDRP$_UQB_FL(R1),R1 ; Adjust pointer
51 00 00 E1 1888 3601 CMPL HQB$_CDT(R3),- ; Is this CDRP for this HQB?
51 00 00 E1 1888 3602 CDRP$_CDTX-CDRP$_UQB_FL(R1) ; No
51 00 00 E1 1888 3603 BNEQ 30$ ; No
51 00 00 E1 1888 3604 ASSUME MSCP$_CMD_REF EQ 0
51 00 00 E1 1888 3605 TSTL CDRP$_MSGBUF- ; Is there a message buffer?
51 00 00 E1 1888 3606 CDRP$_UQB_FL(R1) ; No, so can never match
51 00 00 E1 1888 3607 BEQL 30$ ; No, so can never match
51 00 00 E1 1888 3608 CMPL @CDRP$_MSGBUF- ; Match com ref no in host chain
51 00 00 E1 1888 3609 CDRP$_UQB_FL(R1),R6
51 00 00 E1 1888 3610
51 00 00 E1 1888 3611
```

|    |    |      |      |      |      |        |                      |  |                                  |
|----|----|------|------|------|------|--------|----------------------|--|----------------------------------|
|    | 02 | 12   | 1C0F | 3612 |      | BNEQ   | 308                  |  | : No, leave                      |
|    | 50 | D6   | 1C11 | 3613 |      | INCL   | R0                   |  | : Set success                    |
|    | 56 | BED0 | 1C13 | 3614 | 308: | POPL   | R6                   |  |                                  |
|    |    | 05   | 1C16 | 3615 | 408: | RSB    |                      |  |                                  |
|    |    |      | 1C17 | 3616 |      |        |                      |  |                                  |
|    | 50 | D4   | 1C17 | 3617 | 508: | CLRL   | R0                   |  | : Set for failure                |
| S2 | 51 | D0   | 1C19 | 3618 |      | MOVL   | R1,R2                |  | : Store pointer                  |
| S1 | 61 | D0   | 1C1C | 3619 | 608: | MOVL   | (R1),R1              |  | : No, try next entry             |
| S2 | 51 | 01   | 1C1F | 3620 |      | CMPL   | R1,R2                |  | : End of que?                    |
|    | 14 | 13   | 1C22 | 3621 |      | BEQL   | 708                  |  | : Yes                            |
|    | 14 | A3   | D1   | 1C24 |      | CMPL   | HQB\$ (DT(R3),-      |  | : Is this CDRP for this HQB?     |
|    | 60 | A5   | 1C27 | 3623 |      |        | CDRPSL_CDTX(R5)      |  |                                  |
|    |    | F1   | 1C29 | 3624 |      | BNEQ   | 608                  |  | : No, flink on                   |
|    |    |      | 1C2B | 3625 |      | ASSUME | MSCPSL_CMD_REF EQ 0  |  |                                  |
|    | C8 | A1   | D5   | 1C2B |      | TSTL   | CDRPSL_MSGBUFX-      |  | : Is there a message buffer?     |
|    |    |      | 1C2E | 3627 |      |        | CDRPSL_UQB_FL(R1)    |  |                                  |
|    |    | EC   | 13   | 1C2E |      | BEQL   | 608                  |  | : No, so can never match         |
| S6 | C8 | B1   | D1   | 1C30 |      | CMPL   | @CDRPSL_MSGBUFX-     |  |                                  |
|    |    |      |      | 1C34 |      |        | CDRPSL_UQB_FL(R1),R6 |  | : Match com ref no in host chain |
|    |    | 6    | 12   | 1C34 |      | BNEQ   | 608                  |  | : No, loop                       |
|    |    | 50   | D6   | 1C36 |      | INCL   | R0                   |  | : Set success                    |
|    |    | 05   | 1C38 | 3633 | 708: | RSB    |                      |  |                                  |

## .SBTTL DATA STRUCTURE ALLOCATION

## GET\_STRUCT

Input:

NONE

Output:

R0 = Status

Low bit clear if none available

Low bit set if found

R5 --&gt; CDRP found

## GET\_STRUCT:

REMOUE @CDRP\_LIST,R5

: Pick up a CDRP

BVS 308

: None left

DECM NUM\_PKT

: Drop number of free packets

MOVL S\*SSS\_NORMAL,R0

: Set success status

RSB

208: MOVZWL #SSS\_INSMEM,R0

: ERROR

POPR #M&lt;R1,R2,R3,R4&gt;

: Restore volatile registers

RSB

: Leave

308: PUSHR #M&lt;R1,R2,R3,R4&gt;

: Save volatile registers

MOVZWL #BIG\_CDRP,R1

: Get total size

BSBW ALLOMEM

: Allocate them

BLBC R0,208

: No memory available

: Fill in skeletal CDRP

MOVQ R1,-(SP)

: Save

MOVCS #0,(SP),#0,R1,(R2)

: Clear it all out

MOVQ (SP)+,R4

: Recover length

MOVAL -CDRPSL\_10QFL(R5),R5

: Set to the CDRP portion

MOVB #DYNB\_CDRP,-

CDRPSB\_IRP\_TYPE(R5)

: Fill in size/type/rmod fields

MOVW R4,CDRPSB\_IRP\_SIZE(R5)

: Fill in size/type/rmod fields

MOVB #IPLS\_SCS,CDRPSB\_FIPL(R5)

: CDRP IPL

MOVB #DYNB\_CDRP,-

CDRPSB\_CD\_TYPE(R5)

: CDRP block type

MOVW #CDRPSL\_10QFL,-

CDRPSB\_CDRPSIZE(R5)

: Ditto for smaller portion

MOVB IO\_PRI,CDRPSB\_PRI(R5)

: Set the IO priority

MOVAL CDRPSB\_LBUFF(R5),-

: Set pointer to local buffer handle

CDRPSL\_LBUFH\_AD(R5)

INCM INI\_PKT

: Total packets

POPR #M&lt;R1,R2,R3,R4&gt;

: Restore volatile registers

BSBB FREE\_STRUCT

: Put on free list -- has side effect

: of initializing CDRPSL\_FQFL linkage

55 E433 DF 0F 1C39 3635  
10 1D 1C3E 3636  
E404 CF B7 1C40 3637  
50 01 D0 1C44 3638  
05 1C47 3639  
1C48 3640  
50 0124 BF 3C 1C48 3641  
1E BA 1C4D 3642  
05 1C4F 3643  
1C50 3644  
1E BB 1C50 3645  
51 00E0 BF 3C 1C52 3646  
E9FA 30 1C57 3647  
EB 50 E9 1C5A 3648  
1C5D 3649  
1C5D 3650  
1C5D 3651  
1C5D 3652  
7E 51 7D 1C5D 3653  
6E 00 2C 1C60 3654  
54 8E 7D 1C66 3655  
55 60 A5 DE 1C69 3656  
AA A5 0A 90 1C6D 3657  
1C71 3658  
A8 A5 54 B0 1C71 3659  
08 A5 08 90 1C75 3660  
0A A5 39 90 1C79 3661  
1C7D 3662  
FFA0 BF B0 1C7D 3663  
08 A5 1C81 3664  
C3 A5 E851 CF 90 1C83 3665  
2C A5 68 A5 DE 1C89 3666  
1C8E 3667  
E384 CF B6 1C8E 3668  
1E BA 1C92 3669  
02 10 1C94 3670  
1C96 3671

MOVQ R1,-(SP) : Save  
MOVCS #0,(SP),#0,R1,(R2) : Clear it all out  
MOVQ (SP)+,R4 : Recover length  
MOVAL -CDRPSL\_10QFL(R5),R5 : Set to the CDRP portion  
MOVB #DYNB\_CDRP,- :  
CDRPSB\_IRP\_TYPE(R5) : Fill in size/type/rmod fields  
MOVW R4,CDRPSB\_IRP\_SIZE(R5) : Fill in size/type/rmod fields  
MOVB #IPLS\_SCS,CDRPSB\_FIPL(R5) : CDRP IPL  
MOVB #DYNB\_CDRP,- :  
CDRPSB\_CD\_TYPE(R5) : CDRP block type  
MOVW #CDRPSL\_10QFL,- :  
CDRPSB\_CDRPSIZE(R5) : Ditto for smaller portion  
MOVB IO\_PRI,CDRPSB\_PRI(R5) : Set the IO priority  
MOVAL CDRPSB\_LBUFF(R5),- : Set pointer to local buffer handle  
CDRPSL\_LBUFH\_AD(R5) :  
INCM INI\_PKT : Total packets  
POPR #M<R1,R2,R3,R4> : Restore volatile registers  
BSBB FREE\_STRUCT : Put on free list -- has side effect  
: of initializing CDRPSL\_FQFL linkage

```
A1 11 1C96 3692 BRB GET_STRUCT ; Try again
      1C98 3693 .....
      1C98 3694 .....
      1C98 3695 .....
      1C98 3696 .....
      1C98 3697 .....
      1C98 3698 .....
      1C98 3699 .....
      1C98 3700 .....
      1C98 3701 .....
