```
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDDDDDDDDDDD     RRRRRRRRRRRR        IIIIIIIII     VVV         VVV  EEEEEEEEEEEEEEE  RRRRRRRRRRRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRR          RRR       III       VVV         VVV  EEE              RRR          RRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III       VVV         VVV  EEEEEEEEEEE      RRRRRRRRRRRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR   RRR              III       VVV         VVV  EEE              RRR   RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDD          DDD RRR      RRR           III          VVV   VVV     EEE              RRR      RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
DDDDDDDDDDDD     RRR         RRR     IIIIIIIII          VVV       EEEEEEEEEEEEEEE  RRR         RRR
```

```
TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
TTTTTTTTTT  UU      UU  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  EEEEEEEEEE  RRRRRRRR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RR      RR      II      VV      VV  EE          RR      RR
    TT      UU      UU  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    TT      UU      UU  DD      DD  RRRRRRRR        II      VV      VV  EEEEEEEE    RRRRRRRR
    TT      UU      UU  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
    TT      UU      UU  DD      DD  RR  RR          II      VV      VV  EE          RR  RR
    TT      UU      UU  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
    TT      UU      UU  DD      DD  RR    RR        II        VV  VV    EE          RR    RR        ....
    TT      UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII      VV      EEEEEEEEEE  RR      RR      ....
    TT      UUUUUUUUUU  DDDDDDDD    RR      RR    IIIIII      VV      EEEEEEEEEE  RR      RR      ....

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

```
0000     1          .TITLE  TUDRIVER - TAPE CLASS DRIVER
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;****************************************************************************
0000     6  ;*                                                                          *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     9  ;*  ALL RIGHTS RESERVED.                                                    *
0000    10  ;*                                                                          *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16  ;*  TRANSFERRED.                                                            *
0000    17  ;*                                                                          *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20  ;*  CORPORATION.                                                            *
0000    21  ;*                                                                          *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    24  ;*                                                                          *
0000    25  ;*                                                                          *
0000    26  ;****************************************************************************
0000    27  ;
0000    28  ; Robert Rappaport  16-June-1982
0000    29  ;
0000    30  ;  TAPE CLASS DRIVER
0000    31  ;
0000    32  ; MODIFIED BY:
0000    33  ;
0000    34  ;       V03-161 ROW0398         Ralph O. Weber          21-JUL-1984
0000    35  ;               Setup use of class driver write-lock bit in UCB$W_DEVSTS.
0000    36  ;
0000    37  ;       V03-160 ROW0396         Ralph O. Weber          21-JUL-1984
0000    38  ;               Setup automatic detection of density after an operation which
0000    39  ;               moves the tape position off of the BOT.
0000    40  ;
0000    41  ;       V03-159 ROW0395         Ralph O. Weber          21-JUL-1984
0000    42  ;               Make changes which setup "normal" MSCP command timeout
0000    43  ;               algorithm before calls to DUTU$POLL_FOR_UNITS and
0000    44  ;               BRING_UNIT_ONLINE.  Also setup use of DAP CDRP by both
0000    45  ;               DUTU$POLL_FOR_UNITS and BRING_UNIT_ONLINE.
0000    46  ;
0000    47  ;       V03-158 ROW0394         Ralph O. Weber          20-JUL-1984
0000    48  ;               Remove DPT_STORE setting of ACL queue present bit in the ORB.
0000    49  ;               This should improve performance on devices which do not really
0000    50  ;               have an ACL queue in their device protection ORB.
0000    51  ;
0000    52  ;       V03-157 ROW0393         Ralph O. Weber          20-JUL-1984
0000    53  ;               Add media-id to device type translation table entries for the
0000    54  ;               TA78, TK50, and TA81.
0000    55  ;
0000    56  ;       V03-156 ROW0387         Ralph O. Weber           8-JUL-1984
0000    57  ;               Setup use of DUTU$RECONN_LOOKUP and DUTU$DRAIN_CDDB_CDRPQ.
```

```
0000     58 ;
0000     59 ;       V03-155 ROW0369         Ralph O. Weber          6-JUL-1984
0000     60 ;               Change DU$RE_SYNCH to not do MRESET/MSTART to MSCP servers and
0000     61 ;               then wait for something to happen.  Quite possibly, nothing
0000     62 ;               ever will happen in such cases.  Proceeding directly to the
0000     63 ;               DISCONNECT is the correct action.  This is being done now so
0000     64 ;               that it will not be forgotten when as and if we make a tape
0000     65 ;               MSCP server.
0000     66 ;
0000     67 ;       V03-154 ROW0382         Ralph O. Weber          22-JUN-1984
0000     68 ;               Change START_PACKACK so the an exclusive access online command
0000     69 ;               is sent only the multihost controllers.  For other controllers,
0000     70 ;               just sent an online.
0000     71 ;
0000     72 ;       V03-153 ROW0361         Ralph O. Weber          5-MAY-1984
0000     73 ;               Setup use of new class driver common DAP processing in
0000     74 ;               DUTU$DODAP.  The new routine is designed to eliminate multiple
0000     75 ;               concurrent DAP threads which are known to crash systems.
0000     76 ;
0000     77 ;       V03-152 ROW0354         Ralph O. Weber          30-APR-1984
0000     78 ;               Add setting for DEV$M_NNM in DEVCHAR2 to indicate that tape
0000     79 ;               class driver devices use NODENAME$DDCN device names.
0000     80 ;
0000     81 ;       V03-151 ROW0353         Ralph O. Weber          30-APR-1984
0000     82 ;               Correct message type constant input to ERL$LOGMESSAGE from
0000     83 ;               EMB$C_DM (for disks) to EMB$C_TM (for tapes).
0000     84 ;
0000     85 ;       V03-150 ROW0350         Ralph O. Weber          23-APR-1984
0000     86 ;               Correct more problems causing multiple trips through
0000     87 ;               END_SINGLE_STREAM, with the attendent bugchecks.  First, clear
0000     88 ;               CDDB$V_SNGLSTRM upon entry to DU$CONNECT_ERR.  Second, protect
0000     89 ;               the SCS$UNSTALLUCB loop in END_SINGLE_STREAM from possible
0000     90 ;               connection failures during execution of the loop.
0000     91 ;
0000     92 ;       V03-149 LMP0237         L. Mark Pilant,         19-Apr-1984  11:25
0000     93 ;               Initialize the template ORB.
0000     94 ;
0000     95 ;       V03-148 ROW0347         Ralph O. Weber          11-APR-1984
0000     96 ;               Cause MT$V_HWL to be cleared when tape is not write locked and
0000     97 ;               whenever an AVAILABLE command is sent to the server.
0000     98 ;
0000     99 ;       V03-147 ROW0339         Ralph O. Weber          9-APR-1984
0000    100 ;               Setup use of common invalid command processing routines
0000    101 ;               (macros).  This replaces the old "form the original MSCP
0000    102 ;               command packet by hand" algorithm with a "repeat the code
0000    103 ;               which formed the original MSCP command" algorithm.  The cost
0000    104 ;               is a single, hardly ever taken BLBS in the mainline read/write
0000    105 ;               code path.  The savings are elimination of having to duplicate
0000    106 ;               command packet setup changes in the invalid command case,
0000    107 ;               hundreds of bytes of code, and a not inconsequential amount of
0000    108 ;               static storage.
0000    109 ;
0000    110 ;       V03-146 ROW0338         Ralph O. Weber          7-APR-1984
0000    111 ;               Setup use of DO_ACTION macro to replace INTERPRET_ACTION_TABLE.
0000    112 ;               Start using IF_MSCP where only success or failure of an MSCP
0000    113 ;               command is being tested.  Setup use of ACTION_ENTRY END to end
0000    114 ;               action tables.  Remove action table interpretation routines;
```

```
0000    115 ;               they are now in DUTUSUBS.
0000    116 ;
0000    117 ;       V03-145 ROW0335         Ralph O. Weber          4-APR-1984
0000    118 ;               > Correct positioning of DPT_STORE REINIT and add note that
0000    119 ;                 reinit is not significant because driver is not reloadable.
0000    120 ;               > Add use of DUTU$UNITINIT.  Basicly, this permits future use
0000    121 ;                 of TMSCP devices for booting.
0000    122 ;               > Remove usage of allocation class value in the SCS connect
0000    123 ;                 accept message.  All MSCP servers now supply that
0000    124 ;                 information in the Set Controller Characteristics command
0000    125 ;                 end packet.
0000    126 ;               > Eliminate bug check for IO$_READLBLK and IO$_WRITELBLK.
0000    127 ;                 Make these functions produce SS$_ILLIOFUNC status instead.
0000    128 ;                 Also change function dispatcher to use DISPATCH macro.
0000    129 ;               > Add processing for IO$M_INHRETRY.
0000    130 ;               > Add the multi-host progress counter handling proposed by the
0000    131 ;                 HSC implementors to TU$TMR.  This algorithm simplifies
0000    132 ;                 handling of the case where the MSCP server is busy on an
0000    133 ;                 older command from another host.
0000    134 ;
0000    135 ;       V03-144 ROW0331         Ralph O. Weber          31-MAR-1984
0000    136 ;               Setup use of common cancel support in DUTUSUBS.  Also make
0000    137 ;               functions which use multiple MSCP commands check for cancel
0000    138 ;               after each MSCP command and perform cancel if necessary.
0000    139 ;
0000    140 ;       V03-143 ROW0328         Ralph O. Weber          21-MAR-1984
0000    141 ;               Correct bugs in ROW0319 which caused it to incorrectly miss
0000    142 ;               the end of the CDDB UCB chain.
0000    143 ;
0000    144 ;       V03-142 ROW0324         Ralph O. Weber          12-MAR-1984
0000    145 ;               > Correct set mode and set characteristics so that
0000    146 ;                 MSCP$W_FORMAT is zero except when the UCB$L_RECORD is zero.
0000    147 ;                 This brings the driver into conformance with TMSCP
0000    148 ;                 version 1.6.
0000    149 ;               > Provide for proper setup of the following UCB$L_DEVDEPEND
0000    150 ;                 bits in all cases that I can think of:  MT$V_BOT, MT$V_EOF,
0000    151 ;                 MT$V_EOT, MT$V_HWL, MT$V_LOST, MT$V_SUP_NRZI, MT$V_SUP_PE,
0000    152 ;                 and MT$V_SUP_GCR.
0000    153 ;               > Fix "detect EOT" modifier setup so that the modifier is
0000    154 ;                 NEVER set for physical I/O requests.
0000    155 ;               > Change IOSB status returned when a backwards skip file
0000    156 ;                 encounters the BOT to SS$_NORMAL.
0000    157 ;
0000    158 ;       V03-141 ROW0320         Ralph O. Weber          29-FEB-1984
0000    159 ;               Provide for automatic PACKACK on foreign tapes (DEV$V_FOR set)
0000    160 ;               whenever a request is received and the UCB$V_VALID bit is
0000    161 ;               clear.  Build the sequential NOP function into macros so that
0000    162 ;               its use can be easily duplicated where necessary.
0000    163 ;
0000    164 ;       V03-140 ROW0319         Ralph O. Weber          28-FEB-1984
0000    165 ;               Attempt to eliminate failover to non-operational path by
0000    166 ;               making clearing of CDDB$V_RECONNECT the last thing done in
0000    167 ;               END_SINGLE_STREAM.  Also add sanity check that CDDB$V_RECONNECT
0000    168 ;               is set before it is cleared.
0000    169 ;
0000    170 ;       V03-139 ROW0310         Ralph O. Weber          23-FEB-1984
0000    171 ;               Make IO$_REWINDOFF equivalent to IO$_UNLOAD.
```

```
0000   172 ;
0000   173 ;       V03-138 ROW0307         Ralph O. Weber          15-FEB-1984
0000   174 ;               Fix trace support to work in the common modules environment.
0000   175 ;               Make RECORD_GETUNIT_CHAR preserve R0.
0000   176 ;
0000   177 ;       V03-137 ROW0305         Ralph O. Weber          13-FEB-1984
0000   178 ;               Fix R0 (final IOSB status) corruption problems in successful
0000   179 ;               IO$_PACKACK processing.
0000   180 ;
0000   181 ;       V03-136 ROW0301         Ralph O. Weber          10-FEB-1984
0000   182 ;               Move clearing of CDDB$V_NOCONN from MAKE_CONNECTION to after
0000   183 ;               the new connection information has been propogated to all UCBs
0000   184 ;               in the re-connect code.  While this is not absolutely
0000   185 ;               necessary here and now, it will provide a useful reminder that
0000   186 ;               CDDB$V_NOCONN set blocks mount verification attempts and thus
0000   187 ;               the bit cannot be cleared until connection dependent fields in
0000   188 ;               all UCBs have been altered to reflect the new connection.
0000   189 ;
0000   190 ;       V03-135 ROW0299K(ludge) Ralph O. Weber           9-FEB-1984
0000   191 ;               This kludge detects a HSC tape server in RECORD_STCON and
0000   192 ;               forces it to act like a multihost server for allocation class
0000   193 ;               determination, inspite of the fact that the HSC tape server
0000   194 ;               does not set the multihost controller flag.  This kludge can
0000   195 ;               be removed when the HSC tape server sets the multihost
0000   196 ;               controller flag (as it should).
0000   197 ;
0000   198 ;       V03-134 ROW0298         Ralph O. Weber          9-FEB-1984
0000   199 ;               Setup use of CDRP$W_ENDMSGSIZ to hold the size of an incomming
0000   200 ;               sequenced message.  This replaces use of CDRP$L_IOST2+2 whose
0000   201 ;               use causes valuable input information to be overwritten.
0000   202 ;
0000   203 ;       V03-133 ROW0297         Ralph O. Weber          7-FEB-1984
0000   204 ;               Correct confusion between wait count bumped due to a broken
0000   205 ;               connection and wait count bumped due to a sequential NOP by
0000   206 ;               introducing a UCB$V_TU_SETNOP bit in device dependent status.
0000   207 ;
0000   208 ;       V03-132 ROW0294         Ralph O. Weber          5-FEB-1984
0000   209 ;               Correct RECORD_STCON setup of allocation class information in
0000   210 ;               the DDBs to use DDB$L_CONLINK so that only those DDBs on this
0000   211 ;               connection are effected.
0000   212 ;
0000   213 ;       V03-131 ROW0293         Ralph O. Weber          5-FEB-1984
0000   214 ;               Generally bring tape class driver to same revision level as
0000   215 ;               disk class driver.  The only exception is that there is no
0000   216 ;               mount verification and thus thing which depend upon it for
0000   217 ;               updated operation techniques have been left unchanged.
0000   218 ;               Replace CDRP$V_ERLOGIP in CDRP$W_STS with CDRP$V_ERLIP in
0000   219 ;               CDRP$L_DUTUFLAGS.  Setup use of CDDB$V_NOCONN status bit.
0000   220 ;               Setup use of several routines which have been moved to
0000   221 ;               DUTUSUBS.
0000   222 ;
0000   223 ;       V03-130 ROW0272         Ralph O. Weber          1-JAN-1984
0000   224 ;               Change START_DAP_THREAD to only send Determin Access Paths
0000   225 ;               commands for those UCBs which are UCB$V_VALID.  MSCP servers
0000   226 ;               will ignore DAP commands for units which are not MSCP online,
0000   227 ;               so why should we send them.  Add block which prevents logging
0000   228 ;               errors for DAP attention messages to ACCESS_PATH_ATTN.  This
```

```
0000   229 ;          allows the code which logs DAP attention messages to remain
0000   230 ;          and to be patched back into existance should it be needed.
0000   231 ;
0000   232 ;      V03-129 ROW0270         Ralph O. Weber          1-JAN-1984
0000   233 ;          Eliminate DRIVER_SEND_MSG_BUF by replacing all calls to it
0000   234 ;          with SEND_MSCP_MSG_DRIVER.  Change MAKE_CONNECTION to use the
0000   235 ;          larger of HSTIMEOUT_ARRAY[controller_model] and the controller
0000   236 ;          timeout value as the final host timeout value for the MSCP Set
0000   237 ;          Controller Characteristics command.  Setup use of VMS SCS
0000   238 ;          RECYCL_RSPID and FIND_RSPID_RDTE.  Fix START_SENSECHAR and
0000   239 ;          START_SENSEMODE to clear the MSCP$M_MD_CLSEX (clear serious
0000   240 ;          exception modifier) bit, as this modifier is illegal on Get
0000   241 ;          Unit Status commands.  Make all permanent/DAP CDRP to CDDB
0000   242 ;          conversions use PERMCDRP_TO_CDDB.
0000   243 ;
0000   244 ;      V03-128 ROW0269         Ralph O. Weber          1-JAN-1984
0000   245 ;          Change DU_CONTROLLER_INIT to use DUTU$CREATE_CDDB.
0000   246 ;
0000   247 ;      V03-127 ROW0262         Ralph O. Weber          27-DEC-1983
0000   248 ;          Move all UCB lookup and creation to DUTUSUBS.  Cleanup
0000   249 ;          ATTN_MSG processing in TU$IDR.  Implement usage of $DUTUDEF,
0000   250 ;          all device independent UCB fields, and the IOC$GL_TU_CDDB
0000   251 ;          listhead.  Replace all DPT_STORE macros which init UCB fields
0000   252 ;          with INIT_UCB macros.  INIT_UCB initializes both the DPT and
0000   253 ;          the template UCB.  Its use eliminates possible mismatch of the
0000   254 ;          two UCB sources as well as some setup code in the controller
0000   255 ;          initialization routine.  Make driver not reloadable.  Change
0000   256 ;          POLL_FOR_UNITS to DUTU$POLL_FOR_UNITS.
0000   257 ;
0000   258 ;      V03-126 ROW0261         Ralph O. Weber          22-NOV-1983
0000   259 ;          Move DUMP_COMMAND and DUMP_ENDMESSAGE to DUTUSUBS.  Change
0000   260 ;          TU$END to DUTU$END so that linking with multiple modules does
0000   261 ;          not involve a hack.  Do some common path cleanup to speed
0000   262 ;          passage through the common code paths.  Change subroutine
0000   263 ;          CALL_SEND_MSG_BUF to SEND_MSCP_MSG macro.  Move INIT_TPLATE_UCB
0000   264 ;          to DUTULIB (macro library).
0000   265 ;
0000   266 ;      V03-125 RLRQBUS         Robert L. Rappaport     16-NOV-1983
0000   267 ;          Change building of transfer commands MSCP packet so that
0000   268 ;          PQDRIVER can alter the mapping information during a map
0000   269 ;          request and have the altered information appear in the MSCP
0000   270 ;          packet.
0000   271 ;
0000   272 ;      V03-124 ROW0258         Ralph O. Weber          17-NOV-1983
0000   273 ;                      The Paul Painter Memorial Enhancement
0000   274 ;          Named for one of the unfortunate customers who suffered much
0000   275 ;          to determine the great UCB$L_MT_RECORD secret while trying to
0000   276 ;          create a user-written magtape driver, this change eliminates
0000   277 ;          use of the device dependent field, UCB$L_TU_RECORD in favor of
0000   278 ;          the device independent field, UCB$L_RECORD.
0000   279 ;
0000   280 ;      V03-123 ROW0253         Ralph O. Weber          12-NOV-1983
0000   281 ;          Change device dependent UCB definitions to work with globally
0000   282 ;          defined MSCP extension to the UCB.  This change does not make
0000   283 ;          use of all the UCB fields in the new extension.  It simply
0000   284 ;          eliminates interactions which will prevent this module from
0000   285 ;          building in the presence of the new UCB definitions.  The
```

```
0000   286 ;
0000   287 ;               UCB$L_TU_MEDIATYP field, which was changed to UCB$L_MEDIA_ID
0000   288 ;               ages ago, has also been eliminated.  NB: a gross hack has been
0000   289 ;               employed to keep this driver compatible with the other magtape
0000   290 ;               drivers and the magtape ACP.  This will be corrected when all
0000   291 ;               the involved parties start using the newly defined
0000   292 ;               UCB$L_RECORD.
0000   293 ;
0000   294 ;       V03-122 ROW0245         Ralph O. Weber          19-OCT-1983
0000   295 ;               Correct couple of outstanding bugs:
0000   296 ;                - Change TU$IDR to store incomming message size in
0000   297 ;                  CDRP$L_IOST2+2.  This provides the message size to any code
0000   298 ;                  requiring it.  In particular, the INVALID_STS fixes
0000   299 ;                  mentioned below use this feature.
0000   300 ;                - Fix INVALID_STS to properly place the size of the incomming
0000   301 ;                  MSCP message in R1 before calling ERL$LOG_DMSCP.
0000   302 ;
0000   303 ;       V03-121 ROW0243         Ralph O. Weber          17-OCT-1983
0000   304 ;               Enhance SEQ_ENDCHECK to allow canceled (MSCP aborted) end
0000   305 ;               packets to be received out of sequence.  This produces
0000   306 ;               conformance to a revised version of the TMSCP specification.
0000   307 ;
0000   308 ;       V03-120 ROW0242         Ralph O. Weber          17-OCT-1983
0000   309 ;               Change unit attention processing in DU$IDR to skip altering
0000   310 ;               UCB$M_DU_WAITBMP and UCB$W_RWAITCNT when the CDDB$M_INITING or
0000   311 ;               CDDB$M_RECONNECT is set in CDDB$W_STATUS.  This prevents
0000   312 ;               altering the wait count is such a way that the wait count
0000   313 ;               tests in controller init and reconnection processing fail.
0000   314 ;               Therefore, a spurous disk class driver bugcheck is eliminated.
0000   315 ;
0000   316 ;       V03-119 BLS0234         Benn Schreiber          9-Aug-1983
0000   317 ;               Add missing G^s to calls in exec.
0000   318 ;
0000   319 ;       V03-118 RLRDLATE        Robert L. Rappaport     25-Jul-1983
0000   320 ;               Check for Data Late subcode in Controller Errors on
0000   321 ;               data transfer commands, and return SS$_DATALATE.
0000   322 ;
0000   323 ;       V03-117 RLRDLEOT        Robert L. Rappaport     19-Jul-1983
0000   324 ;               Implement suport for new MSCP$M_MD_DLEOT modifier.
0000   325 ;               Modifier means "Detect Logical End Of Tape" and is
0000   326 ;               used on QIO Skip files and Skip records (forward
0000   327 ;               direction only).
0000   328 ;
0000   329 ;       V03-116 RLRIMMED        Robert L. Rappaport     19-Jul-1983
0000   330 ;               Implement support for new MSCP$M_MD_IMMED modifier
0000   331 ;               that allows us to express that certain commands,
0000   332 ;               namely REWIND and DSE, are to return their End Messages
0000   333 ;               when the command BEGINS to execute rather than when it
0000   334 ;               completes.  A discussion of this is found in the TMSCP
0000   335 ;               spec under "Synchronous versus Asynchronous" operation
0000   336 ;               of lengthy commands.
0000   337 ;
0000   338 ;               The effort here consists of simplifying greatly the
0000   339 ;               previous method of implementing support for IO$M_NOWAIT.
0000   340 ;               This simplification eliminates the need for a REWIND
0000   341 ;               CDRP, as well as the need for special handling of
0000   342 ;               Rewind and Available (UNLOAD) requests.
```