      1C98 3702 Input:
      1C98 3703 RS --> CDRP
      1C98 3704
      1C98 3705 Output:
      1C98 3706 NONE
      1C98 3707
      1C98 3708
      1C98 3709 FREE_STRUCT:
      1C98 3710 ; Mung structure to trap bugs
      1C98 3711 MOVL #1,CDRPSL_IQOBL(R5) ; Zap IRP queue linkage to catch bugs
      1C98 3712 MOVL #1,CDRPSL_FOBL(R5) ; Zap CDRP queue linkage to catch bugs
      1C98 3713 MOVL #1,CDRPSL_UOBL(R5) ; Zap another queue linkage
      1C98 3714 MOVL #1,CDRPSL_CDT(R5) ; Zap CDT address
      1C98 3715 MOVL #1,CDRPSL_CDTX(R5) ; Zap CDT address
      1C98 3716 MOVL #1,CDRPSL_MSGBUF(R5) ; Zap SCS message buffer address
      1C98 3717 MOVL #1,CDRPSL_MSG_BUF(R5) ; Zap SCS message buffer address
      1C98 3718 MOVL #1,CDRPSL_UOBT(R5) ; Zap UQB address
      1C98 3719 MOVL #1,CDRPSL_MST_UQB(R5) ; Zap master UQB address
      1C98 3720 MOVL #1,CDRPSL_FR3X(R5) ; Alias IRPSL_MAIN(CDRP)
      1C98 3721 MOVL #1,CDRPSL_HQB(R5) ; Zap HQB address
      1C98 3722 MOVW #63,CDRPSL_STATE(R5) ; Set state invalid
      1C98 3723 MOVAB B-208,CDRPSL_FPC(R5) ; Zap CDRP fork PC
      1C98 3724 MOVAB B-208,CDRPSL_PID(R5) ; Zap IRP fork PC
      1C98 3725 MOVAB B-208,CDRPSL_FPCX(R5) ; Zap another fork PC
      1C98 3726 ; Now deallocate the structure
      1C98 3727 ; Avoid problems (maybe) by hanging onto packets forever
      1C98 3728 CMPW NUM_PKT,MAX_PKT ; Are enough packets cached?
      1C98 3729 RGEQU 108 ; Branch if yes
      1C98 3730 INSQUE (R5),@CDRP_LIST+4 ; Return the cdrp
      1C98 3731 INCW NUM_PKT ; Account for it
      1C98 3732 RSB
      1C98 3733
      1C98 3734 ; 108: PUSHR #M<R0,R1,R2,R3> ; Save volatile registers
      1C98 3735 MOVAL CDRPSL_IQOFL(R5),R0 ; Structure address
      1C98 3736 BSBW DEALMEM ; Deallocate structure
      1C98 3737 POPR #M<R0,R1,R2,R3> ; Restore volatile registers
      1C98 3738 DECW INI_PKT ; Drop count of allocated structures
      1C98 3739 RSB ; Return
      1C98 3740
      1C98 3741 208: BUG_CHECK MSCPSERV,FATAL
      1C98 3742
```

A4 A5 01 DO 1C98 3711  
04 A5 01 DO 1C9C 3712  
EC A5 01 DO 1CA0 3713  
24 A5 01 DO 1CA4 3714  
60 A5 01 DO 1CAB 3715  
B0 A5 01 DO 1CAC 3716  
1C A5 01 DO 1CB0 3717  
C4 A5 01 DO 1CB4 3718  
78 A5 01 DO 1CB8 3719  
54 A5 01 DO 1CBC 3720  
7C A5 01 DO 1CC0 3721  
D6 A5 3F B0 1CC4 3722  
0C A5 E1'AF 9E 1CC8 3723  
AC A5 E1'AF 9E 1CCD 3724  
50 A5 E1'AF 9E 1CD2 3725  
E398 DF 65 OE 1CD7 3726  
E368 CF B6 1CDC 3731  
05 1CE0 3732  
1CE1 3733  
1CE1 3734  
1CE1 3735  
1CE1 3736  
1CE1 3737  
1CE1 3738  
1CE1 3739  
1CE1 3740  
1CE1 3741  
1CE5 3742

```
00000000
20 20 20 4B 53 49 44 24 50 43 53 4D
20 20 20 20

00 00 00 1C 0C 14 10 00
00 00 00 0C 0C 24 24 0C
00 00 00 20 20 20 20 00
00 00 00 00 00 00 00 00
00 00 00 00 00 20 20 20

00 00 00 1C 30 14 10 00
00 00 00 30 0C 2C 2C 0C
00 00 00 0C 20 20 20 20
00 00 00 00 00 00 00 00
00 00 00 00 00 20 20 20

FFFF 1D4C 3769 .WORD ^CO
FFFF 1D4E 3770 .WORD ^CO
FFFF 1D50 3771 .WORD ^CO
DFFE 1D52 3772 .WORD ^C<MSCPSM_MD_NXUNT!-
1D54 3773 MSCPSM_MD_CLSEX>
FFFF 1D56 3774 .WORD ^CO
FFFF 1D58 3775 .WORD ^CO
FFFF 1D5A 3776 .WORD ^CO
FFFF 1D5C 3777 .WORD ^CO
DFFC 1D5E 3778 .WORD ^C<MSCPSM_MD_SPNDW!-
1D5E 3779 MSCPSM_MD_ALLCD!-
1D5E 3780 MSCPSM_MD_CLSEX>
DFEB 1D5E 3781 .WORD ^C<MSCPSM_MD_RIP!-
1D60 3782 MSCPSM_MD_IGNMF!-
1D60 3783 MSCPSM_MD_STWRP!-
1D60 3784 MSCPSM_MD_SHDSP!-
1D60 3785 MSCPSM_MD_CLSEX>
DFEB 1D60 3786 .WORD ^C<MSCPSM_MD_STWRP!-
1D62 3787 MSCPSM_MD_SHDSP!-
1D62 3788 MSCPSM_MD_CLSEX>
FFFF 1D62 3789 .WORD ^CO
FFFF 1D64 3790 .WORD ^CO
FFFF 1D66 3791 .WORD ^CO
FFFF 1D68 3792 .WORD ^CO
FFFF 1D6A 3793 .WORD ^CO
507F 1D6C 3794 .WORD ^C<MSCPSM_MD_EXPRS!-
1D6E 3795 MSCPSM_MD_CLSEX!-
1D6E 3796 MSCPSM_MD_SCCHM!-
1D6E 3797 MSCPSM_MD_SCCHL!-
1D6E 3798 MSCPSM_MD_SEREC!-
1D6E 3799 MSCPSM_MD_SECOR!-
```

.SBTTL READ ONLY DATA

.ALIGN LONG

LIS\_CDT:.LONG 0

SERV\_INFO:

SERV\_NAME:

.ASCII /MSCP8DISK /

COM\_PKT\_LEN:

.BYTE 0\*4,4\*4,5\*4,3\*4,7\*4,0\*4,0\*4,0

.BYTE 3\*4,9\*4,9\*4,3\*4,0\*4,0\*4,0\*4,0

.BYTE 0\*4,8\*4,8\*4,8\*4,8\*4,0\*4,0\*4,0

.BYTE 0\*4,0\*4,0\*4,0\*4,0\*4,0\*4,0\*4,0

.BYTE 8\*4,8\*4,8\*4,0\*4,0\*4,0\*4,0\*4,0

END\_PKT\_LEN:

.BYTE 0\*4,4\*4,5\*4,12\*4,7\*4,0\*4,0\*4,0

.BYTE 3\*4,11\*4,11\*4,3\*4,0\*4,0\*4,0\*4,0

.BYTE 8\*4,8\*4,8\*4,8\*4,3\*4,0\*4,0\*4,0

.BYTE 0\*4,0\*4,0\*4,0\*4,0\*4,0\*4,0\*4,0

.BYTE 8\*4,8\*4,8\*4,0\*4,0\*4,0\*4,0\*4,0

MOD\_TBL:

1-ABORT

2-GET CMD STS

3-GET UNIT STS

4-SET CTRL CHAR

5

6

7

8-AVAILABLE

9-ONLINE

10-SET UNIT CHAR

11-DETERMINE ACCESS PATH

12

13

14

15

16-ACCESS

|      |      |      |       |                      |                     |
|------|------|------|-------|----------------------|---------------------|
| 507F | 1D6E | 3800 |       | MSCPSM MD SSHDW>     |                     |
|      | 1D6E | 3801 | .WORD | ^C<MSCPSM MD EXPRS!- | : 17-COMP CTRL DATA |
|      | 1D70 | 3802 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D70 | 3803 |       | MSCPSM MD SCCMH!-    |                     |
|      | 1D70 | 3804 |       | MSCPSM MD SCCHL!-    |                     |
|      | 1D70 | 3805 |       | MSCPSM MD SEREC!-    |                     |
|      | 1D70 | 3806 |       | MSCPSM MD SECOR!-    |                     |
|      | 1D70 | 3807 |       | MSCPSM MD SSHDW>     |                     |
| 4E0F | 1D70 | 3808 | .WORD | ^C<MSCPSM MD EXPRS!- | : 18-ERASE          |
|      | 1D72 | 3809 |       | MSCPSM MD SEREC!-    |                     |
|      | 1D72 | 3810 |       | MSCPSM MD ERROR!-    |                     |
|      | 1D72 | 3811 |       | MSCPSM MD WOKVL!-    |                     |
|      | 1D72 | 3812 |       | MSCPSM MD WOKNV!-    |                     |
|      | 1D72 | 3813 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D72 | 3814 |       | MSCPSM MD WRSEQ!-    |                     |
|      | 1D72 | 3815 |       | MSCPSM MD SSHDW>     |                     |
| 5C7C | 1D72 | 3816 | .WORD | ^C<MSCPSM MD EXPRS!- | : 19-FLUSH          |
|      | 1D74 | 3817 |       | MSCPSM MD FLENU!-    |                     |
|      | 1D74 | 3818 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D74 | 3819 |       | MSCPSM MD SECOR!-    |                     |
|      | 1D74 | 3820 |       | MSCPSM MD SEREC!-    |                     |
|      | 1D74 | 3821 |       | MSCPSM MD SSHDW!-    |                     |
|      | 1D74 | 3822 |       | MSCPSM MD VOLTL>     |                     |
| 5FFE | 1D74 | 3823 | .WORD | ^C<MSCPSM MD EXPRS!- | : 20-REPLACE        |
|      | 1D76 | 3824 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D76 | 3825 |       | MSCPSM MD PRIMR>     |                     |
| FFFF | 1D76 | 3826 | .WORD | ^CO                  | : 21                |
| FFFF | 1D78 | 3827 | .WORD | ^CO                  | : 22                |
| FFFF | 1D7A | 3828 | .WORD | ^CO                  | : 23                |
| FFFF | 1D7C | 3829 | .WORD | ^CO                  | : 24-MAINT READ     |
| FFFF | 1D7E | 3830 | .WORD | ^CO                  | : 25-MAINT WRITE    |
| FFFF | 1D80 | 3831 | .WORD | ^CO                  | : 26                |
| FFFF | 1D82 | 3832 | .WORD | ^CO                  |                     |
| FFFF | 1D84 | 3833 | .WORD | ^CO                  | : 28                |
| FFFF | 1D86 | 3834 | .WORD | ^CO                  |                     |
| FFFF | 1D88 | 3835 | .WORD | ^CO                  | : 30                |
| FFFF | 1D8A | 3836 | .WORD | ^CO                  |                     |
| 507F | 1D8C | 3837 | .WORD | ^C<MSCPSM MD EXPRS!- | : 32-COMP HOST DATA |
|      | 1D8E | 3838 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D8E | 3839 |       | MSCPSM MD SCCMH!-    |                     |
|      | 1D8E | 3840 |       | MSCPSM MD SCCHL!-    |                     |
|      | 1D8E | 3841 |       | MSCPSM MD SEREC!-    |                     |
|      | 1D8E | 3842 |       | MSCPSM MD SECOR!-    |                     |
|      | 1D8E | 3843 |       | MSCPSM MD SSHDW>     |                     |
| 107F | 1D8E | 3844 | .WORD | ^C<MSCPSM MD EXPRS!- | : 33-READ           |
|      | 1D90 | 3845 |       | MSCPSM MD COMP!-     |                     |
|      | 1D90 | 3846 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D90 | 3847 |       | MSCPSM MD SCCMH!-    |                     |
|      | 1D90 | 3848 |       | MSCPSM MD SCCHL!-    |                     |
|      | 1D90 | 3849 |       | MSCPSM MD SEREC!-    |                     |
|      | 1D90 | 3850 |       | MSCPSM MD SECOR!-    |                     |
|      | 1D90 | 3851 |       | MSCPSM MD SSHDW>     |                     |
| 0C0F | 1D90 | 3852 | .WORD | ^C<MSCPSM MD EXPRS!- | : 34-WRITE          |
|      | 1D92 | 3853 |       | MSCPSM MD COMP!-     |                     |
|      | 1D92 | 3854 |       | MSCPSM MD CLSEX!-    |                     |
|      | 1D92 | 3855 |       | MSCPSM MD ERROR!-    |                     |
|      | 1D92 | 3856 |       | MSCPSM MD WRSEQ!-    |                     |

```

1D92 3857 MSCPSM_MD_WBKNV!-
1D92 3858 MSCPSM_MD_WBKVL!-
1D92 3859 MSCPSM_MD_SEREC!-
1D92 3860 MSCPSM_MD_SECOR!-
1D92 3861 MSCPSM_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>8-16
00000079 1DA0 3873 PATCH: .LONG 120
00001DA8 1DA4 3874 .LONG PATCH+8
00001E20 1DA8 3875 .BLKB 120
1E20 3876

```

```
1E20 3878 .SBTTL INITIALIZATION ROUTINE
1E20 3879
1E20 3880 :++
1E20 3881 :
1E20 3882 : This routine is called to initialize MSCP server from SYSGEN.
1E20 3883 :
1E20 3884 : Inputs:
1E20 3885 :
1E20 3886 : AP --> #Args
1E20 3887 :
1E20 3888 : action #
00000008 1E20 3888 BUF=8 : BUFFers
0000000C 1E20 3889 PAC=12 : PACKets
00000010 1E20 3890 HOS=16 : HOSTs
00000014 1E20 3891 TIM=20 : TIMEout
00000018 1E20 3892 PRI=24 : PRIority
0000001C 1E20 3893 SMA=28 : SMAllest buffer
00000020 1E20 3894 FRA=32 : FRActional buffer
1E20 3895 :
1E20 3896 : Outputs:
1E20 3897 :
1E20 3898 : R0 Status code
1E20 3899 :
1E20 3900 :--
1E20 3901 :
1E20 3902 .ENABLE LSB
1E20 3903
50 0124 8F 3C 1E20 3904 10$: MOVZWL #SS$_INSFMEM,R0 : Insufficient non-paged pool
00FA 31 1E25 3905 BRW 40$
1E28 3906
1E28 3907 START_CONT:
1E28 3908 MOVL BUF(AP),R1 : Pick up user specified
51 08 AC D0 1E28 3909 MOVAB 511(R1),R1
51 01FF C1 9E 1E2C 3909 BICW #511,R1 : Round up to page
51 01FF 8F AA 1E31 3910 MOVL R1,MAX_BUF : Save size
E219 CF 51 D0 1E36 3911 ADDL #16,R1 : Allow for header
51 10 C0 1E38 3912 SETIPL #2 : Set us non-deletable
1E3E 3913 BSBW ALLOMEM : Allocate it
E810 30 1E41 3914 BLBC R0,10$ : Can't allocate buffer
D9 50 E9 1E44 3915 MOVL R1,BUFF_LENGTH : Length of buffer
E680 CF 51 D0 1E47 3916 MOVL R2,BUFF_ADDR : Buffer header address
E67F CF 52 D0 1E4C 3917 ADDL3 R1,R2,BUFF_END : First byte beyond buffer
E67D CF 52 51 C1 1E51 3918 CLRL (R2)+ : Clear FLINK,BLINK
82 51 00130000 8F C1 1E59 3919 ADDL3 #<DYN$C_BUF10@16>,R1,(R2)+ : Set pool info
82 82 D4 1E61 3921 CLRL (R2)+ : Skip to 16-byte boundary
E1E8 CF 52 D0 1E63 3922 MOVL R2,BUFF_HEAD : Set pointer to buffer pool
82 82 D4 1E68 3923 CLRL (R2)+ : Set up head
62 51 10 C3 1E6A 3924 SUBL3 #16,R1,(R2) : Ditto
1E6E 3925 :
1E6E 3926 : Pre-allocate CDRP's
1E6E 3927 :
1E6E 3928 MOVL PAC(AP),R7 : Pick up user specified
57 0C AC D0 1E6E 3929 CMPL R7,#2 : Must be at least 2
02 57 D1 1E72 3929 BGEQU 15$ : Branch if 2 or more
57 03 1E 1E75 3930 MOVL #2,R7 : Make it 2
E1C5 CF 57 B0 1E77 3931 15$: MOVW R7,PACKET : Send to MSCP
51 10 AC D0 1E7A 3932 MOVL HOS(AP),R1 : Pick un number of hosts
E1C4 CF 51 B0 1E7F 3933 MOVW R1,INI_HOST : Save in comm area
1E83 3934
```

```

E1B7 E1C1 CF S1 B0 1E88 3935      MOVW R1,NUM_HOST      ; Save in comm area
      CF S7 S1 CS 1E8D 3936      MULL3 R1,R7,MAX_PKT      ; Times the hosts = maximum packets to cache
      1E93 3937      :
      1E93 3938      : Set up the queues.