```
0000   343 ;
0000   344 ;               This update almost completely obviates those changes
0000   345 ;               implemented as a result of update RLRRWATN.
0000   346 ;
0000   347 ;               Also in this update fix bug in START_SETCHAR wherein we
0000   348 ;               neglected to call SCS$UNSTALLUCB after decrementing
0000   349 ;               UCB$W_RWAITCNT.
0000   350 ;
0000   351 ;       V03-115 RLRUPTODATE     Robert L. Rappaport     26-Jul-1983
0000   352 ;               Adapt and incorporate relevant changes from Disk
0000   353 ;               Class Driver.  From ;RLRDDB audit of DUDRIVER
0000   354 ;               thru ;RLRODDBCNT.
0000   355 ;
0000   356 ;       V03-114 RLRGROWTH       Robert L. Rappaport     23-Jun-1983
0000   357 ;               Due to growth in the CDDB, the length of the CDDB plus
0000   358 ;               the length of the CDRP is NOT < 256.  We must change
0000   359 ;               a MOVZBL to a MOVZWL.
0000   360 ;
0000   361 ;       V03-113 RLRDPATH2       Robert L. Rappaport     31-May-1983
0000   362 ;               As a result of the previous change (RLRDPATH1),
0000   363 ;               UCB$L_TU_RECORD has moved with respect to UCB$L_DPC
0000   364 ;               breaking an assume statement that must now be fixed.
0000   365 ;
0000   366 ;       V03-112 RLRDPATH1       Robert L. Rappaport     25-May-1983
0000   367 ;               Allow UCB to include new DUAL PORT extension by
0000   368 ;               changing base of where we begin the private TUDRIVER
0000   369 ;               extension from UCB$L_DPC+4 to UCB$L_DP_LINK+4.
0000   370 ;
0000   371 ;       V03-111 RLRRWCPTRa      Robert L. Rappaport     11-Apr-1983
0000   372 ;               Correct bug in RLRRWCPTR fix.
0000   373 ;
0000   374 ;       V03-110 RLRCANCELf      Robert L. Rappaport     11-Apr-1983
0000   375 ;               Initialize CDRP fields before deciding whether to start
0000   376 ;               this I/O request or whether to Q to UCB I/O Queue.  This
0000   377 ;               prevents misinterpreting uninitialized fields.
0000   378 ;
0000   379 ;       V03-109 RLRRWCPTR       Robert L. Rappaport     4-Mar-1983
0000   380 ;               Test for zero UCB$L_RWCPTR in RDTWAIT_DIS_ACT and
0000   381 ;               in RDT_DIS_ACTION.  Such a situation could occur if
0000   382 ;               no RSPID's were available during a re-Connection and
0000   383 ;               if the re-Connection failed and we had to do a
0000   384 ;               re-re-Connection.  Also use Controller timeout for
0000   385 ;               host timeout value for those controllers for which
0000   386 ;               we care to set a host timeout.  Also only use INIT_IMMED_DELTA
0000   387 ;               for timing out the first SET_CONTROLLER_CHAR command. After-
0000   388 ;               words always use CDDB$W_CNTRLTMO.  Also increase
0000   389 ;               INIT_IMMED_DELTA to 30.
0000   390 ;
0000   391 ;       V03-108 RLRTMUCB        Robert L. Rappaport     25-Feb-1983
0000   392 ;               Revamp Template UCB so as to be automatically compliant
0000   393 ;               with new UCB additions.  Also remove initial Breakpoint.
0000   394 ;
0000   395 ;       V03-107 RLRWTMPOS       Robert L. Rappaport     22-Feb-1983
0000   396 ;               Update UCB$L_TU_POSITION after error on WRITE TAPE MARK
0000   397 ;               command.
0000   398 ;
0000   399 ;       V03-106 RLRSEQNOP       Robert L. Rappaport     15-Feb-1983
                        Use REPOSITION command with zeroes as a sequential NOP
```

```
0000   400 ;                in SET CHAR and SET MODE processing.
0000   401 ;
0000   402 ;        V03-105 RLRWRTM         Robert L. Rappaport     14-Feb-1983
0000   403 ;                Accept MSCP$K_ST_DATA as possible status of Write Tape Mark.
0000   404 ;
0000   405 ;        V03-104 RLRRWATN        Robert L. Rappaport     11-Feb-1983
0000   406 ;                Implement REWIND ATTENTION and NOWAIT.  Also add
0000   407 ;                support for REWIND Attention messages received as a
0000   408 ;                AVAILABLE and UNLOAD commands.  Also support ignoring
0000   409 ;                of spurious REWIND Attention messages.
0000   410 ;
0000   411 ;        V03-103 RLRTRACE        Robert L. Rappaport     4-Feb-1983
0000   412 ;                Make IRP trace a per unit rather than a per system
0000   413 ;                structure by moving it to the UCB.
0000   414 ;
0000   415 ; MACRO LIBRARY CALLS
0000   416 ;
0000   417
0000   418          $CDDBDEF                        ;Define CDDB offsets
0000   419          $CDRPDEF                        ;Define CDRP offsets
0000   420          $CDTDEF                         ;Define CDT offsets
0000   421          $CRBDEF                         ;Define CRB offsets
0000   422          $DCDEF                          ;Define Device Classes and Types
0000   423          $DDBDEF                         ;Define DDB offsets
0000   424          $DEVDEF                         ;Define DEVICE CHARACTERISTICS bits
0000   425          $DPTDEF                         ;Define DPT offsets
0000   426          $DYNDEF                         ;Define DYN symbols
0000   427          $EMBLTDEF                       ;Define EMB Log Message Types
0000   428          $FKBDEF                         ;Define FKB offsets
0000   429          $IDBDEF                         ;Define IDB offsets
0000   430          $IODEF                          ;Define I/O FUNCTION codes
0000   431          $IPLDEF                         ;Define symbolic IPL's
0000   432          $IRPDEF                         ;Define IRP offsets
0000   433          $MSCPDEF                        ;Define MSCP packet offsets
0000   434          $MSLGDEF                        ;Define MSCP Error Log offsets
0000   435          $MTDEF                          ;Define MAGTAPE STATUS bits
0000   436          $ORBDEF                         ;Define ORB offsets
0000   437          $PBDEF                          ;Define Path Block offsets
0000   438          $PCBDEF                         ;Define PCB offsets
0000   439          $PDTDEF                         ;Define PDT offsets
0000   440          $PRDEF                          ;Define Processor Registers
0000   441          $SBDEF                          ;Define System Block Offsets
0000   442          $SCSCMGDEF                      ;Define SCS Connect Message offsets
0000   443          $RCTDEF                         ;Define RCT offsets
0000   444          $RDDEF                          ;Define RDTE offsets
0000   445          $RDTDEF                         ;Define RDT offsets
0000   446          $SSDEF                          ;Define System Status values
0000   447          $UCBDEF                         ;Define UCB offsets
0000   448          $VADEF                          ;Define Virtual Address offsets
0000   449          $VECDEF                         ;Define INTERRUPT DISPATCH VECTOR offsets
0000   450          $WCBDEF                         ;Define WCB offsets
0000   451
0000   452
0000   453          $DUTUDEF                        ;Define common class driver CDDB
0000   454                                          ; extensions and other common symbols
0000   455
0000   456
```

```
                     0000   457 ; Constants
                     0000   458
00000001             0000   459 ALLOC_DELTA=1                    ; Number of seconds to wait to retry pool
                     0000   460                                  ;  allocation that failed.
0000001E             0000   461 INIT_IMMED_DELTA=30               ; During Controller Initialization, the
                     0000   462                                  ;  timeout DELTA for immediate MSCP commands.
0000000A             0000   463 CONNECT_DELTA=10                  ; During Controller Initialization, the
                     0000   464                                  ;  time interval for retrying failed
                     0000   465                                  ;  CONNECT attempts.
0000001E             0000   466 HOST_TIMEOUT=30                   ; Host timeout value.
                     0000   467
00000001             0000   468 DISCONNECT_REASON=1
0000000A             0000   469 INITIAL_CREDIT=10
00000002             0000   470 INITIAL_DG_COUNT=2
00000002             0000   471 MAX_RETRY=2
00000002             0000   472 MIN_SEND_CREDIT=2
```

```
0000    474          .SBTTL  MACRO DEFINITIONS
0000    475
0000    476 ;
0000    477 ; Expanded opcode macros - Branch word conditional psuedo opcodes.
0000    478 ;
0000    479
0000    480 ;
0000    481 ; BWNEQ - Branch (word offset) not equal
0000    482 ;
0000    483
0000    484          .MACRO  BWNEQ   DEST,?L1
0000    485          BEQL    L1              ; Branch around if NOT NEQ.
0000    486          BRW     DEST            ; Branch to destination if NEQ.
0000    487 L1:                              ; Around.
0000    488          .ENDM   BWNEQ
0000    489
0000    490
0000    491 ;
0000    492 ; BWEQL - Branch (word offset) equal
0000    493 ;
0000    494
0000    495          .MACRO  BWEQL   DEST,?L1
0000    496          .SHOW
0000    497          BNEQ    L1              ; Branch around if NOT EQL.
0000    498          BRW     DEST            ; Branch to destination if EQL.
0000    499 L1:                              ; Around.
0000    500          .NOSHOW
0000    501          .ENDM   BWEQL
0000    502
0000    503 ;
0000    504 ; BWBS - Branch (word offset) bit set.
0000    505 ;
0000    506
0000    507          .MACRO  BWBS    BIT,FIELD,DEST,?L1
0000    508          .SHOW
0000    509          BBC     BIT,FIELD,L1            ; Branch around if bit NOT set.
0000    510          BRW     DEST                    ; Branch to destination if bit set.
0000    511 L1:                                      ; Around.
0000    512          .NOSHOW
0000    513          .ENDM   BWBS
0000    514
0000    515 ;
0000    516 ; BWBC - Branch (word offset) bit clear.
0000    517 ;
0000    518
0000    519          .MACRO  BWBC    BIT,FIELD,DEST,?L1
0000    520          .SHOW
0000    521          BBS     BIT,FIELD,L1            ; Branch around if bit NOT clear.
0000    522          BRW     DEST                    ; Branch to destination if bit clear.
0000    523 L1:                                      ; Around.
0000    524          .NOSHOW
0000    525          .ENDM   BWBC
0000    526
0000    527          .IF     DF      TU_SEQCHK
0000    528 ;
0000    529 ; SEQFUNC - Macro included in conditional code to check sequentiality
0000    530 ;                of function terminations.
```

```
0000    531 ;
0000    532
0000    533          .MACRO  SEQFUNC CODES
0000    534 MASKL    = 0
0000    535 MASKH    = 0
0000    536          .IRP    X,<CODES>
0000    537          .IF     GT      <IO$_'X&IO$_VIRTUAL>-31
0000    538 MASKH    = MASKH!<1@<<IO$_'X&IO$_VIRTUAL>-32>>
0000    539          .IFF
0000    540 MASKL    = MASKL!<1@<IO$_'X&IO$_VIRTUAL>>
0000    541          .ENDC
0000    542          .ENDM
0000    543          .LONG   MASKL,MASKH
0000    544          .ENDM   SEQFUNC
0000    545          .ENDC
0000    546
0000    547 ;
0000    548 ; START_SEQNOP - macro to start a sequential NOP sequence
0000    549 ;
0000    550 ;       This macro starts a sequential NOP sequence.  A sequential NOP
0000    551 ;       sequence encapsulates a series of TMSCP operations which must occur
0000    552 ;       sequentially with respect to the stream of TMSCP operations flowing
0000    553 ;       through the driver.
0000    554 ;
0000    555 ;       First UCB$W_RWAITCNT is increased by one to prevent future I/O
0000    556 ;       requests from starting.  Then a TMSCP sequential command which does
0000    557 ;       not alter the tape position is sent to the server.  When the
0000    558 ;       sequential command completes, the driver and the server are
0000    559 ;       synchronized.
0000    560 ;
0000    561 ;       Upon exit from this macro, the currently executing thread is the only
0000    562 ;       thread conversing with the server.  When the operations which must be
0000    563 ;       done in this sychronized state are completed, the sequential NOP state
0000    564 ;       should be terminated using the END_SEQNOP macro.
0000    565 ;
0000    566 ; Inputs:
00C0    567 ;
0000    568 ;       R3      UCB address
0000    569 ;       R4      PDT address
0000    570 ;       R5      CDRP address (RSPID & message buffer already allocated and
0000    571 ;                             initialized)
0000    572 ;       (SP)    address of caller's caller
0000    573 ;
0000    574 ; Outputs:
0000    575 ;
0000    576 ;       R3 through R5 unchanged
0000    577 ;       All other registers altered
0000    578
0000    579          .MACRO  START_SEQNOP ?L1
0000    580          BBSS    #UCB$V_TU_SEQNOP,-      ; Set sequential NOP in progress and
0000    581                  UCB$W_DEVSTS(R3),L1     ; branch if its already set.
0000    582          INCW    UCB$W_RWAITCNT(R3)      ; Else, increment wait count to
0000    583                                          ; disallow I/O.
0000    584 L1:      MOVB    #MSCP$K_OP_REPOS,-      ; Transfer REPOSITION opcode
0000    585                  MSCP$B_OPCODE(R2)       ;  to packet.
0000    586          ASSUME  MSCP$V_MD_CLSEX GE 8
0000    587          BICB    #<MSCP$M_MD_CLSEX@-8>,- ; Specifically never clear SEX on the
```

```
0000    588                 MSCP$W_MODIFIER+1(R2)   ;  Seq. NOP command of a SETMODE.
0000    589         SEND_MSCP_MSG                   ; Send message to remote MSCP server.
0000    590         RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
0000    591                                         ;  refresh RSPID, MSG_BUF, etc.
0000    592         .ENDM   START_SEQNOP
0000    593
0000    594 ;
0000    595 ; END_SEQNOP - terminate sequential NOP sequence
0000    596 ;
0000    597 ;       This macro terminates the class driver - server synchronization
0000    598 ;       established by START_SEQNOP and returns the communications to a full
0000    599 ;       stream ahead mode.
0000    600 ;
0000    601 ; Inputs:
0000    602 ;
0000    603 ;       R3      UCB address
0000    604 ;
0000    605 ; Outputs:
0000    606 ;
0000    607 ;       R0 and R3 through R5 unchanged
0000    608 ;       All other registers altered
0000    609
0000    610         .MACRO  END_SEQNOP ?END
0000    611         BICW    #UCB$M_TU_SEQNOP, -     ; Indicate sequential NOP is no longer
0000    612                 UCB$W_DEVSTS(R3)        ; in progress.
0000    613         DECW    UCB$W_RWAITCNT(R3)      ; Decrement wait count to allow I/O.
0000    614         BNEQ    END                     ; Branch if wait count not zero.
0000    615         PUSHR   #^M<R0,R3,R4,R5>        ; Save valuable registers.
0000    616         MOVL    R3, R5                  ; R5 => UCB for SCS$UNSTALLUCB.
0000    617         JSB     G^SCS$UNSTALLUCB        ; Start up any waiting IRPs on this UCB.
0000    618         POPR    #^M<R0,R3,R4,R5>        ; Restore valuable registers.
0000    619 END:
0000    620         .ENDM   END_SEQNOP
```

```
0000    622          .SBTTL  ASSUMES
0000    623
0000    624 ; The following set of ASSUME statements will all be true as long as
0000    625 ;       the IRP and CDRP definitions remain consistent.
0000    626
0000    627          ASSUME  CDRP$L_IOQFL-CDRP$L_IOQFL       EQ      IRP$L_IOQFL
0000    628          ASSUME  CDRP$L_IOQBL-CDRP$L_IOQFL       EQ      IRP$L_IOQBL
0000    629          ASSUME  CDRP$W_IRP_SIZE-CDRP$L_IOQFL    EQ      IRP$W_SIZE
0000    630          ASSUME  CDRP$B_IRP_TYPE-CDRP$L_IOQFL    EQ      IRP$B_TYPE
0000    631          ASSUME  CDRP$B_RMOD-CDRP$L_IOQFL        EQ      IRP$B_RMOD
0000    632          ASSUME  CDRP$L_PID-CDRP$L_IOQFL         EQ      IRP$L_PID
0000    633          ASSUME  CDRP$L_AST-CDRP$L_IOQFL         EQ      IRP$L_AST
0000    634          ASSUME  CDRP$L_ASTPRM-CDRP$L_IOQFL      EQ      IRP$L_ASTPRM
0000    635          ASSUME  CDRP$L_WIND-CDRP$L_IOQFL        EQ      IRP$L_WIND
0000    636          ASSUME  CDRP$L_UCB-CDRP$L_IOQFL         EQ      IRP$L_UCB
0000    637          ASSUME  CDRP$W_FUNC-CDRP$L_IOQFL        EQ      IRP$W_FUNC
0000    638          ASSUME  CDRP$B_EFN-CDRP$L_IOQFL         EQ      IRP$B_EFN
0000    639          ASSUME  CDRP$B_PRI-CDRP$L_IOQFL         EQ      IRP$B_PRI
0000    640          ASSUME  CDRP$L_IOSB-CDRP$L_IOQFL        EQ      IRP$L_IOSB
0000    641          ASSUME  CDRP$W_CHAN-CDRP$L_IOQFL        EQ      IRP$W_CHAN
0000    642          ASSUME  CDRP$W_STS-CDRP$L_IOQFL         EQ      IRP$W_STS
0000    643          ASSUME  CDRP$L_SVAPTE-CDRP$L_IOQFL      EQ      IRP$L_SVAPTE
0000    644          ASSUME  CDRP$W_BOFF-CDRP$L_IOQFL        EQ      IRP$W_BOFF
0000    645          ASSUME  CDRP$L_BCNT-CDRP$L_IOQFL        EQ      IRP$L_BCNT
0000    646          ASSUME  CDRP$W_BCNT-CDRP$L_IOQFL        EQ      IRP$W_BCNT
0000    647          ASSUME  CDRP$L_IOST1-CDRP$L_IOQFL       EQ      IRP$L_IOST1
0000    648          ASSUME  CDRP$L_MEDIA-CDRP$L_IOQFL       EQ      IRP$L_MEDIA
0000    649          ASSUME  CDRP$L_IOST2-CDRP$L_IOQFL       EQ      IRP$L_IOST2
0000    650          ASSUME  CDRP$L_TT_TERM-CDRP$L_IOQFL     EQ      IRP$L_TT_TERM
0000    651          ASSUME  CDRP$B_CARCON-CDRP$L_IOQFL      EQ      IRP$B_CARCON
0000    652          ASSUME  CDRP$Q_NT_PRVMSK-CDRP$L_IOQFL   EQ      IRP$Q_NT_PRVMSK
0000    653          ASSUME  CDRP$L_ABCNT-CDRP$L_IOQFL       EQ      IRP$L_ABCNT
0000    654          ASSUME  CDRP$W_ABCNT-CDRP$L_IOQFL       EQ      IRP$W_ABCNT
0000    655          ASSUME  CDRP$L_OBCNT-CDRP$L_IOQFL       EQ      IRP$L_OBCNT
0000    656          ASSUME  CDRP$W_OBCNT-CDRP$L_IOQFL       EQ      IRP$W_OBCNT
0000    657          ASSUME  CDRP$L_SEGVBN-CDRP$L_IOQFL      EQ      IRP$L_SEGVBN
0000    658          ASSUME  CDRP$L_JNL_SEQNO-CDRP$L_IOQFL   EQ      IRP$L_JNL_SEQNO
0000    659          ASSUME  CDRP$L_DIAGBUF-CDRP$L_IOQFL     EQ      IRP$L_DIAGBUF
0000    660          ASSUME  CDRP$L_SEQNUM-CDRP$L_IOQFL      EQ      IRP$L_SEQNUM
0000    661          ASSUME  CDRP$L_EXTEND-CDRP$L_IOQFL      EQ      IRP$L_EXTEND
0000    662          ASSUME  CDRP$L_ARB-CDRP$L_IOQFL         EQ      IRP$L_ARB
```