      1E93 3939      :
      55 E169 CF 9E 1E93 3940      MOVAB MSCP_BEGIN,R5      ; Point to our header
      54 E1D4 CF DE 1E98 3941      MOVAL CDRP_LIST,R4      ; Start address of queue
      50 06 DO 1E9D 3942      MOVL #2*MSUSC_NUMQUE,R0      ; Number of queues
      84 55 CO 1EA0 3943 20$: ADDL2 R5,(R4)+      ; Make addresses absolute
      FA 50 FS 1EA3 3944      SOBGTR R0,20$      ; Loop thru
      50 E60A CF 7E 1EA6 3945      MOVAQ HQBLIST,R0      ; List of HQB's
      60 50 DO 1EAB 3946      MOVL R0,(R0)      ; Initialize forward pointer
      60 80 DE 1EAE 3947      MOVAL (R0)+,(R0)      ; Initialize backward pointer
      1EB1 3948
E620 CF 20 1B AC 83 1EB1 3949      SUBB3 PRI(AP),#32,'^_PRI      ; Set the IO priority to internal value
      1EB8 3950
      1EB8 3951      :
      1EB8 3952      : Set some miscellaneous values
      1EB8 3953      :
      50 20 AC DO 1EB8 3954 30$: MOVL FRA(AP),R0      ; Set the fractional buffer per I/O
      0000FE00 BF 50 D1 1EBC 3955      CMPL R0,#127*512      ; Is it bigger than 127 block?
      07 1B 1EC3 3956      BLEQU 32$      ; No, all is well
      50 0000FE00 BF DO 1EC5 3957      MOVL #127*512,R0      ; Biggest fraction is 127 block
      E188 CF 50 DO 1ECC 3958 32$: MOVL R0,FRACTIC      ; Must be I/O viable on any disk
      50 1C AC DO 1ED1 3959      MOVL SMA(AP),R0      ; Set the smallest fragment per I/O
      E17A CF 50 D1 1ED5 3960      CMPL R0,MAX_BUF      ; SMALL may not be larger than MAX_BUF
      05 1B 1EDA 3961      BLEQU 35$      ; Branch if less than MAX_BUF
      50 E174 CF DO 1EDC 3962      MOVL MAX_BUF,R0      ; Use MAX_BUF
      E172 CF 50 DO 1EE1 3963 35$: MOVL R0,SMALL      ; Set the smallest fragment per I/O
      E2B0 CF 14 AC B0 1EE6 3964      MOVW TIM(AP),CTRL_INFO+4      ; Set the controller timeout
      00000000'GF DO 1EEC 3965      MOVL G^SCS$GB_SYSTEMID,-      ;
      E29B CF B0 1EF2 3966      CTRL_ID      ; Set the controller id
      00000004'GF B0 1EFS 3967      MOVW G^SCS$GB_SYSTEMID+4,-      ;
      E296 CF B0 1EFB 3968      CTRL_ID+4      ;
      1EFE 3969      :
      1EFE 3970      : Establish link with PORT_DRIVER
      1EFE 3971      :
      1EFE 3972      :
      1F01 3973      SETIPL #IPL$ SCS      ; Coordinate activities
      1F01 3974      LISTEN MSGADR=LIS_FORK,-      ; Establish ourselves as listener
      1F01 3975      ERRADR=LIS_ERR,-
      1F01 3976      LPRNAM=SERV_NAME,-
      1F01 3977      PRINFO=SERV_INFO
      FDC9 CF 53 DO 1F1A 3977      MOVL R3,LIS_CDT      ; Store the CDT away
      50 01 DO 1F1F 3978      MOVL #SS$_NORMAL,R0      ; Set success
      05 1F22 3979 40$: SETIPL #0      ; Drop IPL
      1F25 3980      RSB      ; Leave, finished initialization
      1F26 3981
      1F26 3982      .DISABLE      LSB
      1F26 3983
      00001F26 1F26 3984      LC=
      00001F30 1F26 3985      .=<LC+15>B-16
      1F30 3986
      1F30 3987 MSCP_END:
      1F30 3988      .END
```

MSCP  
Symbol table

- MSCP Server - Emulator

L 3

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR:2

Page 92  
(1)

```

ABORT          00000998 R 02
ABORTER        00000147D R R 02
ABORTER        00000147F R R 02
ACCESS         000001379 R R 02
ADD            0000004FE R R 02
ALLOWEM       000000654 R R 02
ATTN_CLEANUP   000000C09 R R 02
AVAIL          000000D47 R R 02
AVAIL_ALL      000000CFF R R 02
AV_CASE2       000000D03 R R 02
AV_CASE2X      000000C80 R R 02
AV_CASE3       000000B3F R 02
BACK           000000CE0 R 02
BIG_CDRP       000000280 R R 02
BLK_COUNT      000000840 R R 02
BLOCKEDM       000000847 R R 02
BLOCKEDT       000001288 R 02
BLOCK_CDRP     0000000C8 R 02
BUF            000000400 R R 02
BUF_ADDR       000000404 R R 02
BUF_END        000000050 R R 02
BUF_HEAD       0000004CC R 02
BUGS_MSCPSEV   ***** X 02
CASE2A         000000EBF R R 02
CASE2A_DONE    000001059 R R 02
CASE2A_ERR     000000F18 R R 02
CASE2A_ERR_OUT 000000F20 R R 02
CCHAR          00000018C R R 02
CCH            000001060 R 02
CDRPSB_CD_TYPE 00000000A R 02
CDRPSB_EIR     0000000C2 R 02
CDRPSB_FIPL    00000000B R 02
CDRPSB_FLAGS   0000000FA R 02
CDRPSB_IRP_TYPE 0000000AA R 02
CDRPSB_LBUF    000000068 R 02
CDRPSB_OPCODE  0000000FB R 02
CDRPSB_PRI     0000000C3 R 02
CDRPSB_SHOW_CNT 0000000C2 R 02
CDRPSB_CD_LEN  000000048 R 02
CDRPSB-ST_BLK_QUE 000000007 R 02
CDRPSB-ST_DATA_WAIT 000000003 R 02
CDRPSB-ST_DRVBD_WAIT 000000002 R 02
CDRPSB-ST_DRVBD_WAIT 000000004 R 02
CDRPSB-ST_INVALID 00000000F R 02
CDRPSB-ST_MAP_WAIT 000000001 R 02
CDRPSB-ST_MEM_WAIT 000000000 R 02
CDRPSB-ST_MSGT_WAIT 000000009 R 02
CDRPSB-ST_MSG_WAIT 000000003 R 02
CDRPSB-ST_OVER_QUE 000000008 R 02
CDRPSB-ST_SHOW_QUE 000000006 R 02
CDRPSB-ABORT   0000000E0 R 02
CDRPSB-ABORT   0000000FB R 02
CDRPSB-AST     0000000B0 R 02
CDRPSB-ASTPRM  0000000B4 R 02
CDRPSB-BLNT    0000000D2 R 02
CDRPSB-BUFADR  0000000B8 R 02

```

```

CDRPSB-BUFLEN  0000000B4 R 02
CDRPSB-CDT     000000024 R 02
CDRPSB-CDTH    000000060 R 02
CDRPSB-CHD_STS 0000000F4 R 02
CDRPSB-DIAGBUF 0000000EC R 02
CDRPSB-EX-END   0000000F4 R 02
CDRPSB-IPC      00000000C R 02
CDRPSB-IRCA     000000050 R 02
CDRPSB-IUML     000000004 R 02
CDRPSB-IQFL     000000000 R 02
CDRPSB-FRS      000000010 R 02
CDRPSB-FRSX     000000054 R 02
CDRPSB-FRAX     000000058 R 02
CDRPSB-HOB      00000007C R 02
CDRPSB-IOOBL    0000000A4 R 02
CDRPSB-IOOFL    0000000A0 R 02
CDRPSB-IOSD     0000000C4 R 02
CDRPSB-IOST1    0000000DB R 02
CDRPSB-LBN      000000048 R 02
CDRPSB-LBOFF    000000030 R 02
CDRPSB-LBUFH_AD 00000002C R 02
CDRPSB-MSGBUF   0000000B0 R 02
CDRPSB-MSGBUF   00000001C R 02
CDRPSB-MSG-UB   000000078 R 02
CDRPSB-OBCT     0000000E4 R 02
CDRPSB-PID      0000000AC R 02
CDRPSB-RBOFF    000000038 R 02
CDRPSB-RBUFH_AD 000000054 R 02
CDRPSB-RSPID    000000020 R 02
CDRPSB-RUCPTR   000000018 R 02
CDRPSB-SAVD_RTN 000000018 R 02
CDRPSB-SAVD_RTX 00000005C R 02
CDRPSB-SEGVBN   0000000E8 R 02
CDRPSB-SHOW_LNK 00000004C R 02
CDRPSB-STRT_UB  000000074 R 02
CDRPSB-SVAPTE   0000000CC R 02
CDRPSB-UCB      0000000BC R 02
CDRPSB-UCB      0000000C4 R 02
CDRPSB-UCB-BL   0000000EC R 02
CDRPSB-UCB-FL   0000000E8 R 02
CDRPSB-WIND      0000000B8 R 02
CDRPSB-XCT_LEN  00000003C R 02
CDRPSB-XCT_LENX 000000064 R 02
CDRPSV-ABORT    000000004 R 02
CDRPSV-BUFFER    000000001 R 02
CDRPSV-NONSTBUF 000000005 R 02
CDRPSV-SLAVE     000000000 R 02
CDRPSV-BOFF      0000000D0 R 02
CDRPSV-CDRPSIZE 0000000C8 R 02
CDRPSV-FUNC      0000000C0 R 02
CDRPSV-IRP_SIZE 0000000AB R 02
CDRPSV-SLOT      0000000FB R 02
CDRPSV-STATE     0000000D6 R 02
CDRPSV-STS       0000000CA R 02
CDRPSV-LTST      000000070 R 02
CDRPSV-SAVE      00000006C R 02
CDRPSV-SIZE      000000068 R 02

```

MSCP  
Symbol table

- MSCP Server - Emulator

L 3

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 92  
(1)

MS  
V0

|                       |             |   |    |                  |   |          |   |    |
|-----------------------|-------------|---|----|------------------|---|----------|---|----|
| ABORT                 | 00000998    | R | 02 | CDRPSL_BUFLN     | = | FFFFFFB4 |   |    |
| ABO_ER                | 0000147D    | R | 02 | CDRPSL_CDT       | = | 00000024 |   |    |
| ABX_ER                | 0000147F    | R | 02 | CDRPSL_CDTX      | = | 00000060 |   |    |
| ACCESS                | 00001379    | R | 02 | CDRPSL_CMD_STS   | = | FFFFFFF4 |   |    |
| ADD                   | 000004FE    | R | 02 | CDRPSL_DIAGBUF   | = | FFFFFFEC |   |    |
| ALLOMEM               | 00000654    | R | 02 | CDRPSL_EXTEND    | = | FFFFFFF4 |   |    |
| ATTN_CLEANUP          | 0000056B    | R | 02 | CDRPSL_FPC       | = | 0000000C |   |    |
| AVAIL                 | 00000C09    | R | 02 | CDRPSL_FPC_A     | = | 00000050 |   |    |
| AVAIL_ALL             | 00000D47    | R | 02 | CDRPSL_FQBL      | = | 00000004 |   |    |
| AV_CASE2              | 00000CFE    | R | 02 | CDRPSL_FQFL      | = | 00000000 |   |    |
| AV_CASE2X             | 00000D05    | R | 02 | CDRPSL_FR3       | = | 00000010 |   |    |
| AV_CASE3              | 00000CBD    | R | 02 | CDRPSL_FR3X      | = | 00000054 |   |    |
| BACK                  | 00001B3F    | R | 02 | CDRPSL_FR4X      | = | 00000058 |   |    |
| BIG_CDRP              | = 000000E0  |   |    | CDRPSL_HQB       | = | 0000007C |   |    |
| BLK_COUNT             | 000002B0    | R | 02 | CDRPSL_IOQBL     | = | FFFFFFA4 |   |    |
| BLOCKEDH              | 00000840    | R | 02 | CDRPSL_IOQFL     | = | FFFFFFA0 |   |    |
| BLOCKEDT              | 00000847    | R | 02 | CDRPSL_IOSB      | = | FFFFFFC4 |   |    |
| BLOCK_CDRP            | 000012B8    | R | 02 | CDRPSL_IOST1     | = | FFFFFFD8 |   |    |
| BUF                   | = 00000008  |   |    | CDRPSL_LBN       | = | 00000048 |   |    |
| BUFF_ADDR             | 000004D0    | R | 02 | CDRPSL_LBOFF     | = | 00000030 |   |    |
| BUFF_END              | 000004D4    | R | 02 | CDRPSL_LBUFH_AD  | = | 0000002C |   |    |
| BUFF_HEAD             | 00000050    | R | 02 | CDRPSL_MSGBUF_X  | = | FFFFFFB0 |   |    |
| BUFF_LENGTH           | 000004CC    | R | 02 | CDRPSL_MSG_BUF   | = | 0000001C |   |    |
| BUGS_MSCPSERV         | *****       | X | 02 | CDRPSL_MST_UMB   | = | 00000078 |   |    |
| CASE2A                | 00000EBF    | R | 02 | CDRPSL_OBCNT     | = | FFFFFFE4 |   |    |
| CASE2A_DONE           | 00001059    | R | 02 | CDRPSL_PID       | = | FFFFFFAC |   |    |
| CASE2A_ERR            | 00000F1B    | R | 02 | CDRPSL_RBOFF     | = | 00000038 |   |    |
| CASE2A_ERR_OUT        | 00000F20    | R | 02 | CDRPSL_RBUFH_AD  | = | 00000034 |   |    |
| CCHAR                 | 00000EBC    | R | 02 | CDRPSL_RSPID     | = | 00000020 |   |    |
| CCX                   | 00001060    | R | 02 | CDRPSL_RWCPT     | = | 00000028 |   |    |
| CDRPSB_CD_TYPE        | = 0000000A  |   |    | CDRPSL_SAVD_RTN  | = | 00000018 |   |    |
| CDRPSB_EFN            | = FFFFFFFC2 |   |    | CDRPSL_SAVD_RTX  | = | 0000005C |   |    |
| CDRPSB_FIPL           | = 0000000B  |   |    | CDRPSL_SEGVBN    | = | FFFFFFE8 |   |    |
| CDRPSB_FLAGS          | = FFFFFFFFA |   |    | CDRPSL_SHDW_LNK  | = | 0000004C |   |    |
| CDRPSB_IRP_TYPE       | = FFFFFFFAA |   |    | CDRPSL_STRT_UMB  | = | 00000074 |   |    |
| CDRPSB_LBUFF          | = 00000068  |   |    | CDRPSL_SVAPTE    | = | FFFFFFFC |   |    |
| CDRPSB_OPCODE         | = FFFFFFFFB |   |    | CDRPSL_UCB       | = | FFFFFFBC |   |    |
| CDRPSB_PRI            | = FFFFFFFC3 |   |    | CDRPSL_UMB       | = | FFFFFFC4 |   |    |
| CDRPSB_SHDW_CNT       | = FFFFFFFC2 |   |    | CDRPSL_UMB_BL    | = | FFFFFFEC |   |    |
| CDRPSK_CD_LEN         | = 00000048  |   |    | CDRPSL_UMB_FL    | = | FFFFFFE8 |   |    |
| CDRPSK_ST_BLK_QUE     | = 00000007  |   |    | CDRPSL_WIND      | = | FFFFFFB8 |   |    |
| CDRPSK_ST_DATA_WAIT   | = 00000003  |   |    | CDRPSL_XCT_LEN   | = | 0000003C |   |    |
| CDRPSK_ST_DRVRO_WAIT  | = 00000002  |   |    | CDRPSL_XCT_LEN_X | = | 00000064 |   |    |
| CDRPSK_ST_DRVWRT_WAIT | = 00000004  |   |    | CDRPSV_ABORT     | = | 00000004 |   |    |
| CDRPSK_ST_INVALID     | = 0000003F  |   |    | CDRPSV_BUFFER    | = | 00000001 |   |    |
| CDRPSK_ST_MAP_WAIT    | = 00000001  |   |    | CDRPSV_NOHSTBUF  | = | 00000005 |   |    |
| CDRPSK_ST_MEM_WAIT    | = 00000000  |   |    | CDRPSV_SLAVE     | = | 00000000 |   |    |
| CDRPSK_ST_MSGT_WAIT   | = 00000009  |   |    | CDRPSW_BOFF      | = | FFFFFFD0 |   |    |
| CDRPSK_ST_MSG_WAIT    | = 00000005  |   |    | CDRPSW_CDRPSIZE  | = | 00000008 |   |    |
| CDRPSK_ST_ONBLK_QUE   | = 00000008  |   |    | CDRPSW_FUNC      | = | FFFFFFC0 |   |    |
| CDRPSK_ST_SHQ_QUE     | = 00000006  |   |    | CDRPSW_IRP_SIZE  | = | FFFFFFA8 |   |    |
| CDRPSL_ABENT          | = FFFFFFFE0 |   |    | CDRPSW_SLOT      | = | FFFFFFF8 |   |    |
| CDRPSL_ARB            | = FFFFFFFF8 |   |    | CDRPSW_STATE     | = | FFFFFFD6 |   |    |
| CDRPSL_AST            | = FFFFFFFB0 |   |    | CDRPSW_STS       | = | FFFFFFCA |   |    |
| CDRPSL_ASTPRM         | = FFFFFFFB4 |   |    | CDRPS_LIST       | = | 00000070 | R | 02 |
| CDRPSL_BCNT           | = FFFFFFFD2 |   |    | CDRPS_SAVE       | = | 0000006C | R | 02 |
| CDRPSL_BUFADR         | = FFFFFFFB8 |   |    | CDRPS_SIZE       | = | 00000068 | R | 02 |

MSCP  
Symbol table

- MSCP Server - Emulator

M 3

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 93  
(1)

|                 |   |          |   |    |
|-----------------|---|----------|---|----|
| CDT\$B_RSTATION | = | 00000020 |   |    |
| CDT\$B_AUXSTRUC | = | 0000005C |   |    |
| CDT\$B_LCONID   | = | 00000018 |   |    |
| CDT\$B_PDT      | = | 00000010 |   |    |
| CHECK_ONLINE    |   | 00000DF5 | R | 02 |
| CLEAN_UP        |   | 00000B21 | R | 02 |
| CLEAN_UP_ONLCPY |   | 0000128C | R | 02 |