```
                     0000   664          .SBTTL  TAPE CLASS DRIVER DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS
                     0000   665
                     0000   666          $DEFINI UCB
                     0000   667
                     0000   668
000000EC             0000   669 .=UCB$K_MSCP_TAPE_LENGTH
                     00EC   670
                     00EC   671 $DEF     UCB$L_TU_MAXWRCNT                   ; Largest size record likely to have
000000F0             00EC   672                         .BLKL   1            ;  reliability statistics.
                     00F0   673 $DEF     UCB$W_TU_FORMAT .BLKW   1            ; Format (density).
                     00F2   674 $DEF     UCB$W_TU_SPEED  .BLKW   1            ; Current speed.
                     00F4   675 $DEF     UCB$W_TU_NOISE  .BLKW   1            ; Size of noise records ignored by
                     00F6   676                                               ;  controller.
                     00F6   677          .IF     DF      TU_SEQCHK
                     00F6   678 $DEF     UCB$B_TU_OLDINX .BLKB   1            ; Index of oldest Sequence number.
                     00F6   679 $DEF     UCB$B_TU_NEWINX .BLKB   1            ; Index of nexat available Seq. # slot.
                     00F6   680 $DEF     UCB$L_TU_SEQARY .BLKL   64           ; Array of 64 longwords wherein we
                     00F6   681                                               ;  we save IRP sequence numbers.
                     00F6   682          .IFF
000000F8             00F6   683                         .BLKW   1            ; Reserved.
                     00F8   684          .ENDC
                     00F8   685
                     00F8   686          .IF     DF      TU_TRACE
                     00F8   687 $DEF     UCB$L_TRACEBEG  .BLKL   1            ; Pointer to beginning of trace ring.
                     00F8   688 $DEF     UCB$L_TRACEPTR  .BLKL   1            ; Pointer to next available slot.
                     00F8   689 $DEF     UCB$L_TRACEND   .BLKL   1            ; Pointer to beyond trace ring.
                     00F8   690
                     00F8   691          .ENDC
                     00F8   692
000000F8             00F8   693 UCB$K_TU_LENGTH=.
                     00F8   694
                     00F8   695          $DEFEND UCB
                     0000   696
                     0000   697
                     0000   698          .SBTTL  Allocate Space for Template UCB
                     0000   699
                     0000   700 ; Allocate zeroed space for template UCB.
                     0000   701
                     0000   702          INIT_UCB size=UCB$K_TU_LENGTH
                     0000   703          INIT_ORB size=ORB$C_LENGTH
```

```
0000    705          .SBTTL  DRIVER PROLOGUE AND DISPATCH TABLES (and UCB Initialization)
0000    706 ;
0000    707 ; LOCAL DATA
0000    708 ;
0000    709 ; DRIVER PROLOGUE TABLE
0000    710 ;
0000    711
0000    712          DPTAB   -                               ;DEFINE DRIVER PROLOGUE TABLE
0000    713                  END=DUTU$END,-                  ; End of driver
0000    714                  ADAPTER=NULL,-                  ; No Adapter
0000    715                  FLAGS=<DPT$M_SCS -              ; Driver requires that SCS be loaded
0000    716                        !DPT$M_NOUNLOAD>,-        ; Driver cannot be reloaded
0000    717                  UCBSIZE=UCB$K_TU_LENGTH,-       ;Sysgen insists on making a UCB
0000    718                  MAXUNITS=1,-                    ;Sysgen insists on making a UCB
0000    719                  NAME=TUDRIVER                   ; Driver name
0038    720          DPT_STORE INIT                          ; Control block init values
0038    721          DPT_STORE DDB,DDB$L_ACPD,L,<^A\MTA\>    ; Default ACP name
003F    722
003F    723
003F    724          ; The following UCB initialization requests alter the template UCB
003F    725          ; as well as producing equivalent DPT_STORE entries.  Thus both
003F    726          ; structures reflect the required initial UCB state and the UCBs
003F    727          ; initially processed by this driver are identical whether they are
003F    728          ; produced by SYSGEN or by IOC$COPY_UCB.
003F    729
003F    730          INIT_UCB        W_SIZE,WORD,UCB$K_TU_LENGTH
003F    731          INIT_UCB        B_TYPE,BYTE,DYN$C_UCB
003F    732          INIT_UCB        B_FIPL,BYTE,IPL$_SCS
0043    733          INIT_UCB        L_DEVCHAR,LONG,<<DEV$M_FOD!-
0043    734                                          DEV$M_DIR!-
0043    735                                          DEV$M_AVL!-
0043    736                                          DEV$M_ELG!-
0043    737                                          DEV$M_IDV!-
0043    738                                          DEV$M_ODV!-
0043    739                                          DEV$M_SDI!-
0043    740                                          DEV$M_SQD>>
004A    741          INIT_UCB        L_DEVCHAR2,LONG,<<DEV$M_CLU!-
004A    742                                           DEV$M_MSCP!-
004A    743                                           DEV$M_NNM>>
0051    744          INIT_UCB        B_DEVCLASS,BYTE,DC$_TAPE
0055    745          INIT_UCB        W_DEVBUFSIZ,WORD,2048
005A    746          INIT_UCB        L_DEVDEPEND,LONG,<<<MT$K_NORMAL11 @ MT$V_FORMAT>!-
005A    747                                             <MT$K_PE_1600 @ MT$V_DENSITY>>>
0061    748          INIT_UCB        W_RWAITCNT,WORD,1
0066    749          INIT_UCB        B_DIPL,BYTE,IPL$_SCS
006A    750          INIT_UCB        W_DEVSTS,WORD, <<UCB$M_MSCP_INITING -
006A    751                                         !UCB$M_MSCP_WAITBMP>>
006F    752
006F    753          ; The following ORB initialization requests alter the template ORB
006F    754          ; as well as producing equivalent DPT_STORE entries.  Thus both
006F    755          ; structures reflect the required initial ORB state and the ORBs
006F    756          ; initially processed by this driver are identical whether they are
006F    757          ; produced by SYSGEN or by IOC$COPY_UCB.
006F    758
006F    759          INIT_ORB        W_SIZE,WORD,ORB$C_LENGTH
006F    760          INIT_ORB        B_TYPE,BYTE,DYN$C_ORB
006F    761          INIT_ORB        B_FLAGS,BYTE,<< -
```

```
006F    762                                 ORB$M_PROT_16>>     ; SOGW protection word
0073    763         INIT_ORB        W_PROT,WORD,0               ; default protection
0078    764         INIT_ORB        L_OWNER,LONG,0              ; no owner as yet
0078    765         DPT_STORE REINIT                    ; Control block re-initialization values
0078    766
0078    767         ; N.B. Causing the following values to be setup during re-initializa-
0078    768         ; tion is not significant because this driver cannot be reloaded.
0078    769         ; However, were the driver to be reloadable the following values would
0078    770         ; need to be re-initialized upon each driver reload.
0078    771
0078    772         DPT_STORE CRB, -                            ; Controller init routine.
0078    773                 CRB$L_INTD+VEC$L_INITIAL,D,TU_CONTROLLER_INIT
007D    774         DPT_STORE DDB,DDB$L_DDT,D,TU$DDT            ; DDT address.
0082    775
0082    776         DPT_STORE END
0000    777
0000    778 ;
0000    779 ; DRIVER DISPATCH TABLE
0000    780 ;
0000    781
0000    782         DDTAB   DEVNAM=TU,-                 ;DRIVER DISPATCH TABLE
0000    783                 START=TU_STARTIO,-          ;START I/O OPERATION
0000    784                 UNSOLIC=TU_UNSOLNT,-        ;UNSOLICITED INTERRUPT
0000    785                 FUNCTB=TU_FUNCTABLE,-       ;FUNCTION DECISION TABLE
0000    786                 CANCEL=DUTU$CANCEL, -       ;CANCEL I/O ENTRY POINT
0000    787                 REGDMP=0,-                  ;REGISTER DUMP ROUTINE
0000    788                 DIAGBF=MSCP$K_MXCMDLEN+MSCP$K_LEN+20+12,-; DIAG BUFF SIZE
0000    789                 ERLGBF=0,-                  ; ERLG BUFF SIZE
0000    790                 UNITINIT=DUTU$UNITINIT,-; Unit initialization routine.
0000    791                 ALTSTART=0                  ; Alternate Start I/O entry.
```

```
0038    793          .SBTTL  DISK CLASS DRIVER FUNCTION DECISION TABLE
0038    794 ;+
0038    795 ; TAPE CLASS DRIVER FUNCTION DECISION TABLE
0038    796 ;-
0038    797
0038    798 TU_FUNCTABLE:                                ;Function Decision Table
0038    799          FUNCTAB ,-                          ; LEGAL FUNCTIONS
0038    800                  <NOP,-                      ;  No operation
0038    801                   UNLOAD,-                   ;  Unload (make available + spindown)
0038    802                   AVAILABLE,-                ;  Available (no spindown)
0038    803                   SPACERECORD,-              ;  Space Records
0038    804                   RECAL,-                    ;  Recalibrate (REWIND)
0038    805                   PACKACK,-                  ;  Pack Acknowledge
0038    806                   ERASETAPE,-                ;  Erase Tape (Erase Gap)
0038    807                   SENSECHAR,-                ;  Sense Characteristics
0038    808                   SETCHAR,-                  ;  Set Characteristics
0038    809                   SENSEMODE,-                ;  Sense Mode
0038    810                   SETMODE,-                  ;  Set Mode
0038    811                   SPACEFILE,-                ;  Space File
0038    812                   WRITECHECK,-               ;  Write Check
0038    813                   READPBLK,-                 ;  Read  PHYSICAL Block
0038    814                   WRITEPBLK,-                ;  Write PHYSICAL Block
0038    815                   READLBLK,-                 ;  Read  LOGICAL  Block
0038    816                   WRITELBLK,-                ;  Write LOGICAL  Block
0038    817                   READVBLK,-                 ;  Read  VIRTUAL  Block
0038    818                   WRITEVBLK,-                ;  Write VIRTUAL  Block
0038    819                   WRITEMARK,-                ;  Write Tape Mark
0038    820                   DSE,-                      ;  Data Security Erase
0038    821                   REWIND,-                   ;  Rewind
0038    822                   REWINDOFF,-                ;  Rewind AND Set Offline (UNLOAD)
0038    823                   SKIPRECORD,-               ;  Skip Records
0038    824                   SKIPFILE,-                 ;  Skip Files
0038    825                   WRITEOF,-                  ;  Write End Of File
0038    826                   ACCESS,-                   ;  Access file and/or find directory entry
0038    827                   ACPCONTROL,-               ;  ACP Control Function
0038    828                   CREATE,-                   ;  Create file and/or create directory entry
0038    829                   DEACCESS,-                 ;  Deaccess file
0038    830                   DELETE,-                   ;  Delete file and/or directory entry
0038    831                   MODIFY,-                   ;  Modify file attributes
0038    832                   MOUNT>                     ;  Mount volume
0040    833          FUNCTAB ,-                          ; BUFFERED I/O FUNCTIONS
0040    834                  <NOP,-                      ;  No Operation
0040    835                   UNLOAD,-                   ;  Unload (make available + spindown)
0040    836                   AVAILABLE,-                ;  Available (no spindown)
0040    837                   SPACERECORD,-              ;  Space Records
0040    838                   RECAL,-                    ;  Recalibrate (REWIND)
0040    839                   PACKACK,-                  ;  Pack Acknowledge
0040    840                   ERASETAPE,-                ;  Erase Tape (Erase Gap)
0040    841                   SENSECHAR,-                ;  Sense Characteristics
0040    842                   SETCHAR,-                  ;  Set   Characteristics
0040    843                   SENSEMODE,-                ;  Sense Mode
0040    844                   SETMODE,-                  ;  Set   Mode
0040    845                   SPACEFILE,-                ;  Space File
0040    846                   WRITEMARK,-                ;  Write Tape Mark
0040    847                   DSE,-                      ;  Data Security Erase
0040    848                   REWIND,-                   ;  Rewind
0040    849                   REWINDOFF,-                ;  Rewind AND Set Offline (UNLOAD)
```

```
0040    850                     SKIPRECORD,-            ;  Skip Records
0040    851                     SKIPFILE,-              ;  Skip Files
0040    852                     WRITEOF,-               ;  Write End Of File
0040    853                     ACCESS,-                ;  Access file and/or find directory entry
0040    854                     ACPCONTROL,-            ;  ACP Control Function
0040    855                     CREATE,-                ;  Create file and/or create directory entry
0040    856                     DEACCESS,-              ;  Deaccess file
0040    857                     DELETE,-                ;  Delete file and/or directory entry
0040    858                     MODIFY,-                ;  Modify file attributes
0040    859                     MOUNT>                  ;  Mount volume
0048    860             FUNCTAB +ACP$READBLK,-          ; READ FUNCTIONS
0048    861                     <READLBLK,-             ;  Read LOGICAL Block
0048    862                      READPBLK,-             ;  Read  PHYSICAL Block
0048    863                      READVBLK>              ;  Read VIRTUAL Block
0054    864             FUNCTAB +ACP$WRITEBLK,-         ; WRITE FUNCTIONS
0054    865                     <WRITECHECK,-           ;  Write Check
0054    866                      WRITEPBLK,-            ;  Write PHYSICAL Block
0054    867                      WRITELBLK,-            ;  Write LOGICAL Block
0054    868                      WRITEVBLK>             ;  Write VIRTUAL Block
0060    869             FUNCTAB +ACP$ACCESS,-           ;
0060    870                     <ACCESS,CREATE>         ; ACCESS AND CREATE FILE OR DIRECTORY
006C    871             FUNCTAB +ACP$DEACCESS,<DEACCESS>; DEACCESS FILE
0078    872             FUNCTAB +ACP$MODIFY,-           ;
0078    873                     <ACPCONTROL,-           ;  ACP Control Function
0078    874                      DELETE,-               ;  Delete file or directory entry
0078    875                      MODIFY>                ;  Modify File Attributes
0084    876             FUNCTAB +ACP$MOUNT,<MOUNT>      ; Mount Volume
0090    877             FUNCTAB +MT$CHECK_ACCESS,-      ;MAGTAPE CHECK ACCESS FUNCTIONS
0090    878                     <ERASETAPE,-            ;  Erase Tape (Erase Gap)
0090    879                      WRITEMARK,-            ;  Write Tape Mark
0090    880                      DSE,-                  ;  Data Security Erase
0090    881                      WRITEOF>               ;  Write End Of File
009C    882             FUNCTAB +EXE$ZEROPARM,-         ; ZERO PARAMETER FUNCTIONS
009C    883                     <NOP,-                  ;  No Operation
009C    884                      UNLOAD,-               ;  Unload (make available + spindown)
009C    885                      RECAL,-                ;  Recalibrate (REWIND)
009C    886                      REWIND,-               ;  Rewind
009C    887                      REWINDOFF,-            ;  Rewind AND Set Offline (UNLOAD)
009C    888                      ERASETAPE,-            ;  Erase Tape (Erase Gap)
009C    889                      SENSECHAR,-            ;  Sense Characteristics
009C    890                      SENSEMODE,-            ;  Sense Mode
009C    891                      WRITEMARK,-            ;  Write Tape Mark
009C    892                      DSE,-                  ;  Data Security Erase
009C    893                      WRITEOF,-              ;  Write End Of File
009C    894                      AVAILABLE,-            ;  Available (no spindown)
009C    895                      PACKACK>               ;  Pack Acknowledge
00A8    896             FUNCTAB +EXE$ONEPARM,-          ;ONE PARAMETER FUNCTIONS
00A8    897                     <SPACERECORD,-          ;  Space Records
00A8    898                      SPACEFILE,-            ;  Space Files
00A8    899                      SKIPRECORD,-           ;  Skip Records
00A8    900                      SKIPFILE>              ;  Skip Files
00B4    901             FUNCTAB +EXE$SETMODE,-          ;SET TAPE CHARACTERISTICS
00B4    902                     <SETCHAR,-              ;
00B4    903                      SETMODE>               ;
```

```
                     00C0   905          .SBTTL  Static Storage
                     00C0   906          .SBTTL  -       Data Area Shared With Common Subroutines Module
                     00C0   907 ;++
                     00C0   908 ;
                     00C0   909 ; Data Area Shared With Common Subroutines Module
                     00C0   910 ;
                     00C0   911 ; Functional Description:
                     00C0   912 ;
                     00C0   913 ;       This PSECT contains those constant (link-time) values which would
                     00C0   914 ;       otherwise be passed as arguments to the disk and tape class driver
                     00C0   915 ;       common routines in module DUTUSUBS.
                     00C0   916 ;
                     00C0   917 ;--
                     00C0   918
                     00C0   919          .SAVE
                     00C0   920
                00000000    921          .PSECT  $$$220_DUTU_DATA_01 RD,WRT,EXE,LONG
                     0000   922
                     0000   923          .ASSUME  DUTU$L_CDDB_LISTHEAD EQ 0
                     0000   924
                     0000   925 ;base + DUTU$L_CDDB_LISTHEAD                    ; Location containing the
                     0000   926                                                 ; address of the CDDB listhead
00000000'            0000   927          .ADDRESS IOC$GL_TU_CDDB                ; for CDDBs belonging to the
                     0004   928                                                 ; tape device type
                     0004   929
                000000C0    930          .RESTORE
```

```
                     00C0   932          .SBTTL  -       Media-id to Device Type Conversion Table
                     00C0   933 ;++
                     00C0   934 ;
                     00C0   935 ; Media-id to Device Type Conversion Table
                     00C0   936 ;
                     00C0   937 ; Functional Description:
                     00C0   938 ;
                     00C0   939 ;       This table is used by DUTU$GET_DEVTYPE to convert a MSCP media
                     00C0   940 ;       identifier to a VMS device type.
                     00C0   941 ;
                     00C0   942 ;       Entries are made here in order of expected frequency of use.  This
                     00C0   943 ;       speeds lookup for the more common cases.
                     00C0   944 ;
                     00C0   945 ;--
                     00C0   946
                     00C0   947          MEDIA   <MU>, <TU81>
                     0000
6D695051             0000                        .LONG   $$MEDIA$$
        08           0004                        .BYTE   DT$_TU81
                     0005
                     00C0   948          MEDIA   <MU>, <TA78>
                     0005
6D68104E             0005                        .LONG   $$MEDIA$$
        06           0009                        .BYTE   DT$_TA78
                     000A
                     00C0   949          MEDIA   <MU>, <TA81>
                     000A
6D681051             000A                        .LONG   $$MEDIA$$
        09           000E                        .BYTE   DT$_TA81
                     000F
                     00C0   950          MEDIA   <MU>, <TK50>
                     000F
6D68B032             000F                        .LONG   $$MEDIA$$
        0A           0013                        .BYTE   DT$_TK50
                     0014
                     00C0   951          MEDIA   <MF>, <TU78>
                     0014
69A9504E             0014                        .LONG   $$MEDIA$$
        05           0018                        .BYTE   DT$_TU78
                     0019
```

```
                        00C0   953          .SBTTL  Controller Initialization Routine
                        00C0   954
                        00C0   955 ;+
                        00C0   956 ; MSCP speaking intelligent controller initialization routine.
                        00C0   957 ;
                        00C0   958 ; INPUTS:
                        00C0   959 ;       R4 => System ID of intelligent controller.
                        00C0   960 ;       R5 => IDB
                        00C0   961 ;       R6 => DDB
                        00C0   962 ;       R8 => CRB for intelligent controller.
                        00C0   963 ;
                        00C0   964
                        00C0   965 TU_CONTROLLER_INIT:
                  06    11 00C0   966          BRB     0$                      ; Branch around breakpoint.
      00000000'GF       16 00C2   967          JSB     G^INI$BRK               ; Breakpoint for debugging.
                        00C8   968 0$:
                        00C8   969
                        00C8   970 ; Check for CDDB already present.  If a CDDB is present, this call results
                        00C8   971 ; from a power failure.  This driver performs power failure recovery as a
                        00C8   972 ; result of virtual circuit closure notification.  No action need be taken
                        00C8   973 ; here.
                        00C8   974
              10 A8     D5 00C8   975          TSTL    CRB$L_AUXSTRUC(R8)      ; Is there a CDDB present?
                 01     13 00CB   976          BEQL    5$                      ; Branch if CDDB is not present.
                        05 00CD   977          RSB                             ; Else, just exit.
                        00CE   978
                        00CE   979 ; Check that only one UCB is chained onto the input DDB.  This UCB could be
                        00CE   980 ; the boot device UCB.  Therefore, make the UCB online so that I/O may be
                        00CE   981 ; performed on it.  All other initialization of the UCB is performed as the
                        00CE   982 ; result of DPT_STORE entries place in the INIT section of the DPT by the
                        00CE   983 ; INIT_UCB macro.
                        00CE   984
                        00CE   985 5$:
           55   04 A6    D0 00CE   986          MOVL    DDB$L_UCB(R6),R5        ; R5 => first UCB if any.
           64 A5  10     C8 00D2   987          BISL    #UCB$M_ONLINE, -        ; Set the possibily boot UCB online.
                           00D6   988                  UCB$L_STS(R5)
                30 A5    D5 00D6   989          TSTL    UCB$L_LINK(R5)          ; Is there another UCB?
                   04    13 00D9   990          BEQL    10$                     ; EQL implies no more UCB's.
                           00DB   991          BUG_CHECK       TAPECLASS,FATAL ; For now.
                           00DF   992 10$:
                           00DF   993
                           00DF   994 ; Setup those values which must be correct before IPL is lowered from 31.
                           00DF   995 ; Then FORK to create an IPL$_SCS fork thread which will complete controller
                           00DF   996 ; initialization.  Initialization of an MSCP server requires several message
                           00DF   997 ; exchanges and consumes several seconds.  Therefore, this work is conducted
                           00DF   998 ; in a fork thread with other system initialization proceeding concurrently.
                           00DF   999
            10 A8    55    D0 00DF  1000          MOVL    R5, CRB$L_AUXSTRUC(R8)  ; The UCB will act as a CDDB until the
                           00E3  1001                                          ; real one is built.
        00CC C5     64    7D 00E3  1002          MOVQ    (R4), -                 ; Setup remote system ID for call to
                           00E8  1003                  UCB$Q_UNIT_ID(R5)       ; DUTU$CREATE_CDDB.
                           00E8  1004
                           00E8  1005          FORK                            ; Create initialization fork thread.
                           00EE  1006
                           00EE  1007 ; Create and initialize the CDDB.
                           00EE  1008
                 FF0F'     30 00EE  1009          BSBW    DUTU$CREATE_CDDB
```