| CLUSGC_ALOCCLS  |   | *****    | 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\$C_FDT      | = | 00000008 |   |    |
| DEALLOC_BUFF    |   | 000012CE | R | 02 |
| DEALMEM         |   | 0000064E | R | 02 |
| DEALRO          |   | 0000064B | R | 02 |
| DEF_RTN         | = | 00000000 |   |    |
| DET_ACC_PATH    |   | 0000133F | R | 02 |
| DEVSV_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_PRT_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_WRTLCK      |   | 00001556 | R | 02 |
| ERR_X           |   | 00000937 | R | 02 |
| EX\$ALLOCATE    |   | *****    | X | 02 |
| EX\$ALONONPAGED |   | *****    | X | 02 |
| EX\$DEALLOCATE  |   | *****    | X | 02 |
| EX\$DEANONPAGED |   | *****    | X | 02 |
| EX\$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_REF_C     |   | 0000002C |   |    |
| HQB\$W_SIZE      |   | 00000008 |   |    |
| HQB\$W_USE_FRAC  |   | 0000001C |   |    |
| HQBLIST          |   | 00000484 | R | 02 |
| IMMED            |   | 00000989 | R | 02 |
| INI_HOST         |   | 0000004C | R | 02 |
| INI_PKT          |   | 00000046 | R | 02 |
| INV_FUNC         |   | 00001A70 | R | 02 |
| IOSM_DATACHECK   | = | 00004000 |   |    |
| IOSM_FORCERR     | = | 00000100 |   |    |
| IOSM_INHERLOG    | = | 00000800 |   |    |
| IOSM_INHRETRY    | = | 00008000 |   |    |
| IOS\$FCODE       | = | 00000006 |   |    |
| IOSV_FCODE       | = | 00000000 |   |    |
| IOS_AVAILABLE    | = | 00000011 |   |    |
| IOS_DSE          | = | 00000015 |   |    |
| IOS_PACKACK      | = | 00000008 |   |    |
| IOS_READPBLK     | = | 0000000C |   |    |
| IOS_UNLOAD       | = | 00000001 |   |    |
| IOS_WRITECHECK   | = | 0000000A |   |    |
| IOS_WRITEPBLK    | = | 0000000B |   |    |
| IOC\$CVTLOGPHY   |   | *****    | X | 02 |
| IO_PRI           |   | 000004D8 | R | 02 |
| IPC\$SCS         | = | 00000008 |   |    |
| IRP\$B_FLAGS     | = | 0000005A |   |    |
| IRP\$B_SHDW_CNT  | = | 00000022 |   |    |
| IRP\$C_LENGTH    | = | 000000C4 |   |    |
| IRP\$L_BCNT      | = | 00000032 |   |    |

MSCP  
Symbol table

- MSCP Server - Emulator

N 3

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 94  
(1)

MS  
V0

|                  |   |          |   |    |
|------------------|---|----------|---|----|
| IRPSL_FPCX       | = | 000000B0 |   |    |
| IRPSL_FR4X       | = | 000000B8 |   |    |
| IRPSL_IOST1      | = | 00000038 |   |    |
| IRPSL_LBN        | = | 000000A8 |   |    |
| IRPSL_MAINCDRP   | = | 000000B4 |   |    |
| IRPSL_PID        | = | 0000000C |   |    |
| IRPSL_SEQNUM     | = | 00000050 |   |    |
| IRPSL_SHDW_LNK   | = | 000000AC |   |    |
| IRPSL_SVAPTE     | = | 0000002C |   |    |
| IRPSL_UCB        | = | 0000001C |   |    |
| IRPSL_UQB        | = | 00000024 |   |    |
| IRPSM_PHYSIO     | = | 00000100 |   |    |
| IRPSV_PHYSIO     | = | 00000008 |   |    |
| IRPSW_FUNC       | = | 00000020 |   |    |
| IRPSW_SIZE       | = | 00000008 |   |    |
| IRPSW_SLOT       | = | 00000058 |   |    |
| IRPSW_STATE      | = | 00000036 |   |    |
| IRPSW_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 |
| MSCPSB_BUFFER    | = | 00000010 |   |    |
| MSCPSB_CNT_ALCS  | = | 00000004 |   |    |
| MSCPSB_OPCODE    | = | 00000008 |   |    |
| MSCPSB_RBNS      | = | 0000002E |   |    |
| MSCPSB_RCT_CPYS  | = | 0000002F |   |    |
| MSCPSK_CL_CNTRL  | = | 00000001 |   |    |
| MSCPSK_CM_EMULA  | = | 00000004 |   |    |
| MSCPSK_MIN_SIZE  | = | 0000000C |   |    |
| MSCPSK_OP_ABORT  | = | 00000001 |   |    |
| MSCPSK_OP_ACCES  | = | 00000010 |   |    |
| MSCPSK_OP_AVAIL  | = | 00000008 |   |    |
| MSCPSK_OP_COMP   | = | 00000020 |   |    |
| MSCPSK_OP_DTACP  | = | 0000000B |   |    |
| MSCPSK_OP_END    | = | 00000080 |   |    |
| MSCPSK_OP_ONLIN  | = | 00000009 |   |    |
| MSCPSK_OP_WRITE  | = | 00000022 |   |    |
| MSCPSK_SC_ALONL  | = | 00000008 |   |    |
| MSCPSK_SC_UNKNO  | = | 00000000 |   |    |
| MSCPSK_ST_ABRTD  | = | 00000002 |   |    |
| MSCPSK_ST_AVLBL  | = | 00000004 |   |    |
| MSCPSK_ST_CNTRLR | = | 0000000A |   |    |
| MSCPSK_ST_COMP   | = | 00000007 |   |    |
| MSCPSK_ST_DATA   | = | 00000008 |   |    |
| MSCPSK_ST_DRIVE  | = | 0000000B |   |    |
| MSCPSK_ST_HSTBF  | = | 00000009 |   |    |
| MSCPSK_ST_ICMD   | = | 00000001 |   |    |
| MSCPSK_ST_MFMT   | = | 00000005 |   |    |

|                 |   |          |
|-----------------|---|----------|
| MSCPSK_ST_OFFLN | = | 00000003 |
| MSCPSK_ST_SBCOD | = | 00000020 |
| MSCPSK_ST_SHST  | = | 0000000C |
| MSCPSK_ST_SUCC  | = | 00000000 |
| MSCPSK_ST_WRTPR | = | 00000006 |
| MSCPSL_BYTE_CNT | = | 0000000C |
| MSCPSL_CMD_REF  | = | 00000000 |
| MSCPSL_CMD_STS  | = | 00000010 |
| MSCPSL_EXCC_LBA | = | 00000014 |
| MSCPSL_FRST_BAD | = | 0000001C |
| MSCPSL_LBN      | = | 0000001C |
| MSCPSL_MEDIA_ID | = | 0000001C |
| MSCPSL_OUT_REF  | = | 0000000C |
| MSCPSL_UNT_SIZE | = | 00000024 |
| MSCPSL_VOL_SER  | = | 00000028 |
| MSCPSM_CF_ALTHS | = | 00000004 |
| MSCPSM_CF_REPLC | = | 00008000 |
| MSCPSM_MD_ALLCD | = | 00000001 |
| MSCPSM_MD_CLSEX | = | 00002000 |
| MSCPSM_MD_COMP  | = | 00004000 |
| MSCPSM_MD_ERROR | = | 00001000 |
| MSCPSM_MD_EXPRS | = | 00008000 |
| MSCPSM_MD_FLENU | = | 00000001 |
| MSCPSM_MD_IGNMF | = | 00000002 |
| MSCPSM_MD_NXUNT | = | 00000001 |
| MSCPSM_MD_PRIMR | = | 00000001 |
| MSCPSM_MD_RIP   | = | 00000001 |
| MSCPSM_MD_SCCHH | = | 00000800 |
| MSCPSM_MD_SCCHL | = | 00000400 |
| MSCPSM_MD_SECOR | = | 00000200 |
| MSCPSM_MD_SEREC | = | 00000100 |
| MSCPSM_MD_SHDSP | = | 00000010 |
| MSCPSM_MD_SPNDW | = | 00000002 |
| MSCPSM_MD_SSHDW | = | 00000080 |
| MSCPSM_MD_STWRP | = | 00000004 |
| MSCPSM_MD_VOLTL | = | 00000002 |
| MSCPSM_MD_WBKNV | = | 00000040 |
| MSCPSM_MD_WBKVL | = | 00000020 |
| MSCPSM_MD_WRSEQ | = | 00000010 |
| MSCPSM_UF_576   | = | 00000004 |
| MSCPSM_UF_CMPRD | = | 00000001 |
| MSCPSM_UF_CMPWR | = | 00000002 |
| MSCPSM_UF_REFLC | = | 00008000 |
| MSCPSM_UF_RMVBL | = | 00000080 |
| MSCPSM_UF_SCCHH | = | 00000800 |
| MSCPSM_UF_SCCHL | = | 00000400 |
| MSCPSM_UF_SSMEM | = | 00004000 |
| MSCPSM_UF_SSMST | = | 00000200 |
| MSCPSM_UF_WBKNV | = | 00000040 |
| MSCPSM_UF_WRTPH | = | 00002000 |
| MSCPSM_UF_WRTPS | = | 00001000 |
| MSCPSQ_CNT_ID   | = | 00000014 |
| MSCPSQ_TIME     | = | 00000014 |
| MSCPSQ_UNIT_ID  | = | 00000014 |
| MSCPSV_MD_COMP  | = | 0000000E |
| MSCPSV_MD_NXUNT | = | 00000000 |
| MSCPSV_MD_SEREC | = | 00000008 |

MSCP  
Symbol table

- MSCP Server - Emulator

B 4

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 95  
(1)

|                 |   |          |   |    |
|-----------------|---|----------|---|----|
| MSCPSV_MD_SHDSP | = | 00000004 |   |    |
| MSCPSV_MD_SPNDW | = | 00000001 |   |    |
| MSCPSV_MD_STWRP | = | 00000002 |   |    |
| MSCPSV_UF_SSMEM | = | 0000000E |   |    |
| MSCPSV_UF_SSMST | = | 00000009 |   |    |
| MSCPSV_UF_WRTPH | = | 0000000D |   |    |
| MSCPSV_UF_WRTPS | = | 0000000C |   |    |
| MSCPSW_CNT_FLGS | = | 0000000E |   |    |
| MSCPSW_COPY_SPD | = | 00000022 |   |    |
| MSCPSW_CYLINDER | = | 00000028 |   |    |
| MSCPSW_EXCL_LBC | = | 00000018 |   |    |
| MSCPSW_GROUP    | = | 00000026 |   |    |
| MSCPSW_HST_TMO  | = | 00000010 |   |    |
| MSCPSW_MODIFIER | = | 0000000A |   |    |
| MSCPSW_MULT_UNT | = | 0000000C |   |    |
| MSCPSW_RCT_SIZE | = | 0000002C |   |    |
| MSCPSW_SHDW_STS | = | 00000022 |   |    |
| MSCPSW_SHDW_UNT | = | 00000020 |   |    |
| MSCPSW_STATOS   | = | 0000000A |   |    |
| MSCPSW_TRACK    | = | 00000024 |   |    |
| MSCPSW_UNIT     | = | 00000004 |   |    |
| MSCPSW_UNT_FLGS | = | 0000000E |   |    |
| MSCPSW_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_UMB         | = | 0000002C |   |    |
| MSUSB_CTRL_DAT  |   | 000001A0 |   |    |
| MSUSB_SUBTYPE   |   | 0000000B |   |    |
| MSUSB_TYPE      |   | 0000000A |   |    |
| MSUSB_VEC       |   | 00000024 |   |    |
| MSUSC_LENGTH    |   | 000004CC |   |    |
| MSUSC_NUMQUE    | = | 00000003 |   |    |
| MSUSK_AC_ADD    | = | 00000003 |   |    |
| MSUSK_AC_DISC   | = | 00000005 |   |    |
| MSUSK_AC_REM    | = | 00000004 |   |    |
| MSUSK_AC_STOP   | = | 00000002 |   |    |
| MSUSK_AC_STRT   | = | 00000001 |   |    |
| MSUSK_LENGTH    |   | 000004CC |   |    |
| MSUSK_MAX_HOST  | = | 00000010 |   |    |
| MSUSK_MAX_OPC   | = | 00000040 |   |    |
| MSUSK_MAX_UNT   | = | 00000010 |   |    |
| MSUSK_UNT_SIZ   | = | 00000004 |   |    |
| MSUSL_ABORT     |   | 000001B4 |   |    |
| MSUSL_AVAIL     |   | 000001D0 |   |    |
| MSUSL_BLKCOUNT  |   | 000002B0 |   |    |
| MSUSL_BUFF_HEAD |   | 00000050 |   |    |
| MSUSL_CDRP_LIST |   | 00000070 |   |    |
| MSUSL_CDRP_SAVE |   | 0000006C |   |    |
| MSUSL_CDSV_SIZE |   | 00000068 |   |    |
| MSUSL_D1        |   | 00000000 |   |    |
| MSUSL_D2        |   | 00000004 |   |    |
| MSUSL_FRACTION  |   | 0000005C |   |    |
| MSUSL_GETCMD    |   | 000001B8 |   |    |
| MSUSL_GETUNIT   |   | 000001BC |   |    |

|                 |   |          |   |    |
|-----------------|---|----------|---|----|
| MSUSL_HOST_DSBL |   | 00000088 |   |    |
| MSUSL_HOST_VEC  |   | 00000110 |   |    |
| MSUSL_MAX_BUF   |   | 00000054 |   |    |
| MSUSL_MEM_WAIT  |   | 00000078 |   |    |
| MSUSL_ONLINE    |   | 000001D4 |   |    |
| MSUSL_OPCOUNT   |   | 000001B0 |   |    |
| MSUSL_P1        |   | 0000000C |   |    |
| MSUSL_P2        |   | 00000010 |   |    |
| MSUSL_READ      |   | 00000234 |   |    |
| MSUSL_SETCTRL   |   | 000001C0 |   |    |
| MSUSL_SETUNIT   |   | 000001D8 |   |    |
| MSUSL_SMALL     |   | 00000058 |   |    |
| MSUSL_STATE     |   | 00000040 |   |    |
| MSUSL_UNIT_VEC  |   | 00000090 |   |    |
| MSUSL_UMB_LIST  |   | 00000080 |   |    |
| MSUSL_WRITE     |   | 00000238 |   |    |
| MSUSQ_CTRL_ID   |   | 00000190 |   |    |
| MSUSQ_CTRL_INFO |   | 00000198 |   |    |
| MSUSQ_HQBLIST   |   | 000004B4 |   |    |
| MSUSQ_QUEUE     |   | 00000070 |   |    |
| MSUST_NAME      |   | 00000014 |   |    |
| MSUSV_START     | = | 0000001F |   |    |
| MSUSW_INI_HOST  |   | 0000004C |   |    |
| MSUSW_INI_PKT   |   | 00000046 |   |    |
| MSUSW_MAX_PKT   |   | 0000004A |   |    |
| MSUSW_MAX_QUE   |   | 00000062 |   |    |
| MSUSW_MIN_PKT   |   | 0000004A |   |    |
| MSUSW_NUM_HOST  |   | 0000004E |   |    |
| MSUSW_NUM_PKT   |   | 00000048 |   |    |
| MSUSW_NUM_QUE   |   | 00000060 |   |    |
| MSUSW_PACKET    |   | 00000044 |   |    |
| MSUSW_SIZE      |   | 00000008 |   |    |
| NEW_UMB         |   | 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_UMB         |   | 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 |   |    |

MSCP  
Symbol table

- MSCP Server - Emulator

C 4

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 96  
(1)

|                   |             |    |    |
|-------------------|-------------|----|----|
| PACKET            | 00000044    | R  | 02 |
| PATCH             | 00001DA0    | R  | 02 |
| PDT\$B_PDT_TYPE   | = 00000007  |    |    |
| PDT\$C_PS         | = 00000004  |    |    |
| PDT\$SL_ALLOCMMSG | = 00000014  |    |    |
| PDT\$SL_DEALLOMSG | = 00000020  |    |    |
| PDT\$SL_DEALRGMSG | = 00000024  |    |    |
| PDT\$SL_MAPIRP    | = 00000034  |    |    |
| PDT\$SL_RCLMSGBUF | = 00000048  |    |    |
| PDT\$SL_REJECT    | = 0000004C  |    |    |
| PDT\$SL_REQDATA   | = 00000050  |    |    |
| PDT\$SL_SENDDATA  | = 00000054  |    |    |
| PDT\$SL_SNDCNTMSG | = 00000060  |    |    |
| PDT\$SL_UNMAP     | = 00000064  |    |    |
| PM\$SEND_RQ       | *****       | X  | 02 |
| PM\$SGL_IOPFMPDB  | *****       | X  | 02 |
| PM\$SGL_IOPFMSEQ  | *****       | X  | 02 |
| PM\$START_RQ      | *****       | X  | 02 |
| PR\$_IPL          | = 00000012  |    |    |
| PRI               | = 00000018  |    |    |
| PRMSW             | = 00000001  |    |    |
| PRT\$C_ER         | = 00000007  |    |    |
| PRT\$C_EW         | = 00000005  |    |    |
| QUEUES            | 00000070    | R  | 02 |
| QUE_NUM           | = 00000003  |    |    |
| READ              | 000013E1    | R  | 02 |
| READ_CONT         | 0000140B    | R  | 02 |
| READ_ERR          | 0000156F    | R  | 02 |
| READ_INVALID_CDT  | 00001407    | R  | 02 |
| READ_LOOP         | 000013F4    | R  | 02 |
| READ_REPAIR       | 000015AF    | R  | 02 |
| RED_OQB           | = 00000028  |    |    |
| REMOVE            | 00000575    | R  | 02 |
| REPLACE           | 000013A2    | R  | 02 |
| RSBX              | 00000669    | R  | 02 |
| SC\$ACCEPT        | *****       | GX | 02 |
| SC\$ALLOC_RSPID   | *****       | X  | 02 |
| SC\$DEALL_RSPID   | *****       | X  | 02 |
| SC\$DISCONNECT    | *****       | X  | 02 |
| SC\$GB_SYSTEMID   | *****       | X  | 02 |
| SC\$SGL_MSCP      | *****       | X  | 02 |
| SC\$SLISTEN       | *****       | GX | 02 |
| SC\$SLKP_MSGWAIT  | *****       | X  | 02 |
| SC\$SLKP_RDTCDRP  | *****       | X  | 02 |