```
                                00F1  1010 ;
                                00F1  1011 ; Here we call an internal subroutine which:
                                00F1  1012 ;
                                00F1  1013 ;       1. Makes a connection to the MSCP server in the intelligent
                                00F1  1014 ;             controller.
                                00F1  1015 ;
                                00F1  1016 ;       2. Sends an MSCP command to SET CONTROLLER CHARACTERISTICS.
                                00F1  1017 ;
                                00F1  1018 ;       3. Allocates an MSCP buffer and RSPID for our future use in
                                00F1  1019 ;             connection management.
                                00F1  1020 ;
                                00F1  1021 ; Upon return R4 => PDT and R5 => CDRP.
                                00F1  1022 ;
                                00F1  1023
          55    00D0 C5   DE    00F1  1024          MOVAL   CDDB$A_PRMCDRP(R5), R5  ; Get permanent CDRP address.
                      0088  30    00F6  1025          BSBW    MAKE_CONNECTION         ; Call internal subroutine to make
                                00F9  1026                                          ;  a connection to the MSCP server in
                                00F9  1027                                          ;  the intelligent controller. Input
                                00F9  1028                                          ;  and output are R5 => CDRP.
                                00F9  1029
                                00F9  1030          PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                                00F9  1031                  cdrp=R5, cddb=R3
             50   18 A3   D0    0100  1032          MOVL    CDDB$L_CRB(R3),R0       ; Get CRB address.
        1C A0   0EF0'CF   9E    0104  1033          MOVAB   W^TU$TMR, -             ; Establish permanent timeout routine.
                                010A  1034                  CRB$L_TOUTROUT(R0)
             51   2A A3   3C    010A  1035          MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Get controller timeout interval.
    18 A0   00000000'GF   51    C1    010E  1036          ADDL3   R1, G^EXE$GL_ABSTIM, -  ; Use that to set next timeout
                                0117  1037                  CRB$L_DUETIME(R0)       ; wakeup time.
                                0117  1038
                                0117  1039          ; The normal MSCP timeout mechanism is now in effect.  Henceforth,
                                0117  1040          ; no fork thread may use the CDDB permanent CDRP as a fork block.
                                0117  1041
                                0117  1042          ASSUME  CDDB$V_DAPBSY GE 8
            13 A3    04   88    0117  1043          BISB    #<CDDB$M_DAPBSY @ -8>, -; Set DAP CDRP in use flag.
                                011B  1044                  CDDB$W_STATUS+1(R3)
            55   54 A3   D0    011B  1045          MOVL    CDDB$L_DAPCDRP(R3), R5  ; Get DAP CDRP address.
                  FEDE'  30    011F  1046          BSBW    DUTU$POLL_FOR_UNITS     ; Poll controller for units.
                                0122  1047
        12 A3   0080 8F   AA    0122  1048          BICW    #CDDB$M_NOCONN, -       ; Now that connection is good, clear
                                0128  1049                  CDDB$W_STATUS(R3)       ; the no connection active bit.
                                0128  1050
    55  53   0000007C 8F   C3    0128  1051          SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB address in R0.
                                0130  1052                   -CDDB$C_UCBCHAIN>, R3, R5
                                0130  1053
          55   00C4 C5   D0    0130  1054 100$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Link to next UCB (if any).
                       1A   13    0135  1055          BEQL    120$                    ; EQL implies no more UCB's.
                                0137  1056          .IF     DEFINED TU_TRACE
                                0137  1057          BSBW    TRACE_INIT              ; Init IRP trace table.
                                0137  1058          .ENDC
        68 A5   0400 8F   AA    0137  1059          BICW    #UCB$M_MSCP_WAITBMP, -  ; Indicate RWAITCNT no longer bumped.
                                013D  1060                  UCB$W_DEVSTS(R5)
                  56 A5   B7    013D  1061          DECW    UCB$W_RWAITCNT(R5)      ; Decrement wait count to allow I/O.
                     03   13    0140  1062          BEQL    110$                    ; Branch if wait count is zero.
                   FEBB'  30    0142  1063          BSBW    DUTU$CHECK_RWAITCNT     ; Else, check wait count validity.
                     3F   BB    0145  1064 110$:     PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers before call.
                00000000'GF   16    0147  1065          JSB     G^SCS$UNSTALLUCB        ; Startup any queued up I/O requests.
                     3F   BA    014D  1066          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers after call.
```

```
                DF  11  0'4F  1067          BRB     100$                    ; Loop back to test more UCB's (if any).
          12 A3 0404 8F  AA  0151  1068 120$:   BICW    #<CDDB$M_INITING -      ; Clear "initing" and DAP CDRP busy
                                0157  1069              !CDDB$M_DAPBSY>, -      ; flags.
                                0157  1070              CDDB$W_STATUS(R3)
                    05          0157  1071          RSB                     ; Terminate this thread of execution.
                                0158  1072
                                0158  1073 INIT_TIMEOUT:                   ; Controller Init Timeout handler.
                  0BF0  31      0158  1074          BRW     TU$RE_SYNCH     ; If we timeout, try to restart.
```

```
                        015B  1077          .SBTTL  MAKE_CONNECTION
                        015B  1078
                        015B  1079 ; MAKE_CONNECTION - Internal subroutine, called from TU_CONTROLLER_INIT and
                        015B  1080 ;       TU$CONNECT_ERR, that establishes a connection to the MSCP server
                        015B  1081 ;       in the intelligent controller.
                        015B  1082 ;
                        015B  1083 ; INPUTS:
                        015B  1084 ;       R5 => permanent CDRP
                        015B  1085 ;
                        015B  1086 ; OUTPUTS:
                        015B  1087 ;       Connection established and initial SET CONTROLLER CHARACTERISTICS
                        015B  1088 ;       command is sent to controller.  Also an MSCP buffer and an RSPID
                        015B  1089 ;       are allocated for the connection.
                        015B  1090 ;
                        015B  1091 ;       Side effects include the fact that all registers, except R5, are
                        015B  1092 ;       modified.
                        015B  1093 ;
                        015B  1094
5F 4C 43 5F 45 50 41 54 24 53 4D 56   015B  1095 CLASS_DRVR_NAME:        .ASCII  /VMS$TAPE_CL_DRVR/
                              52 56 52 44   0167
20 20 20 45 50 41 54 24 50 43 53 4D   016B  1096 MSCP_SRVR_NAME:         .ASCII  /MSCP$TAPE       /
                              20 20 20 20   0177
                        0178  1097
                        0178  1098 HSTIMEOUT_ARRAY:                          ; Host timeouts for various controllers.
                        0178  1099          ASSUME  MSCP$K_CM_HSC50 EQ      1
                        0178  1100          ASSUME  MSCP$K_CM_UDA50 EQ      2
                        0178  1101          ASSUME  MSCP$K_CM_RC25  EQ      3
                        0178  1102          ASSUME  MSCP$K_CM_EMULA EQ      4
                        0178  1103          ASSUME  MSCP$K_CM_TU81  EQ      5
                        0178  1104          ASSUME  MSCP$K_CM_UDA52 EQ      6
                     1E 0178  1105          .BYTE   HOST_TIMEOUT            ; Use default constant for HSC50.
                     00 017C  1106          .BYTE   0                       ; Use zero for dedicated controller.(UDA50)
                     00 017D  1107          .BYTE   0                       ; Use zero for dedicated controller.(AZTEC)
                     1E 017E  1108          .BYTE   HOST_TIMEOUT            ; Use default constant for Emulator.
                     00 017F  1109          .BYTE   0                       ; Use zero for dedicated controller.(TU81)
                     00 0180  1110          .BYTE   0                       ; Use zero for dedicated controller.(UDA52)
                        0181  1111
                        0181  1112 MAKE_CONNECTION:
                        0181  1113
                        0181  1114          PERMCDRP_TO_CDDB -              ; Get CDDB address from CDRP.
                        0181  1115                  cdrp=R5, cddb=R2
           44 A2 8ED0   0188  1116          POPL    CDDB$L_SAVED_PC(R2)     ; Save caller's return in CDDB field.
                        018C  1117 5$:
     00000000'GF    D0   018C  1118          MOVL    G^EXE$GL_ABSTIM,-       ; Copy absolute time that we entered
           30 A2        0192  1119                  CDDB$L_OLDCMDSTS(R2)    ;  this routine, or the last time that
                        0194  1120                                          ;  terminated all pending I/O.
                        0194  1121 10$:
50   00000000'GF    D0   0194  1122          MOVL    G^SGN$GL_VMSD3,R0       ; Pickup interval of seconds that we
                        019B  1123                                          ;  should try to CONNECT until we
                        019B  1124                                          ;  decide to terminate pending I/O.
              12    13   019B  1125          BEQL    15$                     ; EQL implies infinite timeout.
        50    30 A2   C0   019D  1126          ADDL    CDDB$L_OLDCMDSTS(R2),R0 ; Sum is end of timeout interval.
00000000'GF    50    D1   01A1  1127          CMPL    R0,G^EXE$GL_ABSTIM      ; See if we have timed out.
              05    14   01A8  1128          BGTR    15$                     ; GTR means no, time remains.
            0150    30   01AA  1129          BSBW    TERMINATE_PENDING       ; Else call to terminate all pending I/O
              DD    11   01AD  1130          BRB     5$                      ; Loop back to establish a new timeout
                        01AF  1131                                          ;  period.
```

```
                     01AF  1132 15$:
                     01AF  1133
                     01AF  1134 CONNECT TU$IDR,-                    ; Entry point of Input Dispatcher Routine.
                     01AF  1135         TU$DGDR,-                   ; Entry point of Datagram Dispatcher.
                     01AF  1136         TU$CONNECT_ERR,-            ; Error entry point.
                     01AF  1137         CDDB$B_SYSTEMID(R2),-       ; Destination SYSTEM ID.
                     01AF  1138         ,-                          ; Remote station address.
                     01AF  1139         MSCP_SRVR_NAME,-            ; MSCP server name.
                     01AF  1140         CLASS_DRVR_NAME,-           ; Ascii of class driver name.
                     01AF  1141         #INITIAL_CREDIT,-           ;  Needs definition
                     01AF  1142         #MIN_SEND_CREDIT,-          ; Minimum send credit
                     01AF  1143         #INITIAL_DG_COUNT,-         ; Initial DataGram count
                     01AF  1144         ,-                          ; Block transfer priority
                     01AF  1145         ,-                          ; Connect data
                     01AF  1146         (R2),-                      ; Also pass CDDB address to CDT$L_AUXSTRUC
                     01E7  1147         ,                           ; Bad Response packet address
           28 50  E8 01E7  1148         BLBS    R0,30$              ; LBS implies success, so branch around.
                     01EA  1149
      52   08 A5  32 01EA  1150         CVTWL   CDRP$W_CDRPSIZE(R5),R2 ; R2 has negative offset, from base of
                     01EE  1151                                     ;  CDRP, of base of CDDB.
        52   55  C0 01EE  1152         ADDL    R5,R2               ; R2 => CDDB.
      53   18 A2  D0 01F1  1153         MOVL    CDDB$L_CRB(R2),R3   ; R3 => CRB.
                     01F5  1154
   1C A3   04'AF  9E 01F5  1155         MOVAB   B^20$,CRB$L_TOUTROUT(R3); Establish LABEL as place to call, for
                     01FA  1156                                     ;  now, for periodic wakeups.
           0A  C1 01FA  1157         ADDL3   #CONNECT_DELTA,-       ; Establish due time as a little in
    00000000'GF  01FC  1158                 G^EXE$GL_ABSTIM,-      ;  the future.
           18 A3  0201  1159                 CRB$L_DUETIME(R3)
              05  0203  1160         RSB                            ; Return to caller's caller and kill
                     0204  1161                                     ;  this thread.
                     0204  1162 20$:
      52   10 A3  D0 0204  1163         MOVL    CRB$L_AUXSTRUC(R3),R2 ; R2 => CDDB.
   55   00D0 C2  9E 0208  1164         MOVAB   CDDB$A_PRMCDRP(R2),R5 ; Get permanent CDRP address.
                0D  1165         SETIPL  #IPL$_SCS              ; Lower IPL after wakeup.
              82  11 0210  1166         BRB     10$                ; Loop back and try CONNECT again.
                     0212  1167 30$:    ; A connection has been established
                     0212  1168         PERMCDRP_TO_CDDB -             ; Get CDDB address from CDRP.
                     0212  1169                 cdrp=R5, cddb=R1
   00F4 C1   53  D0 0219  1170         MOVL    R3, CDDB$L_CDT(R1)     ; Save CDT address (in perm CDRP).
     14 A1   54  D0 021E  1171         MOVL    R4, CDDB$L_PDT(R1)     ; Save PDT address.
   01B8 C1   53  D0 0222  1172         MOVL    R3, CDDB$L_DAPCDT(R1)  ; Save CDT address in DAP CDRP too.
        53   51  D0 0227  1173         MOVL    R1, R3                 ; Now that CDT is saved, move CDDB addr.
                     022A  1174
      51   18 A3  D0 022A  1175         MOVL    CDDB$L_CRB(R3), R1     ; Get CRB address.
      18 A1   01  CE 022E  1176         MNEGL   #1, CRB$L_DUETIME(R1)  ; Infinite time till next timeout, now.
   1C A1   FF22 CF 9E 0232  1177         MOVAB   INIT_TIMEOUT, -        ; Establish timeout routine that will
                     0238  1178                 CRB$L_TOUTROUT(R1)     ; serve for rest of controller init.
                     0238  1179
                     0238  1180 ;
                     0238  1181 ; Here we prepare to send a SET CONTROLLER CHARACTERISTICS MSCP Packet to
                     0238  1182 ;       the intelligent controller over the connection that we have just
                     0238  1183 ;       established.
                     0238  1184 ;
                     0238  1185
                     0238  1186         ALLOC_RSPID                    ; ALLOCate a ReSPonse ID.
                     023E  1187         ALLOC_MSG_BUF                  ; Allocate an MSCP buffer (and also
                     0241  1188                                        ;  allocate a unit of flow control).
```

```
                07 50   E8  0241  1189          BLBS    R0,50$                  ; If success, branch around.
           53   18 A3   D0  0244  1190          MOVL    CDDB$L_CRB(R3),R3       ; TU$RE_SYNCH expects R3 => CDDB.
                0B00    31  0248  1191          BRW     TU$RE_SYNCH             ; Failure here means we must re-CONNECT.
                        024B  1192 50$:                                         ; Here R2 => MSCP buffer allocated.
                  51    D4  024B  1193          CLRL    R1                      ; First set Controller Characteristics
                        024D  1194                                              ;  with zero (i.e. infinite) host timeout.
                  3C    10  024D  1195          BSBB    PRP_STCON_MSG           ; Call to prepare MSCP command.
                        024F  1196          SEND_MSCP_MSG_DRIVER                ; Returns with end-message addr. in R2.
                006A    30  0252  1197          BSBW    RECORD_STCON            ; Record Controller Characteristics.
                        0255  1198
                        0255  1199          RECYCH_MSG_BUF                      ; We recycle the END PACKET and
                        0258  1200                                              ;  thereby allocate a new send credit.
                        0258  1201          RECYCL_RSPID                        ; We also recycle the RSPID.
                        025E  1202
                        025E  1203 ; Determine the correct host timeout interval.  This is the larger of
                        025E  1204 ; HSTIMEOUT_ARRAY[controller_model] and the controller timeout interval
                        025E  1205 ; returned by the just completed Set Controller Characteristics.  There is,
                        025E  1206 ; however, one wrinkle.  Zero represents an infinite timeout and therefore is
                        025E  1207 ; larger than any other number.  Also, the controller already believes the
                        025E  1208 ; host timeout interval to be infinite, as the result of the previous Set
                        025E  1209 ; Controller Characteristics command.  Therefore, no further action need be
                        025E  1210 ; taken when the timeout interval is infinite.
                        025E  1211
           51   26 A3   9A  025E  1212          MOVZBL  CDDB$B_CNTRLMDL(R3),R1  ; Get controller model type.
      51    FF13 CF41   9A  0262  1213          MOVZBL  HSTIMEOUT_ARRAY-1[R1],R1; Get corresponding host timeout value.
                  1E    13  0268  1214          BEQL    60$                     ; If zero, branch around.
           50   2A A3   3C  026A  1215          MOVZWL  CDDB$W_CNTRLTMO(R3), R0 ; Get controller timeout interval.
                  18    13  026E  1216          BEQL    60$                     ; If controller timeout is infinite,
                        0270  1217                                              ; use already set infinite host timeout.
                51  50   D1  0270  1218          CMPL    R0, R1                  ; Compare with HSTIMEOUT_ARRAY value.
                  03    1F  0273  1219          BLSSU   55$                     ; Branch if HSTIMEOUT_ARRAY is larger.
                51  50   D0  0275  1220          MOVL    R0, R1                  ; Else, use controller timeout as
                        0278  1221                                              ; host timeout interval.
                        0278  1222 55$:
                  11    10  0278  1223          BSBB    PRP_STCON_MSG           ; Else reset controller characteristics.
                        027A  1224          SEND_MSCP_MSG_DRIVER                ; Returns with end-message addr. in R2.
                  40    10  027D  1225          BSBB    RECORD_STCON            ; Record Controller Characteristics.
                        027F  1226          RECYCH_MSG_BUF                      ; Again we recycle the END PACKET and
                        0282  1227                                              ;  thereby allocate a new send credit.
                        0282  1228          RECYCL_RSPID                        ; We also recycle the RSPID.
                        0288  1229 60$:
                        0288  1230
                44 B3   17  0288  1231          JMP     @CDDB$L_SAVED_PC(R3)    ; Return to caller.
```

```
                     028B  1233 ; PRP_STCON_MSG - Prepare a Set Controller Characteristics Command Message.
                     028B  1234 ;
                     028B  1235 ; Inputs:
                     028B  1236 ;       R1 =  Host Timeout Value
                     028B  1237 ;       R2 => MSCP buffer to fill
                     028B  1238 ;       R3 => CDDB
                     028B  1239 ;       R5 => CDRP
                     028B  1240 ;
                     028B  1241
                     028B  1242 PRP_STCON_MSG:
                     028B  1243
          51    DD   028B  1244         PUSHL   R1                      ; Save important register.
                     028D  1245         INIT_MSCP_MSG                   ; Initialize buffer for MSCP message.
          51  8ED0   0290  1246         POPL    R1                      ; Restore important register.
                     0293  1247
          04    90   0293  1248         MOVB    #MSCP$K_OP_STCON,-      ; Insert SET CONTROLLER CHARACTERISTICS
       08 A2         0295  1249                 MSCP$B_OPCODE(R2)       ;  opcode with NO modifiers.
                     0297  1250
       28 A3    B0   0297  1251         MOVW    CDDB$W_CNTRLFLGS(R3),-  ; Set host settable characteristics
       0E A2         029A  1252                 MSCP$W_CNT_FLGS(R2)     ;  bits into MSCP command message.
                     029C  1253
 10 A2    51    B0   029C  1254         MOVW    R1,MSCP$W_HST_TMO(R2)   ; Set host timeout into MSCP packet.
                     02A0  1255
00000000'GF    7D   02A0  1256         MOVQ    G^EXE$GQ_SYSTIME,-      ; Transmit time of century in clunks.
       14 A2         02A6  1257                 MSCP$Q_TIME(R2)
                     02A8  1258
 50    18 A3    D0   02A8  1259         MOVL    CDDB$L_CRB(R3),R0       ; R0 => CRB.
 7E    2A A3    3C   02AC  1260         MOVZWL  CDDB$W_CNTRLTMO(R3),-(SP); Pickup controller delta.
          03    12   02B0  1261         BNEQ    70$                     ; NEQ implies this controller has been
                     02B2  1262                                         ;  init'ed at least once before.
    6E    1E    D0   02B2  1263         MOVL    #INIT_IMMED_DELTA,(SP)  ; Else use compiled in timeout.
                     02B5  1264 70$:
          8E    C1   02B5  1265         ADDL3   (SP)+,-                 ; Establish delta time for time out
00000000'GF         02B7  1266                 G^EXE$GL_ABSTIM,-       ;  to prevent against controller never
       18 A0         02BC  1267                 CRB$L_DUETIME(R0)       ;  responding.
                     02BE  1268
          05    05   02BE  1269         RSB                             ; Return to caller.
```

```
                                02BF  1271 ; RECORD_STCON - Record data from a Set Controller Characteristics end message
                                02BF  1272 ;               in the CDDB.
                                02BF  1273 ;
                                02BF  1274 ; Inputs:
                                02BF  1275 ;       R2 => MSCP End Message
                                02BF  1276 ;       R3 => CDDB
                                02BF  1277
                                02BF  1278
                                02BF  1279 RECORD_STCON:
                    0E A2   B0  02BF  1280          MOVW    MSCP$W_CNT_FLGS(R2),-   ; Pickup NON-host settable characteristics
                    28 A3       02C2  1281                  CDDB$W_CNTRLFLGS(R3)    ;  from END PACKET and save in CDDB.
                                02C4  1282
                    10 A2   B0  02C4  1283          MOVW    MSCP$W_CNT_TMO(R2),-    ; Likewise with controller timeout.
                    2A A3       02C7  1284                  CDDB$W_CNTRLTMO(R3)
                                02C9  1285
                    14 A2   7D  02C9  1286          MOVQ    MSCP$Q_CNT_ID(R2),-     ; Also save controller unique ID.
                    20 A3       02CC  1287                  CDDB$Q_CNTRLID(R3)
                                02CE  1288
            29 12 A3    06  E2  02CE  1289          BBSS    #CDDB$V_ALCLS_SET, -    ; Branch if allocation class already
                                02D3  1290                  CDDB$W_STATUS(R3), 90$  ; set, and indicate it is now set.
                                02D3  1291          ; The allocation class is about to be set for this device.  The object
                                02D3  1292          ; is to give every reasonable chance for the value to be non-zero.
        50 A3   00000000'GF  D0  02D3  1293          MOVL    G^CLU$GL_ALLOCLS, -     ; Assume a local, single host
                                02DB  1294                  CDDB$L_ALLOCLS(R3)      ; controller.
              26 A3     01  91  02DB  1295          CMPB    #MSCP$K_CM_HSC50, -     ; Is this an HSC?
                                02DF  1296                  CDDB$B_CNTRLMDL(R3)
                        05  13  02DF  1297          BEQL    1099$                   ; Branch to multihost leg, if HSC.
            05 28 A3    02  E1  02E1  1298          BBC     #MSCP$V_CF_MLTHS, -     ; Branch if a single host controller.
                                02E6  1299                  CDDB$W_CNTRLFLGS(R3), -
                                02E6  1300                  80$
                                02E6  1301 1099$:
        50 A3   04 A2   9A      02E6  1302          MOVZBL  MSCP$B_CNT_ALCS(R2), -  ; Get set controller characteristics
                                02EB  1303                  CDDB$L_ALLOCLS(R3)      ; allocation class.
           50   E4 A3   9E      02EB  1304 80$:     MOVAB   <CDDB$L_DDB -           ; Init loop through all DDBs.
                                02EF  1305                  -DDB$L_CONLINK>(R3), R0
           50   38 A0   D0      02EF  1306 82$:     MOVL    DDB$L_CONLINK(R0), R0   ; Link to next DDB.
                    07  13      02F3  1307          BEQL    90$                     ; Branch if no more DDBs.
        3C A0   50 A3   D0      02F5  1308          MOVL    CDDB$L_ALLOCLS(R3), -   ; Copy allocation class to this
                                02FA  1309                  DDB$L_ALLOCLS(R0)       ; DDB.
                    F3  11      02FA  1310          BRB     82$                     ; Loop till no more DDBs.
                                02FC  1311
                        05      02FC  1312 90$:     RSB
```

```
                                02FD  1314          .SBTTL  TERMINATE_PENDING
                                02FD  1315
                                02FD  1316 ; TERMINATE_PENDING - internal routine called from MAKE_CONNECTION.
                                02FD  1317 ;       The purpose of this routine is to terminate all pending I/O on
                                02FD  1318 ;       this connection because the amount of time specified in a SYSGEN
                                02FD  1319 ;       parameter has passed without being able to CONNECT.
                                02FD  1320 ;
                                02FD  1321 ; Inputs:
                                02FD  1322 ;       R2 => CDDB
                                02FD  1323 ;       R5 => CDRP
                                02FD  1324 ;
                                02FD  1325 ; Outputs:
                                02FD  1326 ;       Registers R0, R1, R3 are modified.
                                02FD  1327 ;
                                02FD  1328
                                02FD  1329 TERMINATE_PENDING:
                          02    E0  02FD  1330          BBS     #CDDB$V_INITING,-        ; Do not time out during initialization.
                       3D 12  A2      02FF  1331                  CDDB$W_STATUS(R2),50$
                                0302  1332 10$:
                       50     3C  B2  0F  0302  1333          REMQUE  @CDDB$L_RSTRTQFL(R2),R0 ; REMQUE a pending CDRP. R0 => CDRP.
                                  0F  1D  0306  1334          BVS     20$                     ; VS implies queue empty.
                                      0308  1335          POST_CDRP status=SS$_CTRLERR    ; Terminate this CDRP.
                                  EB  11  0315  1336          BRB     10$                     ; Loop thru all CDRP's on CDDB Q.
                                0317  1337 20$:
                52    0000007C  8F  C3  0317  1338          SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB in R3.
                                0318  1339                    -CDDB$L_UCBCHAIN>, -
                                  53      0318  1340                  R2, R3
                                0319  1341
                     53   00C4  C3  D0  0319  1342 30$:     MOVL    UCB$L_CDDB_LINK(R3), R3 ; Chain to next UCB (if any).
                                  19  13  0320  1343          BEQL    50$                     ; EQL implies no more UCB's here.
                                0322  1344 40$:
                      50    4C  B3  0F  0322  1345          REMQUE  @UCB$L_IOQFL(R3),R0     ; R0 => IRP on Q.
                                  F3  1D  0326  1346          BVS     30$                     ; VS implies I/O queue empty.
                50    60    A0  9E  0328  1347          MOVAB   -CDRP$L_IOQFL(R0),R0    ; R0 => CDRP portion of IRP.
                                0330  1348          POST_CDRP status=SS$_CTRLERR    ; Terminate this CDRP.
                                  E7  11  0335  1349          BRB     40$                     ; Loop thru all IRP's on UCB.
                                0337  1350 50$:
                                  05  0337  1351          RSB                             ; Return to caller.
```

```
                     0340  1353          .SBTTL  BRING_UNIT_ONLINE
                     0340  1354
                     0340  1355 ; BRING_UNIT_ONLINE - Internal subroutine to bring an available unit online.
                     0340  1356 ;       This subroutine is called from TU$CONNECT_ERR.
                     0340  1357 ;
                     0340  1358 ; INPUTS:
                     0340  1359 ;       R3 => CDDB
                     0340  1360 ;       R4 => PDT
                     0340  1361 ;       R5 => UCB
                     0340  1362 ;
                     0340  1363 ; Implicit Inputs:
                     0340  1364 ;
                     0340  1365 ;       CDDB$W_STATUS(R3)       CDDB$V_DAPBSY set
                     0340  1366 ;
                     0340  1367 ;       The normal class driver MSCP operation timeout mechanism must be
                     0340  1368 ;       enabled.
                     0340  1369 ;
                     0340  1370
                     0340  1371 BRING_UNIT_ONLINE:
                     0340  1372
     44 A3 8ED0      0340  1373          POPL    CDDB$L_SAVED_PC(R3)     ; Save caller's return address.
50   54 A3   D0      0344  1374          MOVL    CDDB$L_DAPCDRP(R3), R0  ; Get DAP CDRP address.
   53   55   D0      0348  1375          MOVL    R5, R3                  ; Copy UCB address.
   55   50   D0      034B  1376          MOVL    R0, R5                  ; Copy CDRP address.
                     034E  1377
BC A5   53   D0      034E  1378          MOVL    R3, CDRP$L_UCB(R5)      ; Setup UCB address in CDRP.
                     0352  1379
                     0352  1380          ALLOC_MSG_BUF                   ; Allocate a message buffer.
     01 50   E8      0355  1381          BLBS    R0, 3$                  ; Branch if connection is not broken.
                05   0358  1382          RSB                             ; Else, just kill this fork thread.
                     0359  1383 3$:       ALLOC_RSPID                     ; Allocate a response-id.
                     035F  1384          INIT_MSCP_MSG ucb=(R3)          ; Initialize buffer for MSCP message.
                     0362  1385
          09   90    0362  1386          MOVB    #MSCP$K_OP_ONLIN,-      ; ONLINE command, zero modifiers.
        08 A2         0364  1387                  MSCP$B_OPCODE(R2)
                     0366  1388
                A8   0366  1389          BISW    #MSCP$M_MD_CLSEX-       ; Do exclusive ONLINE and clear serious
                     0367  1390                  !MSCP$M_MD_EXCLU,-      ;  exception.
0A A2     2020 8F    0367  1391                  MSCP$W_MODIFIER(R2)
                     036C  1392
        00E0 C3   B0 036C  1393          MOVW    UCB$W_UNIT_FLAGS(R3),-  ; Copy UNIT flags to MSCP packet.
          0E A2      0370  1394                  MSCP$W_UNT_FLGS(R2)
                     0372  1395
        00D8 C3   D0 0372  1396          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
          1C A2      0376  1397                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                     0378  1398
             08   EF 0378  1399          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
                05   037A  1400                  #MT$S_DENSITY,-         ;  last established for this unit
     50   4' A3      037B  1401                  UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
                     037E  1402
           008B   30 037E  1403          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
     20 A2   51   B0 0381  1404          MOVW    R1,MSCP$W_FORMAT(R2)    ; Move MSCP density in R1 into packet.
                     0385  1405
             05   E1 0385  1406          BBC     #MSCP$V_UF_VSMSU,-      ; Test if we are suppressing variable
  0D 0E A2           0387  1407                  MSCP$W_UNT_FLGS(R2),10$ ;  speed mode, and branch if NOT.
             18   EF 038A  1408          EXTZV   #MT$V_SPEED,-           ; Extract user's speed specification
                08   038C  1409                  #MT$S_SPEED,-           ;  from UCB.
```

```
                50   44 A3        038D  1410                    UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
                      009D   30   0390  1411            BSBW    SPEEDTOMSCP
                22 A2    50   B0 0393  1412            MOVW    R0,MSCP$W_SPEED(R2)     ; Move MSCP speed in R0 into packet.
                                0397  1413
                                0397  1414 10$:        SEND_MSCP_MSG DRIVER            ; ONLIN - returns end pkt. addr. in R2.
                                039A  1415            IF_MSCP FAILURE, then=30$       ; Branch if ONLIN failed.
                                03A0  1416
                                03A0  1417 ; If here then various fields in the END PACKET are valid.
                                03A0  1418 ;       Here we have just brought ONLINE a unit that was online before
                                03A0  1419 ;       as a result of a failed previous CONNECTION.  We assume
                                03A0  1420 ;       that the volume is identical to the one that was ONLINE here before.
                                03A0  1421 ;       And then setup the UCB accordingly.
                                03A0  1422 ;
                                03A0  1423
                     03F2   30  03A0  1424            BSBW    RECORD_ONLINE           ; Move data from end message to UCB.
                                03A3  1425
                                03A3  1426            RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
                                03A6  1427
                       03   90  03A6  1428            MOVB    #MSCP$K_OP_GTUNT,-      ; GET UNIT STATUS command, zero modifiers.
                     08 A2      03A8  1429                    MSCP$B_OPCODE(R2)
                                03AA  1430
                                03AA  1431            SEND_MSCP_MSG DRIVER            ; GTUNT - returns end pkt. addr. in R2.
                                03AD  1432            IF_MSCP FAILURE, then=30$       ; Branch if GTUNT failed.
                                03B3  1433
                     03ED   30  03B3  1434            BSBW    RECORD_GETUNIT_CHAR     ; Record UNIT status data in UCB.
                                03B6  1435
                                03B6  1436 ; Here reposition out to where we were before.
                                03B6  1437
                                03B6  1438            RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
                                03B9  1439
                       25   90  03B9  1440            MOVB    #MSCP$K_OP_REPOS,-      ; Reposition command.
                     08 A2      03BB  1441                    MSCP$B_OPCODE(R2)
                          A8    03BD  1442            BISW    #MSCP$M_MD_REWND-       ; Rewind and then space out an absolute
                                03BE  1443                    !MSCP$M_MD_OBJCT,-      ;  number of objects.
                22 A2    06         03BE  1444                    MSCP$W_MODIFIER(R2)
                   00B0 C3    D0    03C1  1445            MOVL    UCB$L_RECORD(R3),-      ; Copy number of objects (gaps) to skip
                     0C A2          03C5  1446                    MSCP$L_REC_CNT(R2)      ;  into MSCP command packet.
                                03C7  1447
                                03C7  1448            SEND_MSCP_MSG DRIVER            ; REPOS - returns end pkt. addr. in R2.
                                03CA  1449            IF_MSCP FAILURE, then=30$       ; Branch if REPOS failed.
                                03D0  1450
                      FC2D'  30 03D0  1451 20$:       BSBW    DUTU$DEALLOC_ALL        ; Deallocate all CDRP resources.
                                03D3  1452
                                03D3  1453            PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                                03D3  1454                    cdrp=R5, cddb=R3
                55   BC A5    D0   03DA  1455            MOVL    CDRP$L_UCB(R5), R5      ; Restore input UCB address.
                     44 B3    17   03DE  1456            JMP     @CDDB$L_SAVED_PC(R3)    ; Return to caller.
                                03E1  1457
                                03E1  1458 30$:                                         ; HERE if volume has changed.
                                03E1  1459            ASSUME  UCB$V_VALID GE 8
                65 A3    08   8A  03E1  1460            BICB    #<UCB$M_VALID @ -8>, -  ; If could not put the drive ONLINE,
                                03E5  1461                    UCB$W_STS+1(R3)         ; clear the volume valid bit.
                          07   E1  03E5  1462            BBC     #MSCP$V_SC_DUPUN,-      ; Branch around if NOT duplicate
                  03 0A A2         03E7  1463                    MSCP$W_STATUS(R2),40$   ;  unit substatus.
                      FC13'  30   03EA  1464            BSBW    DUTU$SEND_DUPLICATE_UNIT; Notify operator of duplicate unit.
                                03ED  1465 40$:
                                03ED  1466            RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
```

```
                08    90  03F0  1467          MOVB    #MSCP$K_OP_AVAIL,-      ; Available command
            08  A2        03F2  1468                  MSCP$B_OPCODE(R2)
                          03F4  1469          SEND_MSCP_MSG DRIVER            ; AVAIL - returns end pkt. addr. in R2.
                D7    11  03F7  1470          BRB     20$                     ; Join common exit code.
```

```
                        .IF     DF      TU_SEQCHK
                        .SBTTL  -       OVERRIDE_SEQCHK and REMOVE_SEQARY

                ;+
                ; OVERRIDE_SEQCHK - Set UCB$M_TU_OVRSQCHK bit in UCB$W_DEVSTS and then fall
                ;                  thru to
                ; REMOVE_SEQARY - Remove this IRP$L_SEQNUM from the UCB$L_TU_SEQARY and
                ;                 collapse the array.
                ;
                ; Inputs:
                ;       R5 => CDRP
                ;

03F9  1486 OVERRIDE_SEQCHK:

03F9  1488              PUSHL   R0                      ; Save R0.
03F9  1489              MOVL    CDRP$L_UCB(R5),R0       ; R0 => UCB.
03F9  1490              BISW    #UCB$M_TU_OVRSQCHK,-    ; Set bit to override sequence
03F9  1491                      UCB$W_DEVSTS(R0)        ;  checking on this operation.
03F9  1492              POPL    R0                      ; Restore R0.
03F9  1493
03F9  1494 REMOVE_SEQARY:
03F9  1495
03F9  1496              MOVQ    R0,-(SP)                ; Save registers.
03F9  1497              PUSHL   R3                      ;
03F9  1498              MOVL    CDRP$L_UCB(R5),R3       ; R3 => UCB.
03F9  1499              EXTZV   #0,#6,-                 ; Extract index of oldest array slot.
03F9  1500                      UCB$B_TU_OLDINX(R3),R0
03F9  1501              EXTZV   #0,#6,-                 ; Extract index of next  array slot.
03F9  1502                      UCB$B_TU_NEWINX(R3),R1
03F9  1503 10$:
03F9  1504              EXTZV   #0,#6,R0,R0             ; Reduce R0 to 6-bit index.
03F9  1505              CMPL    R0,R1                   ; Have we run thru entire array?
03F9  1506              BEQL    50$                     ; EQL implies yes.
03F9  1507              CMPL    CDRP$L_SEQNUM(R5),-     ; If not, is this array slot ours?
03F9  1508                      UCB$L_TU_SEQARY(R3)[R0]
03F9  1509              BEQL    20$                     ; EQL implies YES.
03F9  1510              INCL    R0                      ; Bump index.
03F9  1511              BRB     10$                     ; And continue loop.
03F9  1512 20$:                                         ; Here R0 has array slot index.
03F9  1513              EXTZV   #0,#6,-                 ; Extract index of oldest array slot.
03F9  1514                      UCB$B_TU_OLDINX(R3),-(SP)
03F9  1515 30$:                                         ; Here we collapse the array by moving
03F9  1516                                              ;  each slot preceeding the slot to
03F9  1517                                              ;  remove, one position forward.  We
03F9  1518                                              ;  begin with the slot immediately
03F9  1519                                              ;  preceeding the found one.
03F9  1520              EXTZV   #0,#6,R0,R0             ; Reduce R0 to 6-bit index.
03F9  1521              CMPL    R0,(SP)                 ; Are we done?
03F9  1522              BEQL    40$                     ; EQL implies we are done.
03F9  1523              SUBL3   #1,R0,R1                ; R1 has index of preceeding slot.
03F9  1524              EXTZV   #0,#6,R1,R1             ; Reduce R1 to 6-bit index.
03F9  1525              MOVL    UCB$L_TU_SEQARY(R3)[R1],-       ; Move slot contents forward one
03F9  1526                      UCB$L_TU_SEQARY(R3)[R0]         ;  position.
03F9  1527              DECL    R0                      ; Decrement index.
03F9  1528              BRB     30$                     ; And continue in loop.
03F9  1529 40$:
```

```
03F9  1530          INCB    UCB$B_TU_OLDINX(R3)     ; Increment index to reflect collapse.
03F9  1531          TSTL    (SP)+                   ; Remove junk from stack.
03F9  1532 50$:
03F9  1533          POPL    R3                      ; Restore registers.
03F9  1534          MOVQ    (SP)+,R0
03F9  1535          RSB                             ; Return to caller.
03F9  1536          .ENDC
```

```
                        .SBTTL  Density and Speed Conversion Routines

                ;+
                ; VMSTOMSCP_DENS - Internal subroutine to convert from a VMS density
                ;       code to a MSCP density code.
                ;
                ; Inputs:
                ;       R0 = VMS density code
                ;
                ; Outputs:
                ;       R1 = MSCP density code
                ;               R0 = 0 which implies that the VMS code was such that we chose
                ;                    the default MSCP code
                ;               R0 = 1 which implies that the VMS code was a perfect match for
                ;                    one of the codes.

                TU_VMSDENS:
03              .BYTE   MT$K_NRZI_800
04              .BYTE   MT$K_PE_1600
05              .BYTE   MT$K_GCR_6250
04              .BYTE   MT$K_PE_T600    ; Redundant for NOT FOUND case default.

                TU_MSCPDENS:
01              .BYTE   MSCP$M_TF_800
02              .BYTE   MSCP$M_TF_PE
04              .BYTE   MSCP$M_TF_GCR

                TU_ABSDENS:
0320            .WORD   800
0640            .WORD   1600
186A            .WORD   6250
0640            .WORD   1600            ; Redundant for NOT FOUND case.

                TU_ABSPEED:
19              .BYTE   25
4B              .BYTE   75
7D              .BYTE   125
FF              .BYTE   255

                VMSTOMSCP_DENS:

                        ASSUME  MT$K_NRZI_800   EQ      3
                        ASSUME  MT$K_PE_1600    EQ      4
                        ASSUME  MT$K_GCR_6250   EQ      5

51  50  03  C3          SUBL3   #3,R0,R1                ; Subtract out NRZI bias from VMS code.
        08  19          BLSS    10$                     ; LSS implies input NOT valid VMS code.
    50  01  D0          MOVL    #1,R0                   ; Setup for possible success return.
    03  51  D1          CMPL    R1,#3                   ; See if input in range.
        05  19          BLSS    20$                     ; LSS implies yes.

                10$:
    50      D4          CLRL    R0                      ; Indicate we picked up default.
    51  01  D0          MOVL    #1,R1                   ; Default is MSCP 1600 bpi.

                20$:
51  DA AF41             MOVZBW  TU_MSCPDENS[R1],R1      ; Extract MSCP code from array.
        05              RSB                             ; Return to caller.
```

```
                     0425  1595 ;+
                     0425  1596 ; MSCPTOVMS_DENS - Internal routine to convert from MSCP density code to
                     0425  1597 ;       VMS density code.
                     0425  1598 ;
                     0425  1599 ; Inputs:
                     0425  1600 ;       R0 = MSCP density code
                     0425  1601 ;
                     0425  1602 ; Outputs:
                     0425  1603 ;       R0 = VMS density code
                     0425  1604 ;-
                     0425  1605
                     0425  1606 MSCPTOVMS_DENS:
                     0425  1607
                     0425  1608         ASSUME  MSCP$V_TF_800   EQ      0
                     0425  1609         ASSUME  MSCP$V_TF_PE    EQ      1
                     0425  1610         ASSUME  MSCP$V_TF_GCR   EQ      2
50   50   03   00   EA 0425  1611         FFS     #0,#3,R0,R0             ; R0 contains 0, 1 or 2 (or 3 if not
                     042A  1612                                         ;  found).
     50   CB AF40   9A 042A  1613         MOVZBL  TU_VMSDENS[R0],R0       ; R0 contains system density code.
                     05 042F  1614         RSB                             ; Return to caller.
                     0430  1615
                     0430  1616 ;+
                     0430  1617 ; SPPEDTOMSCP - internal routine to calculate MSCP speed value.
                     0430  1618 ;
                     0430  1619 ; Inputs:
                     0430  1620 ;       R0 = Speed in IPS
                     0430  1621 ;       R1 = MSCP density value
                     0430  1622 ;
                     0430  1623 ; OUTPUTS:
                     0430  1624 ;       R0 = MSCP speed value
                     0430  1625 ;       R1 modified
                     0430  1626 ;-
                     0430  1627
                     0430  1628 SPEEDTOMSCP:
                     0430  1629
                     0430  1630         ASSUME  MSCP$V_TF_800   EQ      0
                     0430  1631         ASSUME  MSCP$V_TF_PE    EQ      1
                     0430  1632         ASSUME  MSCP$V_TF_GCR   EQ      2
51   51   03   00   EA 0430  1633         FFS     #0,#3,R1,R1             ; R1 contains 0, 1 or 2 (or 3 if not
                     0435  1634                                         ;  found).
     51   C7 AF41   3C 0435  1635         MOVZWL  TU_ABSDENS[R1],R1       ; R1 contains system density code.
          50   51   C4 043A  1636         MULL    R1,R0                   ; R0 contains absolute data rate.
50   000003E8 8F   C6 043D  1637         DIVL    #1000,R0                ; MSCP value is rate/1000.
                     05 0444  1638         RSB                             ; Return to caller.
                     0445  1639
                     0445  1640 ;+
                     0445  1641 ; MSCPTOSPEED - internal routine to convert MSCP data rate to speed in IPS.
                     0445  1642 ;
                     0445  1643 ; Inputs:
                     0445  1644 ;       R0 = MSCP Data Rate
                     0445  1645 ;       R1 = MSCP density value
                     0445  1646 ;
                     0445  1647 ; OUTPUTS:
                     0445  1648 ;       R0 = MSCP speed value
                     0445  1649 ;       R1 modified
                     0445  1650 ;-
                     0445  1651
```

```
                        0445  1652 MSCPTOSPEED:
                        0445  1653
                        0445  1654          ASSUME  MSCP$V_TF_800   EQ      0
                        0445  1655          ASSUME  MSCP$V_TF_PE    EQ      1
                        0445  1656          ASSUME  MSCP$V_TF_GCR   EQ      2
51   51   03   00   EA  0445  1657          FFS     #0,#3,R1,R1             ; R1 contains 0, 1 or 2 (or 3 if not
                        044A  1658                                          ;  found).
     51   B2 AF41   3C  044A  1659          MOVZWL  TU_ABSDENS[R1],R1       ; R1 contains system density code.
50    000003E8 8F   C4  044F  1660          MULL    #1000,R0                ; Multiply MSCP data rate by 1000.
          50   51   C6  0456  1661          DIVL    R1,R0                   ; Divide by density.
          50   05   C0  0459  1662          ADDL    #5,R0                   ; Round up.
                        045C  1663
                        045C  1664 ;        ASSUME  MT$S_SPEED      EQ      8
        51   A9 AF   9E  045C  1665          MOVAB   TU_ABSPEED,R1           ; R1 => Start of table.
                        0460  1666 10$:
           81   50   91  0460  1667          CMPB    R0,(R1)+                ; Find first entry > R0.
                FB   1A  0463  1668          BGTRU   10$                     ; If R0 >, loop back.
        50   FF A1   9A  0465  1669          MOVZBL  -1(R1),R0               ; Pickup previous value.
                     05  0469  1670          RSB                             ; Return to caller.
```

```
                     046A  1672          .SBTTL  SET_CLEAR_SEX
                     046A  1673
                     046A  1674 ;+
                     046A  1675 ; SET_CLEAR_SEX - internal subroutine to set (or not to set) the
                     046A  1676 ;         CLEAR_Serious_EXception modifier in an MSCP command.
                     046A  1677 ;         If the tape is NOT in Serious Exception mode, then this modifier
                     046A  1678 ;         is routinely set on each and every command.  If the tape IS in
                     046A  1679 ;         serious exception mode, then the modifier bit is only set if the
                     046A  1680 ;         QIO function code modifier IO$M_CLSEREXCP is specified on this
                     046A  1681 ;         QIO request.
                     046A  1682 ;
                     046A  1683 ;         Whether or not we are in Serious Exception mode is a function
                     046A  1684 ;         of how the tape was mounted and the state of a MT$M_ENSEREXCP bit
                     046A  1685 ;         in UCB$L_DEVDEPEND.
                     046A  1686 ;
                     046A  1687 ;         If the tape is MOUNTED ANSI, this implies that Serious Exception
                     046A  1688 ;         mode is enabled.  In other words, we are in Serious Exception mode
                     046A  1689 ;         if the volume is Mounted ANSI or if the MT$M_ENSEREXCP bit is on in
                     046A  1690 ;         UCB$L_DEVDEPEND.  If a tape is NOT mounted ANSI (i.e. either not
                     046A  1691 ;         mounted or mounted foreign) and MT$M_ENSEREXECP is not set then
                     046A  1692 ;         we implicitly insert a Clear Serious Exception modifier on each
                     046A  1693 ;         and every command.
                     046A  1694 ;
                     046A  1695 ; Inputs:
                     046A  1696 ;         R2 => MSCP command buffer
                     046A  1697 ;         R3 => UCB
                     046A  1698 ;         R5 => CDRP
                     046A  1699
                     046A  1700 SET_CLEAR_SEX:
                     046A  1701
        09   E0      046A  1702          BBS     #IO$V_CLSEREXCP,-       ; Branch to clear if clearing serious
0F C0   A5           046C  1703                  CDRP$Q_FUNC(R5),10$     ;  exception specified.
                     046F  1704
        02   E0      046F  1705          BBS     #MT$V_ENSEREXCP,-       ; Branch if Serious Exception explicitly
12 44   A3           0471  1706                  UCB$L_DEVDEPEND(R3),20$ ;  enabled.
        13   E1      0474  1707          BBC     #DEV$V_MNT,-            ; If Tape NOT mounted, go clear serious
05 38   A3           0476  1708                  UCB$L_DEVCHAR(R3),10$   ;  exception.
        18   E1      0479  1709          BBC     #DEV$V_FOR,-            ; Branch around Serious Exception
08 38   A3           047B  1710                  UCB$L_DEVCHAR(R3),20$   ;  clearing if tape MOUNTED ANSI.
                     047E  1711
                     047E  1712 10$:     ASSUME  MSCP$V_MD_CLSEX GE 8
        20   88      047E  1713          BISB    #<MSCP$M_MD_CLSEX@-8>,- ; Request clearing of possible Serious
   0B   A2           0480  1714                  MSCP$W_MODIFIER+1(R2)    ; Exception condition.
        01   8A      0482  1715          BICB    #MT$M_SEREXCP,-         ; Also explicitly clear software bit.
   44   A3           0484  1716                  UCB$L_DEVDEPEND(R3)
                     0486  1717
             05      0486  1718 20$:     RSB                             ; Return.
```

```
0487  1720          .IF     DF      TU_SEQCHK
0487  1721          .ALIGN  LONG,0
0487  1722  SEQ_MASK:
0487  1723          SEQFUNC <-                      ; SEQUENTIALFUNCTIONS
0487  1724                   UNLOAD,-               ;  Unload (make available + spindown)
0487  1725                   AVAILABLE,-            ;  Available (no spindown)
0487  1726                   SPACERECORD,-          ;  Space Records
0487  1727                   RECAL,-                ;  Recalibrate (REWIND)
0487  1728                   PACKACK,-              ;  Pack Acknowledge
0487  1729                   ERASETAPE,-            ;  Erase Tape (Erase Gap)
0487  1730                   SETCHAR,-              ;  Set Characteristics
0487  1731                   SETMODE,-              ;  Set Mode
0487  1732                   SPACEFILE,-            ;  Space File
0487  1733                   WRITECHECK,-           ;  Write Check
0487  1734                   READPBLK,-             ;  Read  PHYSICAL Block
0487  1735                   WRITEPBLK,-            ;  Write PHYSICAL Block
0487  1736                   READLBLK,-             ;  Read  LOGICAL  Block
0487  1737                   WRITELBLK,-            ;  Write LOGICAL  Block
0487  1738                   READVBLK,-             ;  Read  VIRTUAL  Block
0487  1739                   WRITEVBLK,-            ;  Write VIRTUAL  Block
0487  1740                   WRITEMARK,-            ;  Write Tape Mark
0487  1741                   DSE,-                  ;  Data Security Erase
0487  1742                   REWIND,-               ;  Rewind
0487  1743                   REWINDOFF,-            ;  Rewind AND Set Offline (UNLOAD)
0487  1744                   SKIPRECORD,-           ;  Skip Records
0487  1745                   SKIPFILE,-             ;  Skip Files
0487  1746                   WRITEOF>               ;  Write End Of File
0487  1747          .ENDC
```

```
                        0487  1749          .SBTTL  AUTO_PACKACK - Perform automatic PACKACK for foreign tapes
                        0487  1750 ;++
                        0487  1751 ;
                        0487  1752 ;       This code thread performs a gratuitous PACKACK for foreign mounted
                        0487  1753 ;       tapes.  It executes whenever an I/O request finds the volume valid bit
                        0487  1754 ;       clear, the tape at BOT, and the foreign mounted bit set.
                        0487  1755 ;
                        0487  1756 ;       The input CDRP is given a RSPID and a message buffer.  The message is
                        0487  1757 ;       inititalized.  This thread is then synchronized with the server so
                        0487  1758 ;       that this is the only thread communicating with the server.  Note:
                        0487  1759 ;       there is an implicit synchronization with other SEQNOP threads in that
                        0487  1760 ;       control cannot arrive here while other threads are synchronized by
                        0487  1761 ;       SEQNOP.
                        0487  1762 ;
                        0487  1763 ;       Once synchronization is established, ONLINE and GET UNIT STATUS
                        0487  1764 ;       commands are sent to the server.  This simulates an IO$_PACKACK.
                        0487  1765 ;       If either command fails, the I/O request is completed with a volume
                        0487  1766 ;       invalid error.  If both commands succeed, the device is marked volume
                        0487  1767 ;       valid and BOT.  The original request is requeued at the head of the
                        0487  1768 ;       pending I/O request queue and the SEQNOP condition is ended.  This
                        0487  1769 ;       restarts the original I/O request before any which may have
                        0487  1770 ;       accumulated while the automatic PACKACK was in progress.
                        0487  1771 ;
                        0487  1772 ;       All failures result in the unit being set MSCP AVAILABLE and the UCB
                        0487  1773 ;       being marked volume invalid.  Before completing the original I/O
                        0487  1774 ;       request, the error path also ends the SEQNOP condition.
                        0487  1775 ;--
                        0487  1776
                        0487  1777          .ENABLE LSB
                        0487  1778
                  010B   31   0487  1779 850$:    BRW     MSG_BUF_FAILURE         ; Branch assist.
                        048A  1780
                        048A  1781 AUTO_PACKACK:
                        048A  1782
                        048A  1783          .IIF    DF TU_SEQCHK, BSBW OVERRIDE_SEQCHK      ; Undo seq. checking.
                        048A  1784          ALLOC_RSPID                     ; Allocate RSPID.
                        0490  1785          ALLOC_MSG_BUF                   ; Allocate a message buffer.
                     F1 50   E9   0493  1786          BLBC    R0, 850$                ; Branch if connection broken.
                        0496  1787          INIT_MSCP_MSG ucb=(R3)          ; Initialize message buffer.
                        0499  1788          START_SEQNCP                    ; Synchronize with server.
                        04AF  1789
                  08 A2   09   90   04AF  1790          MOVB    #MSCP$K_OP_ONLIN, -     ; ONLINE command.
                        04B3  1791                  MSCP$B_OPCODE(R2)
               0A A2   2020 8F   A8   04B3  1792          BISW    #<MSCP$M_MD_CLSEX -     ; Do exclusive ONLINE and clear serious
                        04B9  1793                    !MSCP$M_MD_EXCLU>, -  ; exception.
                        04B9  1794                  MSCP$W_MODIFIER(R2)
               0E A2   00E0 C3   B0   04B9  1795          MOVW    UCB$W_UNIT_FLAGS(R3), - ; Copy UNIT flags to MSCP packet.
                        04BF  1796                  MSCP$W_UNT_FLGS(R2)
                     00D8 C3   D0   04BF  1797          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
                       1C A2        04C3  1798                  MSCP$L_DEV_PARM(R2)     ; MSCP packet.
          50   44 A3   05   08   EF   04C5  1799          EXTZV   #MT$V_DENSITY, -        ; Determine density that the user has
                        04CB  1800                  #MT$S_DENSITY, -        ; last established for this unit
                        04CB  1801                  UCB$L_DEVDEPEND(R3), R0 ; and put into R0.
                        FF3E   30   04CB  1802          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
                     20 A2    51   B0   04CE  1803          MOVW    R1, MSCP$W_FORMAT(R2)   ; Move MSCP density in R1 into packet.
                  0D 0E A2    05   E1   04D2  1804          BBC     #MSCP$V_UF_VSMSU, -     ; Test if we are suppressing variable
                        04D7  1805                  MSCP$W_ONT_FLGS(R2), 10$; speed mode, and branch if NOT.
```

```
                18   EF  04D7  1806          EXTZV   #MT$V_SPEED,-            ; Extract user's speed specification
                08       04D9  1807                  #MT$S_SPEED,-            ; from UCB.
        50  44 A3       04DA  1808                  UCB$L_DEVDEPEND(R3), R0
          FF50   30     04DD  1809          BSBW    SPEEDTOMSCP
    22 A2   50   B0     04E0  1810          MOVW    R0, MSCP$W_SPEED(R2)     ; Move MSCP speed in R0 into packet.
                        04E4  1811 10$:     SEND_MSCP_MSG                    ; ONLIN - returns end pkt. addr. in R2.
                        04E7  1812          ASSUME  CDRP$V_CAND EQ 0
      47 40 A5   E8     04E7  1813          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Has operation been canceled?
                        04EB  1814                  900$                     ; Branch if operation canceled.
                        04EB  1815          IF_MSCP FAILURE, then=900$       ; Branch if ONLIN failed.
                        04F1  1816
                        04F1  1817          ; The various fields in the END PACKET are valid and the tape is
                        04F1  1818          ; ONLINE.
                        04F1  1819
            02A1   30   04F1  1820          BSBW    RECORD_ONLINE            ; Move data from end message to UCB.
                        04F4  1821
                        04F4  1822          RESET_MSCP_MSG                   ; Setup message buf. etc. for reuse.
    08 A2   03   90     04F7  1823          MOVB    #MSCP$K_OP_GTUNT, -      ; GET UNIT STATUS command.
                        04FB  1824                  MSCP$B_OPCODE(R2)
                        04FB  1825          SEND_MSCP_MSG                    ; GTUNT - returns end pkt. addr. in R2.
                        04FE  1826          ASSUME  CDRP$V_CAND EQ 0
      30 40 A5   E8     04FE  1827          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Has operation been canceled?
                        0502  1828                  900$                     ; Branch if operation canceled.
                        0502  1829          IF_MSCP FAILURE, then=900$       ; Branch if GTUNT failed.
                        0508  1830
            0298   30   0508  1831          BSBW    RECORD_GETUNIT_CHAR      ; Record UNIT status data in UCB.
                        050B  1832
                        050B  1833          ASSUME  UCB$V_VALID GE 8
    65 A3   08   88     050B  1834          BISB    #<UCB$M_VALID @ -8>, -   ; Make unit volume valid.
                        050F  1835                  UCB$W_STS+1(R3)
                        050F  1836          ASSUME  MT$V_BOT GE 16
       46 A3   01   88  050F  1837          BISB    #<MT$M_BOT @ -16>, -     ; Set beginning of tape.
                        0513  1838                  UCB$L_DEVDEPEND+2(R3)
             FAEA'  30  0513  1839          BSBW    DUTU$DEALLOC_ALL         ; Release all SCS resources.
    4C A3   A0 A5   0E  0516  1840          INSQUE  CDRP$L_IOQFL(R5), -      ; Put this request at the head of
                        051B  1841                  UCB$L_IOQFL(R3)          ; the pending I/O queue.
                        051B  1842          END_SEQNOP                       ; End the sequential NOP state.
                    05  0531  1843          RSB                              ; Kill this thread.
                        0532  1844
                        0532  1845          ; Something went wrong during auto PACKACK.  Fail the I/O request.
                        0532  1846
                        0532  1847 900$:     ASSUME  UCB$V_VALID GE 8
       65 A3   08   AA  0532  1848          BICW    #<UCB$M_VALID @ -8>, -   ; Clear unit volume valid.
                        0536  1849                  UCB$W_STS+1(R3)
    03 0A A2   07   E1  0536  1850          BBC     #MSCP$V_SC_DUPUN, -      ; Branch around if NOT duplicate
                        053B  1851                  MSCP$W_STATUS(R2), 940$  ; unit substatus.
             FAC2'  30  053B  1852          BSBW    DUTU$SEND_DUPLICATE_UNIT ; Notify operator of duplicate unit.
                        053E  1853 940$:     RESET_MSCP_MSG                   ; Setup message buf. etc. for reuse.
       08 A2   08   90  0541  1854          MOVB    #MSCP$K_OP_AVAIL, -      ; Setup available command.
                        0545  1855                  MSCP$B_OPCODE(R2)
                        0545  1856          SEND_MSCP_MSG                    ; AVAIL - returns end pkt. addr. in R2.
                        0548  1857          END_SEQNOP                       ; End the sequential NOP state.
     50   0254 8F   3C  055E  1858          MOVZWL  #SS$_VOLINV, R0          ; Set volume invalid status.
          03 40 A5   E9 0563  1860          BLBC    CDRP$L_DUTUFLAGS(R5), - ; But, if operation was canceled,
                        0567  1861                  950$                     ; use "aborted" status instead.
          50   2C   3C  0567  1862          MOVZWL  #SS$_ABORT, R0
```

```
075B   31  056A  1863 950$:   BRW     FUNCTION_EXIT          ; Terminate origianl I/O request.
           056D  1864
           056D  1865         .DISABLE LSB
```

```
                     056D  1867          .SBTTL  START I/O
                     056D  1868 ;
                     056D  1869 ;+
                     056D  1870 ;
                     056D  1871 ;  Beginning of out of line code to deal with problems that
                     056D  1872 ;        may occur in the common STARTIO code on the next page.
                     056D  1873 ;
                     056D  1874 LOCAL_DEVICE:
 55   00A8 C5   D0  056D  1875          MOVL    UCB$L_2P_ALTUCB(R5),R5  ; R5 => local UCB.
  00000000'GF   17  0572  1876          JMP     G^EXE$INSIOQ            ; Go hand this IRP to local driver.
                     0578  1877 ;
                     0578  1878 ;
                     0578  1879 ; Out of line code to handle Volume Invalid.
                     0578  1880 ;
                     0578  1881 ;
                     0578  1882 VOL_INVALID:
                     0578  1883 ;
09 38 A3    18   E1  0578  1884          BBC     #DEV$V_FOR, -           ; Branch if device is not foreign
                     057D  1885                  UCB$L_DEVCHAR(R3), 10$  ; mounted.
      00B0 C3   D5  057D  1886          TSTL    UCB$L_RECORD(R3)        ; Is device at beginning of tape?
           03   12  0581  1887          BNEQ    10$                     ; Branch if device not at BOT.
         FF04   31  0583  1888          BRW     AUTO_PACKACK            ; Else, go issue gratuitous PACKACK.
           08   E0  0586  1889 10$:     BBS     #IRP$V_PHYSIO,-         ; See if PHYSICAL I/O requested.
        CA A5       0588  1890                  CDRP$W_STS(R5),-        ;  If physical, then branch back to
           53       058A  1891                  PHYIO_VOLINV            ;  continue even tho VOLINV.
                     058B  1892          .IF     DF      TU_SEQCHK
                     058B  1893          BSBW    OVERRIDE_SEQCHK         ; Override sequence checking and
                     058B  1894                                          ;  remove sequence # from array.
                     058B  1895          .ENDC
                     058B  1896 ;
 50   0254 8F   3C  058B  1897          MOVZWL  #SS$_VOLINV,R0          ; Indicate error status.
           51   D4  0590  1898          CLRL    R1                      ; Clear second word of I/O status.
         0733   31  0592  1899          BRW     FUNCTION_EXIT           ; GOTO common exit.
                     0595  1900 ;
                     0595  1901 ;
                     0595  1902 ;
                     0595  1903 MSG_BUF_FAILURE:
                     0595  1904 ;
                     0595  1905 ; We are here only if we had an allocation failure on the Message Buffer.
                     0595  1906 ;  This implies that our CONNECTION to the MSCP server is broken.  The action
                     0595  1907 ;  to be taken is to kill this thread of execution since we are guaranteed
                     0595  1908 ;  that a thread exists that is currently executing that is gathering all
                     0595  1909 ;  CDRP's associated with this CONNECTION.  So we branch to KILL_THIS_THREAD.
                     0595  1910 ;
         FA68'  31   0595  1911          BRW     DUTU$KILL_THIS_THREAD   ; Branch to where we collect all active
                     0598  1912                                          ;  CDRP's prior to re-CONNECTION.
                     0598  1913 ;
                     0598  1914 ;  End of out of line code
                     0598  1915 ;-
```

```
                                0598  1917 TU_STARTIO:
                                0598  1918         ASSUME  UCB$V_BSY GE 8
                     65 A5  01  8A 0598  1919         BICB    #<UCB$M_BSY @ -8>, -    ; Undo bit setting so that multiple
                                059C  1920                 UCB$W_STS+1(R5)         ; IRP's can be started.
                                059C  1921
                                059C  1922 ; If this UCB indicates that the device is a local (non-MSCP) device that
                                059C  1923 ; has also been made available to us via 1) dual porting and 2) an MSCP
                                059C  1924 ; Server on the node to which it is dual ported, then shunt this IRP to
                                059C  1925 ; the local driver.
                                059C  1926
                          03    E0 059C  1927         BBS     #DEV$V_CDP,-            ; This bit, if clear indicates that
                        3C A5       059E  1928                 UCB$L_DEVCHAR2(R5),-    ;  the above condition is NOT true,
                          CC        05A0  1929                 LOCAL_DEVICE            ;  so branch out of line if set.
                     50   60 A3  9E 05A1  1930         MOVAB   -CDRP$L_IOQFL(R3),R0    ; Get address of CDRP portion of IRP.
                                05A5  1931
                                05A5  1932         ASSUME  CDRP$B_CD_TYPE EQ CDRP$W_CDRPSIZE+2
                                05A5  1933         ASSUME  CDRP$B_FIPL    EQ CDRP$W_CDRPSIZE+3
                08 A0 0839FFA0  8F  D0 05A5  1934         MOVL    #< <IPL$_SCS@24> -      ; Initialize CDRP size, type and fork
                                05AD  1935                  ! <DYN$C_CDRP@16> -    ; IPL fields.
                                05AD  1936                  ! <CDRP$L_IOQFL&^xFFFF> >, -
                                05AD  1937                 CDRP$W_CDRPSIZE(R0)
                                05AD  1938
                                05AD  1939         ASSUME  CDRP$L_RSPID    EQ      CDRP$L_MSG_BUF+4
                        1C A0   7C 05AD  1940         CLRQ    CDRP$L_MSG_BUF(R0)      ; Prevent spurious DEALLOC_MSG_BUF and
                                05B0  1941                                         ;  also spurious DEALLOC_RSPID.
                        2C A0   D4 05B0  1942         CLRL    CDRP$L_LBUFH_AD(R0)     ; Prevent spurious UNMAP.
                        56 A5   9E 05B3  1943         MOVAB   UCB$W_RWAITCNT(R5),-    ; Point CDRP field to UCB field.
                        28 A0      05B6  1944                 CDRP$L_RWCPTR(R0)
                        40 A0   D4 05B8  1945         CLRL    CDRP$L_DUTUFLAGS(R0)    ; Initialize class driver flags.
                        56 A5   B5 05BB  1946         TSTW    UCB$W_RWAITCNT(R5)      ; See if any IRP's currently waiting
                                05BE  1947                                         ;  for resources.
                           05   13 05BE  1948         BEQL    TU_REAL_STARTIO         ; EQL implies NO, so GOTO real STARTIO.
                           63   0E 05C0  1949         INSQUE  IRP$L_IOQFL(R3),-       ; To force sequential submission of commands
                     50 B5           05C2  1950                 @UCB$L_IOQBL(R5)        ;  to intelligent controller, we force
                                05C4  1951                                         ;  IRP's to be queued up here if any
                                05C4  1952                                         ;  previous request is possibly hungup
                                05C4  1953                                         ;  waiting for resources between the
                                05C4  1954                                         ;  beginning of STARTIO and the SEND_MSG_BUF
                              05    05 05C4  1955         RSB                     ; Return to caller (QIO system service)
                                05C5  1956
                                05C5  1957 TU_REAL_STARTIO:
                                05C5  1958
                                05C5  1959         .IF     DF      TU_TRACE
                                05C5  1960         BSBW    TRACE_IRP               ; Trace IRP.
                                05C5  1961         MOVAB   -CDRP$L_IOQFL(R3),R0    ; Refresh R0=CDRP if tracing.
                                05C5  1962         .ENDC
                                05C5  1963
                     53   55   D0 05C5  1964         MOVL    R5,R3                   ; Let R3 => UCB.
                     55   50   D0 05C8  1965         MOVL    R0,R5                   ; R5 => CDRP.
                                05CB  1966
                                05CB  1967         .IF     DF      TU_SEQCHK
                                05CB  1968         EXTZV   #IRP$V_FCODE,-          ; Extract I/O function code.
                                05CB  1969                 #IRP$S_FCODE,-
                                05CB  1970                 CDRP$W_FUNC(R5),R1
                                05CB  1971         BBC     R1,SEQ_MASK,TU_RESTARTIO; If non-Sequential I/O branch around.
                                05CB  1972         EXTZV   #0,-                    ; Extract six bit index into array of
                                05CB  1973                 #6,-                    ;  IRP sequence number slots.  R1 =
```

```
                        05CB  1974                    UCB$B_TU_NEWINX(R3),R1  ;  index of next available slot.
                        05CB  1975            INCB    UCB$B_TU_NEWINX(R3)     ; Increment index.
                        05CB  1976            MOVL    CDRP$L_SEQNUM(R5),-     ; Copy sequnce number of this IRP to
                        05CB  1977                    UCB$L_TU_SEQARY(R3)[R1] ;  circular ring slot.
                        05CB  1978            .ENDC
                        05CB  1979
                        05CB  1980 TU_RESTARTIO:                              ; Label where we RESTART CDRP's after
                        05CB  1981                                            ;  virtual circuit re-CONNECTION.
                        05CB  1982
              00C8 C3   D0  05CB  1983            MOVL    UCB$L_CDT(R3),-     ; Place CDT pointer into CDRP for handy
                24 A5       05CF  1984                    CDRP$L_CDT(R5)      ;  reference by SCS routines. Note we
                        05D1  1985                                            ;  do this after label TU_RESTARTIO so
                        05D1  1986                                            ;  that it is refreshed upon restart.
        54      0084 C3   D0  05D1  1987            MOVL    UCB$L_PDT(R3),R4    ; R4 => port's PDT.
                        05D6  1988
     03 64 A3     0B   E0  05D6  1989            BBS     #UCB$V_VALID, -     ; Branch if unit is volume valid.
                        05DB  1990                    UCB$W_STS(R3), PHYIO_VOLINV
                 FF9A   31  05DB  1991            BRW     VOL_INVALID         ; Else, branch to out of line
                        05DE  1992                                            ; volume invalid processing.
                        05DE  1993
                        05DE  1994 PHYIO_VOLINV:
                        05DE  1995            ALLOC_RSPID                     ; ALLOCate a ReSPonse ID.
                        05E4  1996            ALLOC_MSG_BUF                   ; Allocate an MSCP buffer (and also
                        05E7  1997                                            ;  allocate a unit of flow control).
                   AB 50  E9  05E7  1998            BLBC    R0,MSG_BUF_FAILURE  ; If failure, branch out of line.
                        05EA  1999
                        05EA  2000 ; Here a little common MSCP packet initialization.
                        05EA  2001
                50   52   D0  05EA  2002            MOVL    R2, R0              ; Copy message buffer address.
                        05ED  2003            .REPEAT MSCP$K_MXCMDLEN / 8
                        05ED  2004            CLRQ    (R0)+                   ; Zero entire message buffer.
                     80 7C  05ED  2005            .ENDR
                     80 D4  05F5  2006            .IIF    NE MSCP$K_MXCMDLEN & 4, CLRL (R0)+
                        05F7  2007            .IIF    NE MSCP$K_MXCMDLEN & 2, CLRW (R0)+
                        05F7  2008            .IIF    NE MSCP$K_MXCMDLEN & 1, CLRB (R0)+
                        05F7  2009
                   20 A5   D0  05F7  2010            MOVL    CDRP$L_RSPID(R5),-  ; Use RSPID as command reference
                      62       05FA  2011                    MSCP$L_CMD_REF(R2)  ;  number for all commands.
                        05FB  2012
                00D4 C3   B0  05FB  2013            MOVW    UCB$W_MSCPUNIT(R3),- ; Indicate UNIT number in MSCP
                  04 A2       05FF  2014                    MSCP$W_UNIT(R2)     ;  packet.
                        0601  2015
                        0601  2016 TU_BEGIN_IVCMD:
                        0601  2017 TU_REDO_IO:
                        0601  2018
                    FE66   30  0601  2019            BSBW    SET_CLEAR_SEX       ; Go set state of Clear Serious EXception.
                      0F   E1  0604  2020            BBC     #IO$V_INHRETRY,-    ; Branch around if NOT inhibiting RETRY.
                04 C0 A5       0606  2021                    CDRP$W_FUNC(R5),30$
                        0609  2022            ASSUME  MSCP$V_MD_SEREC GE 8    ; Else, set the suppress error
            0B A2    01   88  0609  2023            BISB    #<MSCP$M_MD_SEREC@-8>, -; modifier.
                        060D  2024                    MSCP$W_MODIFIER+1(R2)
                        060D  2025 30$:
                      00 EF  060D  2026            EXTZV   #IRP$V_FCODE,-      ; Extract I/O function code.
                      06     060F  2027                    #IRP$S_FCODE,-
            51    C0 A5       0610  2028                    CDRP$W_FUNC(R5),R1
                        0613  2029
                        0613  2030            DISPATCH R1, type=B, prefix=IO$_, < -          ; Dispatch to correct
```

```
                     0613  2031                <NOP,           START_NOP>, -           ; function processing.
                     0613  2032                <PACKACK,       START_PACKACK>, -
                     0613  2033                <UNLOAD,        START_UNLOAD>, -
                     0613  2034                <AVAILABLE,     START_AVAILABLE>, -
                     0613  2035                <REWIND,        START_REWIND>, -
                     0613  2036                <REWINDOFF,     START_REWINDOFF>, -
                     0613  2037                <READPBLK,      START_READPBLK>, -
                     0613  2038                <WRITECHECK,    START_WRITECHECK>, -
                     0613  2039                <WRITEPBLK,     START_WRITEPBLK>, -
                     0613  2040                <WRITEMARK,     START_WRITEMARK>, -
                     0613  2041                <WRITEOF,       START_WRITEOF>, -
                     0613  2042                <SPACEFILE,     START_SPACEFILE>, -
                     0613  2043                <SKIPFILE,      START_SKIPFILE>, -
                     0613  2044                <SPACERECORD,   START_SPACERECORD>, -
                     0613  2045                <SKIPRECORD,    START_SKIPRECORD>, -
                     0613  2046                <RECAL,         START_RECAL>, -
                     0613  2047                <ERASETAPE,     START_ERASETAPE>, -
                     0613  2048                <DSE,           START_DSE>, -
                     0613  2049                <SENSECHAR,     START_SENSECHAR>, -
                     0613  2050                <SENSEMODE,     START_SENSEMODE>, -
                     0613  2051                <SETCHAR,       START_SETCHAR>, -
                     0613  2052                <SETMODE,       START_SETMODE> -
                     0613  2053                >
                     0669  2054
                     0669  2055        ; Function code is not legal.
                     0669  2056
          F994'  30  0669  2057        BSBW    DUTU$RESTORE_CREDIT     ; Restore allocated send credit.
50   00F4 8F   3C  066C  2058        MOVZWL  #SS$_ILLIOFUNC,R0
            51   D4  0671  2059        CLRL    R1
          0652   31  0673  2060        BRW     FUNCTION_EXIT           ; Branch to exit I/O function.
```

```
                     0676  2062          .SBTTL  START_NOP
                     0676  2063 ; START_NOP - Prepare an MSCP packet to do a GET UNIT STATUS command.
                     0676  2064 ;
                     0676  2065 ; INPUTS:
                     0676  2066 ;       R2 => MSCP buffer
                     0676  2067 ;       R3 => UCB
                     0676  2068 ;       R4 => PDT
                     0676  2069 ;       R5 => CDRP
                     0676  2070 ;
                     0676  2071 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     0676  2072 ;
                     0676  2073
                     0676  2074 START_NOP:
   03    90          0676  2075          MOVB    #MSCP$K_OP_GTUNT,-      ; Transfer GET UNIT STATUS opcode
08 A2                0678  2076                  MSCP$B_OPCODE(R2)       ;  to packet.
                     067A  2077          ASSUME  MSCP$V_MD_CLSEX GE 8
   20    8A          067A  2078          BICB    #<MSCP$M_MD_CLSEX@-8>,- ; The clear serious execption modifier
0B A2                067C  2079                  MSCP$W_MODIFIER+1(R2)   ; is illegal on get unit status cmds.
                     067E  2080
                     067E  2081          IF_IVCMD then=NOP_IVCMD_END     ; Branch if invalid command processing.
                     0682  2082
                     0682  2083          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     0685  2084
                     0685  2085          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                     0688  2086          ACTION_ENTRY    SUCC,  SS$_NORMAL,      NOP_SUCC
                     068D  2087          ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  NOP_OFFLINE
                     0692  2088          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      NOP_AVAIL
                     0697  2089          ACTION_ENTRY    DRIVE, SS$_DRVERR,      NOP_DRVERR
                     069C  2090          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     NOP_CTRLERR
                     06A1  2091          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     NOP_IVCMD
                     06A6  2092          ACTION_ENTRY    END_TABLE
                     06A8  2093
 09CE    31          06A8  2094          BRW     INVALID_STS             ; Unexpected MSCP end status.
                     06AB  2095
                     06AB  2096 NOP_IVCMD:
                     06AB  2097          IVCMD_BEGIN                     ; Begin invalid command processing.
 FF50    31          06AE  2098          BRW     TU_BEGIN_IVCMD          ; Replicate building MSCP command.
                     06B1  2099 NOP_IVCMD_END:
                     06B1  2100          IVCMD_END                       ; Complete invalid command processing.
                     06B3  2101 ; ----- BRB     NOP_SUCC                ; Fall through to complete command.
                     06B3  2102
                     06B3  2103
                     06B3  2104 NOP_SUCC:
                     06B3  2105 NOP_OFFLINE:
                     06B3  2106 NOP_AVAIL:
                     06B3  2107 NOP_CTRLERR:
                     06B3  2108 NOP_DRVERR:
                     06B3  2109 ;NOP_END:
   51    D4          06B3  2110          CLRL    R1                      ; Clear for I/O status block.
 0610    31          06B5  2111          BRW     FUNCTION_EXIT           ; Branch to common exit.
```

```
                     06B8  2114          .SBTTL  START_PACKACK
                     06B8  2115
                     06B8  2116 ; START_PACKACK - Prepare an MSCP packet to do an ONLINE command.
                     06B8  2117 ;
                     06B8  2118 ; INPUTS:
                     06B8  2119 ;       R2 => MSCP buffer
                     06B8  2120 ;       R3 => UCB
                     06B8  2121 ;       R4 => PDT
                     06B8  2122 ;       R5 => CDRP
                     06B8  2123 ;
                     06B8  2124 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     06B8  2125 ;
                     06B8  2126
                     06B8  2127 START_PACKACK:
                     06B8  2128
        09   90      06B8  2129          MOVB    #MSCP$K_OP_ONLIN,-      ; Transfer ONLINE opcode
    08 A2            06BA  2130                  MSCP$B_OPCODE(R2)       ;  to packet.
                     06BC  2131
 50 00BC C3   D0     06BC  2132          MOVL    UCB$L_CDDB(R3), R0      ; Get CDDB address.
04 28 A0   02   E1   06C1  2133          BBC     #MSCP$V_CF_MLTHS, -     ; Branch if not a multi-host server.
                     06C6  2134                  CDDB$W_CNTRLFLGS(R0), 20$
        20   A8      06C6  2135          BISW    #MSCP$M_MD_EXCLU,-      ; Do exclusive ONLINE.
    0A A2            06C8  2136                  MSCP$W_MODIFIER(R2)
                     06CA  2137
0E A2 00E0 C3   B0   06CA  2138 20$:    MOVW    UCB$W_UNIT_FLAGS(R3), - ; Copy unit flags to MSCP packet.
                     06D0  2139                  MSCP$W_UNT_FLGS(R2)
                     06D0  2140
        00D8 C3   D0 06D0  2141          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
          1C A2      06D4  2142                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                     06D6  2143
          08   EF    06D6  2144          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
          05         06D8  2145                  #MT$S_DENSITY,-         ;  last established for this unit
50    44 A3         06D9  2146                  UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
       FD2D   30    06DC  2147          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
20 A2    51   B0    06DF  2148          MOVW    R1,MSCP$W_FORMAT(R2)    ; Move MSCP density in R1 into packet.
                     06E3  2149
                     06E3  2150          IF_IVCMD then=PACKACK_IVCMD_END ; Branch if invalid command processing.
                     06E7  2151
                     06E7  2152          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     06EA  2153
                     06EA  2154          ASSUME  UCB$V_VALID GE 8
65 A3    08   8A     06EA  2155          BICB    #<UCB$M_VALID @ -8>, -  ; Initialize software volume invalid.
                     06EE  2156                  UCB$W_STS+1(R3)
                     06EE  2157
                     06EE  2158          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                     06F1  2159          ACTION_ENTRY    SUCC,  SS$_NORMAL,      PACKACK_SUCC
                     06F6  2160          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      PACKACK_OFFLINE
                     06FB  2161          ACTION_ENTRY    ABRTD, SS$_ABORT,       END_PACKACK
                     0700  2162          ACTION_ENTRY    DRIVE, SS$_DRVERR,      END_PACKACK
                     0705  2163          ACTION_ENTRY    FMTER, SS$_CTRLERR,     END_PACKACK
                     070A  2164          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     END_PACKACK
                     070F  2165          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     PACKACK_IVCMD
                     0714  2166          ACTION_ENTRY    END_TABLE
                     0716  2167
        0960   31    0716  2168          BRW     INVALID_STS             ; Unexpected MSCP end status.
                     0719  2169
                     0719  2170
```

```
                     0719  2171 PACKACK_SUCC:                           ; Action routine for MSCP$K_ST_SUCC.
                     0719  2172
                     0719  2173          ASSUME  CDRP$V_CAND EQ 0
   24 40 A5    E8    0719  2174          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                     071D  2175                  890$                    ; Branch if request was canceled.
         08    E0    071D  2176          BBS     #MSCP$V_SC_ALONL,-      ; Branch around clearing of TU_RECORD
   0C 0A A2          071F  2177                  MSCP$W_STATUS(R2),10$   ;  if REDUNDANT ONLINE.
    00B0 C3    D4    0722  2178          CLRL    UCB$L_RECORD(R3)        ; Successful exclusive ONLINE rewinds
                     0726  2179          ASSUME  MT$V_BOT  GE 16
                     0726  2180          ASSUME  MT$V_EOF  GE 16
                     0726  2181          ASSUME  MT$V_EOT  GE 16
                     0726  2182          ASSUME  MT$V_LOST GE 16
   46 A3    16    8A    0726  2183          BICB    #<<MT$M_EOF ! MT$M_EOT -; Clear position sensitive DEVDEPEND
                     072A  2184                   ! MT$M_LOST> @ -16>, - ; bits.
                     072A  2185                  UCB$L_DEVDEPEND+2(R3)
   46 A3    01    88    072A  2186          BISB    #<MT$M_BOT @ -16>, -    ; Set BOT DEVDEPEND position bit.
                     072E  2187                  UCB$L_DEVDEPEND+2(R3)
                     072E  2188 10$:
         0064    30    072E  2189          BSBW    RECORD_ONLINE           ; Record ONLINE data in UCB.
                     0731  2190
                     0731  2191 ; Here having done an ONLINE we proceed to do a GET UNIT STATUS.
                     0731  2192
                     0731  2193          RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
      03    90    0734  2194          MOVB    #MSCP$K_OP_GTUNT,-      ; Opcode is for GET UNIT STATUS.
      08 A2          0736  2195                  MSCP$B_OPCODE(R2)
                     0738  2196          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     073B  2197
                     073B  2198          IF_MSCP SUCCESS, then=PACKACK_GTUNT_SUCC  ; Branch if GTUNT successful.
                     0741  2199          ASSUME  CDRP$V_CAND EQ 0
   3A 40 A5    E8    0741  2200 890$:    BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                     0745  2201                  PACKACK_CANCEL          ; Branch if request was canceled.
                     0745  2202          RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
       FEB6    31    0748  2203          BRW     TU_REDO_IO              ; Go try again.
                     074B  2204
                     074B  2205 PACKACK_GTUNT_SUCC:
                     074B  2206
         56    10    074B  2207          BSBB    RECORD_GETUNIT_CHAR     ; Record unit status data in UCB.
                     074D  2208
      50    01    3C    074D  2209          MOVZWL  #SS$_NORMAL, R0         ; Set success IOSB status.
         3C    11    0750  2210          BRB     VALID_PACKACK           ; And branch around to success.
                     0752  2211
                     0752  2212   CKACK_IVCMD:
                     0752  2213          IVCMD_BEGIN                     ; Begin invalid command processing.
       FEA9    31    0755  2214          BRW     TU_BEGIN_IVCMD          ; Repeat commands that formed MSCP cmd.
                     0758  2215 PACKACK_IVCMD_END:
                     0758  2216          IVCMD_END                       ; Complete invalid command processing.
         36    11    075A  2217          BRB     END_PACKACK             ; Branch around to end.
                     075C  2218
                     075C  2219 PACKACK_OFFLINE:
                     075C  2220
         07    E1    075C  2221          BBC     #MSCP$V_SC_DUPUN,-      ; Branch around if NOT duplicate
   12 0A A2          075E  2222                  MSCP$W_STATUS(R2),20$   ;  unit substatus.
         55    DD    0761  2223          PUSHL   R5                      ; Save R5.
      55    53    D0    0763  2224          MOVL    R3,R5                   ; R5 => UCB for subroutine.
          F897'    30    0766  2225          BSBW    DUTU$SEND_DUPLICATE_UNIT; Send a message to the operator.
            55 8ED0    0769  2226          POPL    R5                      ; Restore R5.
   50    21C4 8F    3C    076C  2227          MOVZWL  #SS$_DUPUNIT,R0         ; Return final status.
```

```
                1F   11  0771  2228          BRB     END_PACKACK             ; Branch around.
                                0773  2229 20$:
                06   E1  0773  2230          BBC     #MSCP$V_SC_INOPR,-      ; Branch around if NOT unit inoperative
              0A A2          0775  2231                  MSCP$W_STATUS(R2),-     ;  substatus.
                1A           0777  2232                  END_PACKACK
        50    008C 8F   3C  0778  2233          MOVZWL  #SS$_DRVERR,R0          ; Return final status.
                13   11  077D  2234          BRB     END_PACKACK             ; Branch around.
                                077F  2235
                                077F  2236 PACKACK_CANCEL:
                                077F  2237
                                077F  2238          RESET_MSCP_MSG                  ; Ready message for a new MSCP command.
           08 A2     08   90  0782  2239          MOVB    #MSCP$K_OP_AVAIL, -     ; Undo online with available command.
                                0786  2240                  MSCP$B_OPCODE(R2)
                                0786  2241          SEND_MSCP_MSG                   ; Sent AVAILABLE to the server.
              50    2C   3C  0789  2242          MOVZWL  #SS$_ABORT, R0          ; Signal request was canceled.
                04   11  078C  2243          BRB     END_PACKACK             ; Exit function.
                                078E  2244
                                078E  2245 VALID_PACKACK:
                                078E  2246
                                078E  2247          ASSUME  UCB$V_VALID GE 8
           65 A3     08   88  078E  2248          BISB    #<UCB$M_VALID @ -8>, -  ; Set software volume valid.
                                0792  2249                  UCB$W_STS+1(R3)
                                0792  2250 END_PACKACK:
                0533   31  0792  2251          BRW     FUNCTION_EXIT
```

```
                     0795  2253          .SBTTL  PACKACK Support Routines
                     0795  2254
                     0795  2255 ;+
                     0795  2256 ; RECORD_ONLINE - copy data from ONLINE END MESSAGE to UCB.
                     0795  2257 ; RECORD_SETUNIT_CHAR - copy data from SET UNIT CHAR End Message to UCB.
                     0795  2258 ; RECORD_GETUNIT_CHAR - copy data from GET UNIT CHAR End Message to UCB.
                     0795  2259 ;
                     0795  2260 ; Inputs:
                     0795  2261 ;       R2 => End Message
                     0795  2262 ;       R3 => UCB
                     0795  2263 ;
                     0795  2264 ; Outputs:
                     0795  2265 ;       R1 corrupted.
                     0795  2266 ;       All other registers preserved.
                     0795  2267 ;
                     0795  2268 ;       UCB fields set
                     0795  2269 ;-
                     0795  2270
                     0795  2271 RECORD_ONLINE:
                     0795  2272 RECORD_SETUNIT_CHAR:
                     0795  2273
  24 A2    D0        0795  2274          MOVL    MSCP$L_MAXWTREC(R2),-   ; Copy maximum recommended write
00EC C3              0798  2275                  UCB$L_TU_MAXWRCNT(R3)   ;  record size to UCB
  28 A2    B0        079B  2276          MOVW    MSCP$W_NOISEREC(R2),-   ; Copy size of noise records to UCB.
00F4 C3              079E  2277                  UCB$W_TU_NOISE(R3)
    07    11         07A1  2278          BRB     RECORD_COMMON           ; Join common "record" processing.
                     07A3  2279
                     07A3  2280 RECORD_GETUNIT_CHAR:
                     07A3  2281
                     07A3  2282          ASSUME  MT$V_SUP_NRZI EQ 21
                     07A3  2283          ASSUME  MSCP$V_TF_800 EQ  0
                     07A3  2284          ASSUME  MT$V_SUP_PE   EQ 22
                     07A3  2285          ASSUME  MSCP$V_TF_PE  EQ  1
                     07A3  2286          ASSUME  MT$V_SUP_GCR  EQ 23
                     07A3  2287          ASSUME  MSCP$V_TF_GCR EQ  2
44 A3    03    15   24 A2    F0  07A3  2288          INSV    MSCP$W_FORMENU(R2), -   ; Copy supported tape densities to
                     07AA  2289                  #MT$V_SUP_NRZI, #3, -   ; DEVDEPEND.
                     07AA  2290                  UCB$L_DEVDEPEND(R3)
                     07AA  2291
                     07AA  2292 RECORD_COMMON:
                     07AA  2293
          50    DD   07AA  2294          PUSHL   R0                      ; Save R0.
       14 A2    7D   07AC  2295          MOVQ    MSCP$Q_UNIT_ID(R2),-    ; In the event of success, copy unit
00CC C3              07AF  2296                  UCB$Q_UNIT_ID(R3)       ;  characteristics data to UCB.
       1C A2    D0   07B2  2297          MOVL    MSCP$L_MEDIA_ID(R2),-   ; Starting with the UNIT ID, followed
00BC C3              07B5  2298                  UCB$L_MEDIA_ID(R3)      ;  by the media identifier and
     F845'      30   07B8  2299          BSBW    DUTU$GET_DEVTYPE        ;  device type.
                     07BB  2300
    1F00 8F     AA   07BB  2301          BICW    #MT$M_DENSITY,-         ; Clear density field in DEVDEPEND.
       44 A3        07BF  2302                  UCB$L_DEVDEPEND(R3)
                     07C1  2303
    50   20 A2    3C   07C1  2304          MOVZWL  MSCP$W_FORMAT(R2),R0    ; Pickup MSCP density code.
      FC5D      30   07C5  2305          BSBW    MSCPTOVMS_DENS          ; Convert to VMS format.
         50     F0   07C8  2306          INSV    R0,-                    ; Insert system density code into
                     07CA  2307                  #MT$V_DENSITY,-         ;  DEVDEPEND.
       05    08       07CA  2308                  #MT$S_DENSITY,-
       44 A3        07CC  2309                  UCB$L_DEVDEPEND(R3)
```

```
                     07CE  2310
                0E A2  B0  07CE  2311          MOVW    MSCP$W_UNT_FLGS(R2),-   ; Copy new unit flags from end packet.
              00E0 C3      07D1  2312                  UCB$W_UNIT_FLAGS(R3)
                22 A2  B0  07D4  2313          MOVW    MSCP$W_SPEED(R2),-      ; Copy speed to UCB.
              00F2 C3      07D7  2314                  UCB$W_TU_SPEED(R3)
                20 A2  B0  07DA  2315          MOVW    MSCP$W_FORMAT(R2),-     ; Copy format to UCB.
              00F0 C3      07DD  2316                  UCB$W_TU_FORMAT(R3)
                   05  E0  07E0  2317          BBS     #MSCP$V_DF_VSMSU,-      ; Branch if suppressing Variable speed
           04 0E A2        07E2  2318                  MSCP$W_UNT_FLGS(R2),10$ ;  mode.
                     07E5  2319 ;         ASSUME  MT$K_SPEED_DEF  EQ      0
                   50  D4  07E5  2320          CLRL    R0                      ; R0 = default speed.
                   0B  11  07E7  2321          BRB     20$                     ; Branch around.
                     07E9  2322 10$:
     50    22 A2  3C  07E9  2323          MOVZWL  MSCP$W_SPEED(R2),R0     ; Get speed of unit.
     51    20 A2  3C  07ED  2324          MOVZWL  MSCP$W_FORMAT(R2),R1    ; And density.
         FC51   30  07F1  2325          BSBW    MSCPTOSPEED             ; Convert Speed to VMS value.
                     07F4  2326 20$:
                   50  F0  07F4  2327          INSV    R0,-                    ; Insert VMS speed value into UCB.
                     07F6  2328                  #MT$V_SPEED,-
              08   18      07F6  2329                  #MT$S_SPEED,-
                44 A3      07F8  2330                  UCB$L_DEVDEPEND(R3)
                     07FA  2331          ASSUME  MSCP$V_UF_WRTPH GE 8
                     07FA  2332          ASSUME  MSCP$V_UF_WRTPS GE 8
                     07FA  2333          ASSUME  MT$V_HWL GE 16
                     07FA  2334          ASSUME  UCB$V_MSCP_WRTP GE 8
     46 A3    08  8A  07FA  2335          BICB    #<MT$M_HWL @ -16>, -    ; Assume device is not hardware write
                     07FE  2336                  UCB$L_DEVDEPEND+2(R3)   ; locked.
                   20  8A  07FE  2337          BICB    #<UCB$M_MSCP_WRTP@-8>,- ; Ditto for class driver write
                69 A3      0800  2338                  UCB$W_DEVSTS+1(R3)      ; protect flag.
                     93  0802  2339          BITB    #<<MSCP$M_UF_WRTPH -    ; Is the unit hardware or
                     0803  2340                    !MSCP$M_UF_WRTPS>@-8>,-; software write protected?
         0F A2       30  0803  2341                  MSCP$W_UNT_FLGS+1(R2)
                   08  13  0806  2342          BEQL    50$                     ; Branch if not write protected.
     46 A3    08  88  0808  2343          BISB    #<MT$M_HWL @ -16>, -    ; Else, set the hardware write
                     080C  2344                  UCB$L_DEVDEPEND+2(R3)   ; locked bit in DEVDEPEND.
                   20  88  080C  2345          BISB    #<UCB$M_MSCP_WRTP@-8>,- ; Set class driver write
                69 A3      080E  2346                  UCB$W_DEVSTS+1(R3)      ; protect flag too.
                     0810  2347
                50 8ED0  0810  2348 50$:     POPL    R0                      ; Restore R0.
                     05  0813  2349          RSB                             ; Return to caller.
```

```
                     0814  2351          .SBTTL  START_UNLOAD and START_AVAILABLE
                     0814  2352
                     0814  2353 ; START_AVAILABLE - Prepare an MSCP packet to do an AVAILABLE command without
                     0814  2354 ;       the spindown modifier.
                     0814  2355 ;
                     0814  2356 ; START_UNLOAD - Prepare an MSCP packet to do an AVAILABLE command with
                     0814  2357 ;       spindown specified.
                     0814  2358 ;
                     0814  2359 ; INPUTS:
                     0814  2360 ;       R2 => MSCP buffer
                     0814  2361 ;       R3 => UCB
                     0814  2362 ;       R4 => PDT
                     0814  2363 ;       R5 => CDRP
                     0814  2364 ;
                     0814  2365 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     0814  2366 ;
                     0814  2367
                     0814  2368 START_REWINDOFF:
                     0814  2369 START_UNLOAD:
                     0814  2370
          10   A8    0814  2371          BISW    #MSCP$M_MD_UNLOD,-      ; Specify the UNLOAD bit in the
      0A A2            0816  2372                  MSCP$W_MODIFIER(R2)     ;  modifier word.
                     0818  2373
                     0818  2374 START_AVAILABLE:
                     0818  2375
          08   90    0818  2376          MOVB    #MSCP$K_OP_AVAIL,-      ; Transfer AVAILABLE opcode
      08 A2            081A  2377                  MSCP$B_OPCODE(R2)       ;  to packet.
                     081C  2378
                     081C  2379          IF_IVCMD then=AVAIL_IVCMD_END   ; Branch if invalid command processing.
                     0820  2380
                     0820  2381          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     0823  2382
                     0823  2383          ASSUME  UCB$V_VALID GE 8
  65 A3    08    8A  0823  2384          BICB    #<UCB$M_VALID @ -8>, -  ; Initialize software volume invalid.
                     0827  2385                  UCB$W_STS+1(R3)
                     0827  2386
                     0827  2387          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                     082A  2388          ACTION_ENTRY    SUCC,  SS$_NORMAL,      AVAILABLE_SUCC
                     082F  2389          ACTION_ENTRY    AVLBL, SS$_NORMAL,      AVAILABLE_SUCC
                     0834  2390          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, AVAILABLE_SEREX
                     0839  2391          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      AVAILABLE_MEDOFL
                     083E  2392          ACTION_ENTRY    ABRTD, SS$_ABORT,       AVAILABLE_ABORT
                     0843  2393          ACTION_ENTRY    DRIVE, SS$_DRVERR,      AVAILABLE_DRVERR
                     0848  2394          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     AVAILABLE_CTRLERR
                     084D  2395          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     AVAIL_IVCMD
                     0852  2396          ACTION_ENTRY    END_TABLE
                     0854  2397
        0822    31   0854  2398          BRW     INVALID_STS             ; Unexpected MSCP end status.
                     0857  2399
                     0857  2400 AVAIL_IVCMD:
                     0857  2401          IVCMD_BEGIN                     ; Begin invalid command processing.
        FDA4    31   085A  2402          BRW     TU_BEGIN_IVCMD          ; Repeat building the MSCP command.
                     085D  2403 AVAIL_IVCMD_END:
                     085D  2404          IVCMD_END                       ; Complete invalid command processing.
                     085F  2405 ; ----- BRB     AVAILABLE_SUCC          ; Fall through to complete operation.
                     085F  2406
                     085F  2407
```

```
                     085F  2408 AVAILABLE_SUCC:                          ; Action routine for MSCP$K_ST_SUCC.
                     085F  2409 AVAILABLE_MEDOFL:                        ; Action routine for MSCP$K_ST_MEDOFL.
                     085F  2410 AVAILABLE_ABORT:                         ; Action routine for MSCP$K_ST_ABORT.
                     085F  2411 AVAILABLE_DRVERR:                        ; Action routine for MSCP$K_ST_DRVERR.
                     085F  2412 AVAILABLE_CTRLERR:                       ; Action routine for MSCP$K_ST_CNTLR.
          04   CA    085F  2413          BICL    #MT$M_ENSEREXCP,-       ; Clear Serious Exception mode on
        44 A3         0861  2414                  UCB$L_DEVDEPEND(R3)     ;  becoming available.
          00   F0    0863  2415          INSV    #MT$K_SPEED_DEF,-       ; Reset Speed to default.
               18     0865  2416                  #MT$V_SPEED,-
               08     0866  2417                  #MT$S_SPEED,-
        44 A3         0867  2418                  UCB$L_DEVDEPEND(R3)
          20   AA    0869  2419          BICW    #MSCP$M_UF_VSMSU,-      ; Also reset bit.
      00E0 C3         086B  2420                  UCB$W_UNIT_FLAGS(R3)
      00B0 C3   D4    086E  2421          CLRL    UCB$L_RECORD(R3)        ; Clear tape position counter.
                      0872  2422          ASSUME  MT$V_BOT  GE 16
                      0872  2423          ASSUME  MT$V_EOF  GE 16
                      0872  2424          ASSUME  MT$V_EOT  GE 16
                      0872  2425          ASSUME  MT$V_HWL  GE 16
                      0872  2426          ASSUME  MT$V_LOST GE 16
  46 A3    1E   8A    0872  2427          BICB    #<<MT$M_EOF ! MT$M_EOT -    ; Clear position sensitive writelock
                      0876  2428                   ! MT$M_HWL ! MT$M_LOST> - ; DEVDEPEND bits.
                      0876  2429                   @ -16>, UCB$L_DEVDEPEND+2(R3)
  46 A3    01   88    0876  2430          BISB    #<MT$M_BOT @ -16>, -    ; Set BOT DEVDEPEND position bit.
                      087A  2431                  UCB$L_DEVDEPEND+2(R3)
                      087A  2432          ASSUME  UCB$V_MSCP_WRTP GE 8
          20   8A    087A  2433          BICB    #<UCB$M_MSCP_WRTP@-8>,- ; Clear class driver write
        69 A3         087C  2434                  UCB$W_DEVSTS+1(R3)      ; protect flag.
                      087E  2435 AVAILABLE_SEREX:
         0447   31    087E  2436          BRW     FUNCTION_EXIT
```

```
                     0881  2438          .SBTTL  Start WRITEOF, WRITEMARK, ERASETAPE, and DSE.
                     0881  2439
                     0881  2440 ; START_WRITEMARK - Prepare an MSCP packet to do a WRITE TAPE MARK command.
                     0881  2441 ; START_ERASETAPE - Prepare an MSCP packet to do an ERASE GAP command.
                     0881  2442 ; START_DSE - Prepare an MSCP packet to do an ERASE command.
                     0881  2443 ;
                     0881  2444 ; INPUTS:
                     0881  2445 ;       R2 => MSCP buffer
                     0881  2446 ;       R3 => UCB
                     0881  2447 ;       R4 => PDT
                     0881  2448 ;       R5 => CDRP
                     0881  2449 ;
                     0881  2450 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     0881  2451 ;
                     0881  2452
                     0881  2453 START_ERASETAPE:
          16   90    0881  2454          MOVB    #MSCP$K_OP_ERGAP,-      ; Transfer ERASEGAP opcode
     08 A2             0883  2455                  MSCP$B_OPCODE(R2)       ;  to packet.
          14   11    0885  2456          BRB     WTM_ERASE_COM           ; Branch around to common.
                     0887  2457
                     0887  2458 START_DSE:
          12   90    0887  2459          MOVB    #MSCP$K_OP_ERASE,-      ; Transfer ERASE opcode
     08 A2             0889  2460                  MSCP$B_OPCODE(R2)       ;  to packet.
          07   E1    088B  2461          BBC     #IO$V_NOWAIT,-          ; If NOT nowait, branch around.
     C0 A5             088D  2462                  CDRP$Q_FUNC(R5),-
          0B           088F  2463                  WTM_ERASE_COM
                     0890  2464          ASSUME  MSCP$V_MD_IMMED LE 7
  0A A2   40 8F   88 0890  2465          BISB    #MSCP$M_MD_IMMED, -     ; If NOWAIT, then set proper TMSCP
                     0895  2466                  MSCP$W_MODIFIER(R2)     ; modifier in command message.
          04   11    0895  2467          BRB     WTM_ERASE_COM           ; Branch around to common.
                     0897  2468
                     0897  2469 START_WRITEMARK:
                     0897  2470 START_WRITEOF:
          24   90    0897  2471          MOVB    #MSCP$K_OP_WRITM,-      ; Transfer WRITE TAPE MARK opcode
     08 A2             0899  2472                  MSCP$B_OPCODE(R2)       ;  to packet.
                     089B  2473
                     089B  2474 WTM_ERASE_COM:
                     089B  2475
                     089B  2476          IF_IVCMD then=WRITM_IVCMD_END   ; Branch if invalid command processing.
                     089F  2477
                     089F  2478          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     08A2  2479
                     08A2  2480          ASSUME  MT$V_BOT  GE 16
                     08A2  2481          ASSUME  MT$V_EOF  GE 16
                     08A2  2482          ASSUME  MT$V_EOT  GE 16
                     08A2  2483          ASSUME  MT$V_LOST GE 16
  46 A3   17   8A    08A2  2484          BICB    #<<MT$M_BOT ! MT$M_EOF -; Clear position sensitive DEVDEPEND
                     08A6  2485                   ! MT$M_EOT -           ; bits
                     08A6  2486                   ! MT$M_LOST> @ -16>, -
                     08A6  2487                  UCB$L_DEVDEPEND+2(R3)
                     08A6  2488
                     08A6  2489          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                     08A9  2490          ACTION_ENTRY    SUCC,  SS$_NORMAL,      WRITM_SUCC
                     08AE  2491          ACTION_ENTRY    ABRTD, SS$_ABORT,       WRITM_ABORT
                     08B3  2492          ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  WRITM_OFFLINE
                     08B8  2493          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      WRITM_AVAIL
                     08BD  2494          ACTION_ENTRY    WRTPR, SS$_WRITLCK,     WRITM_WRITLCK
```

```
                     08C2  2495          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, WRITM_PRESE
                     08C7  2496          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     WRITM_CTRLERR
                     08CC  2497          ACTION_ENTRY    FMTER, SS$_CTRLERR,     WRITM_FMTER
                     08D1  2498          ACTION_ENTRY    DATA,  SS$_PARITY,      WRITM_DATA_ERROR
                     08D6  2499          ACTION_ENTRY    DRIVE, SS$_DRVERR,      WRITM_DRVERR
                     08DB  2500          ACTION_ENTRY    PLOST, SS$_CTRLERR,     ERASEGAP_PLOST
                     08E0  2501          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     WRITM_IVCMD
                     08E5  2502          ACTION_ENTRY    END_TABLE
                     08E7  2503
           078F   31 08E7  2504          BRW     INVALID_STS                     ; Unexpected MSCP end status.
                     08EA  2505
                     08EA  2506 WRITM_IVCMD:
                     08EA  2507          IVCMD_BEGIN                             ; Begin invalid command processing.
           FD11   31 08ED  2508          BRW     TU_BEGIN_IVCMD                  ; Rebuild fatal MSCP command.
                     08F0  2509 WRITM_IVCMD_END:
                     08F0  2510          IVCMD_END                               ; Complete invalid command processing.
             14    11 08F2  2511          BRB     WRITM_END                       ; Branch around to end.
                     08F4  2512
                     08F4  2513 ERASEGAP_PLOST:
                     08F4  2514          ASSUME  MT$V_LOST GE 16
        46 A3    10 88 08F4  2515          BISB    #<MT$M_LOST @ -16>, -           ; Set position LOST DEVDEPEND bit.
                     08F8  2516                  UCB$L_DEVDEPEND+2(R3)
                     08F8  2517 WRITM_ABORT:
                     08F8  2518 WRITM_OFFLINE:
                     08F8  2519 WRITM_AVAIL:
                     08F8  2520 WRITM_WRITLCK:
                     08F8  2521 WRITM_CTRLERR:
                     08F8  2522 WRITM_FMTER:
                     08F8  2523 WRITM_DRVERR:
                     08F8  2524 WRITM_DATA_ERROR:
                     08F8  2525 WRITM_SUCC:
         00B0 C3    D5 08F8  2526          TSTL    UCB$L_RECORD(R3)                ; Previously at BOT?
              04    12 08FC  2527          BNEQ    10$                             ; Branch if not previously at BOT.
        40 A5    20 88 08FE  2528          BISB    #CDRP$M_DENSCK, -               ; Else, set density check required flag.
                     0902  2529                  CDRP$L_DUTUFLAGS(R5)
00B0 C3    1C A2    D0 0902  2530 10$:    MOVL    MSCP$L_POSITION(R2), -          ; Update tape position information.
                     0908  2531                  UCB$L_RECORD(R3)
                     0908  2532 WRITM_END:
              03    E1 0908  2533          BBC     #MSCP$V_EF_EOT,-                ; See if we passed into End Of Tape
           0C 09 A2       090A  2534                  MSCP$B_FLAGS(R2),40$        ;  region, and branch around if NOT.
                     090D  2535          ASSUME  MT$V_EOT  GE 16
        46 A3    04 88 090D  2536          BISB    #<MT$M_EOT @ -16>, -            ; Set EOT DEVDEPEND position bit.
                     0911  2537                  UCB$L_DEVDEPEND+2(R3)
              05 50  E9 0911  2538          BLBC    R0,40$                          ; If already an error, branch around.
        50    0878 8F  B0 0914  2539          MOVW    #SS$_ENDOFTAPE,R0               ; Return EOT.
                     0919  2540 40$:
                     0919  2541 WRITM_PRESE:
                03AC   31 0919  2542          BRW     FUNCTION_EXIT                   ; Branch to common exit.
```

```
                     091C  2544          .SBTTL  Start REWIND.
                     091C  2545
                     091C  2546 ; START_REWIND - Prepare an MSCP packet to do a REWIND command.
                     091C  2547 ;
                     091C  2548 ; A Rewind QIO request causes us to send an MSCP Reposition Command with
                     091C  2549 ; the MSCP$M_MD_REWIND modifier set and both the MSCP$L_REC_CNT and
                     091C  2550 ; MSCP$L_TMGP_CNT fields zero.  If the user specifies IO$M_NOWAIT, then
                     091C  2551 ; the MSCPP$M_MD_IMMED modifier is set in the command that is sent.
                     091C  2552 ;
                     091C  2553 ; INPUTS:
                     091C  2554 ;       R2 => MSCP buffer
                     091C  2555 ;       R3 => UCB
                     091C  2556 ;       R4 => PDT
                     091C  2557 ;       R5 => CDRP
                     091C  2558 ;
                     091C  2559 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     091C  2560 ;
                     091C  2561
                     091C  2562 START_RECAL:
                     091C  2563 START_REWIND:
                     091C  2564
          25   90    091C  2565          MOVB    #MSCP$K_OP_REPOS,-      ; Transfer REPOS. ION opcode
        08 A2         091E  2566                  MSCP$B_OPCODE(R2)       ;  to packet.
          02   A8    0920  2567          BISW    #MSCP$M_MD_REWND,-      ; Specify rewind.
        0A A2         0922  2568                  MSCP$W_MODIFIER(R2)
                      0924  2569
          07   E1    0924  2570          BBC     #IO$V_NOWAIT,-          ; If NOT nowait, branch around.
     05 C0 A5         0926  2571                  CDRP$W_FUNC(R5),10$
                      0929  2572          ASSUME  MSCP$V_MD_IMMED LE 7
 0A A2    40 8F    88 0929