| SC\$SLKP_RDTWAIT  | *****       | X  | 02 |
| SC\$SL_DST_CONID  | = FFFFFFFF8 |    |    |
| SEARCH_QUE        | 00001B38    | R  | 02 |
| SELECT_FIRST_UNIT | 00001A3D    | R  | 02 |
| SELECT_NEXT_UNIT  | 00001A34    | R  | 02 |
| SEND_ATTEN        | 00000508    | R  | 02 |
| SEND_END          | 00000AED    | R  | 02 |
| SEND_MISMATCH     | 00000A9C    | R  | 02 |
| SEND_PKT          | 00000AD4    | R  | 02 |
| SEND_PKT_STS      | 00000ADE    | R  | 02 |
| SEND_SUCCESS      | 00000AD2    | R  | 02 |
| SEQ               | 00000BA0    | R  | 02 |
| SERV_INFO         | 00001CEC    | R  | 02 |

|                    |            |   |    |
|--------------------|------------|---|----|
| SERV_NAME          | 00001CEC   | R | 02 |
| SETUP              | 000017BF   | R | 02 |
| SETUP_ERR          | 00001A52   | R | 02 |
| SETUP_OUT          | 00001936   | R | 02 |
| SETUP_SHADOW       | 0000193F   | R | 02 |
| SETUP_SHADOW_READ  | 00001A1F   | R | 02 |
| SETUP_SHADOW_WRITE | 00001948   | R | 02 |
| SETX               | 00000C9D   | R | 02 |
| SET_CTRL_CHR       | 00000AA0   | R | 02 |
| SET_UNIT_CHR       | 000012FC   | R | 02 |
| SLVSK_SDATA        | = 00000004 |   |    |
| SMA                | = 0000001C |   |    |
| SMALL              | 00000058   | R | 02 |
| SS                 | 00000A11   | R | 02 |
| SS\$BADPARAM       | = 00000014 |   |    |
| SS\$CTRLERR        | = 00000054 |   |    |
| SS\$DATACHECK      | = 0000005C |   |    |
| SS\$FORCEDERROR    | = 00002144 |   |    |
| SS\$FORMAT         | = 000000BC |   |    |
| SS\$ILLIOFUNC      | = 000000F4 |   |    |
| SS\$ILLSEQOP       | = 000002DC |   |    |
| SS\$INSFMEM        | = 00000124 |   |    |
| SS\$IVBUFLN        | = 0000034C |   |    |
| SS\$MEDOFL         | = 000001A4 |   |    |
| SS\$NORMAL         | = 00000001 |   |    |
| SS\$PARITY         | = 000001F4 |   |    |
| SS\$SHACHASTA      | = 00002284 |   |    |
| SS\$TIMEOUT        | = 0000022C |   |    |
| SS\$VOLINV         | = 00000254 |   |    |
| SS\$WRITLCK        | = 0000025C |   |    |
| START              | 000004EC   | R | 02 |
| START_CONT         | 00001E28   | R | 02 |
| STATE              | 00000040   | R | 02 |
| STOP               | 0000065E   | R | 02 |
| ST_BLK_QUE         | 00000752   | R | 02 |
| ST_DAT_WAIT        | 0000070E   | R | 02 |
| ST_DRV_WAIT        | 000006F4   | R | 02 |
| ST_MAP_WAIT        | 0000073F   | R | 02 |
| ST_MEM_WAIT        | 0000073B   | R | 02 |
| ST_MSG_WAIT        | 0000073F   | R | 02 |
| ST_ONBCK_QUE       | 00000765   | R | 02 |
| ST_SHQ_QUE         | 00000744   | R | 02 |
| TIM                | = 00000014 |   |    |
| TST_END            | 00001445   | R | 02 |
| UCB\$B_ONLCNT      | = 000000AE |   |    |
| UCB\$B_SECTORS     | = 00000044 |   |    |
| UCB\$B_TRACKS      | = 00000045 |   |    |
| UCB\$B_DDT         | = 00000088 |   |    |
| UCB\$B_DEVCHAR     | = 00000038 |   |    |
| UCB\$B_MAXBLOCK    | = 000000B0 |   |    |
| UCB\$B_STS         | = 00000064 |   |    |
| UCB\$V_MNTVERIP    | = 0000000E |   |    |
| UCB\$W_QLEN        | = 0000006A |   |    |
| UNIT_VEC           | 00000090   | R | 02 |
| UQB\$B_CMD_PEND    | 0000000D   |   |    |
| UQB\$B_FLAGS       | 0000002B   |   |    |
| UQB\$B_HOST_CNT    | 0000000C   |   |    |

MSCP  
Symbol table

- MSCP Server - Emulator

D 4

16-SEP-1984 02:00:53 VAX/VMS Macro V04-00  
7-SEP-1984 17:14:02 [MSCP.SRC]MSCP.MAR;2

Page 97  
(1)

|                 |            |   |    |
|-----------------|------------|---|----|
| UQBSB_SLOT      | 0000002A   |   |    |
| UQBSB_TYPE      | 0000000A   |   |    |
| UQBSB_LEN       | 0000009C   |   |    |
| UQBSK_LEN       | 0000009C   |   |    |
| UQBSL_CDRP_BL   | 00000064   |   |    |
| UQBSL_CDRP_FL   | 00000060   |   |    |
| UQBSL_CPY_CDRP  | 00000088   |   |    |
| UQBSL_FENCEH    | 0000005C   |   |    |
| UQBSL_FENCEL    | 00000058   |   |    |
| UQBSL_HOST_ONLN | 00000034   |   |    |
| UQBSL_MAXBLOCK  | 00000048   |   |    |
| UQBSL_MEDIA_ID  | 00000020   |   |    |
| UQBSL_SHDW_BL   | 00000004   |   |    |
| UQBSL_SHDW_FL   | 00000000   |   |    |
| UQBSL_SHDW_LOW  | 00000078   |   |    |
| UQBSL_SHDW_MST  | 00000078   |   |    |
| UQBSL_START_LBN | 0000004C   |   |    |
| UQBSL_UCB       | 00000030   |   |    |
| UQBSL_VOL_SER   | 0000002C   |   |    |
| UQBSM_2PASS     | = 00000004 |   |    |
| UQBSM_BLOCKED   | = 00000002 |   |    |
| UQBSM_CIP       | = 00000001 |   |    |
| UQBSM_CMDPEND   | = 00000008 |   |    |
| UQBSM_RIP       | = 00000020 |   |    |
| UQBSM_SEQ       | = 00000010 |   |    |
| UQBSQ_BLOCKQ    | 00000068   |   |    |
| UQBSQ_SHQ       | 00000070   |   |    |
| UQBSQ_UNITQ     | 00000080   |   |    |
| UQBSQ_UNIT_ID   | 00000018   |   |    |
| UQBST_NAME      | 00000038   |   |    |
| UQBSV_2PASS     | = 00000002 |   |    |
| UQBSV_CIP       | = 00000000 |   |    |
| UQBSV_CMDPEND   | = 00000003 |   |    |
| UQBSV_RIP       | = 00000005 |   |    |
| UQBSW_MAX_BLK   | 0000007E   |   |    |
| UQBSW_MAX_OPS   | 00000056   |   |    |
| UQBSW_MAX_QUE   | 00000052   |   |    |
| UQBSW_MULT_UNT  | 00000010   |   |    |
| UQBSW_NUM_BLK   | 0000007C   |   |    |
| UQBSW_NUM_OPS   | 00000054   |   |    |
| UQBSW_NUM_QUE   | 00000050   |   |    |
| UQBSW_SHDW_STS  | 00000026   |   |    |
| UQBSW_SHDW_UNT  | 00000024   |   |    |
| UQBSW_SIZE      | 00000008   |   |    |
| UQBSW_STATUS    | 0000000E   |   |    |
| UQBSW_UNIT      | 00000028   |   |    |
| UQBSW_UNT_FLGS  | 00000012   |   |    |
| UQB_LIST        | 00000080   | R | 02 |
| VASS_BYTE       | = 00000009 |   |    |
| VASS_VPN        | = 00000015 |   |    |
| VASV_VPN        | = 00000009 |   |    |
| VC_ERR          | 0000006A   | R | 02 |
| VERSION         | 00000198   | R | 02 |
| WAIT_FOR_INT    | 00000863   | R | 02 |
| WRITE           | 000016EE   | R | 02 |
| WRITE_ERR       | 00001787   | R | 02 |
| WRT_UBB         | = 00000024 |   |    |

M  
V

-----  
! Psect synopsis !  
-----

| PSECT name | Allocation        | PSECT No. | Attributes                                              |
|------------|-------------------|-----------|---------------------------------------------------------|
| . ABS .    | 00000000 ( 0.)    | 00 ( 0.)  | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| \$ABSS     | 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+EXECMLS/LIB

0249

AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY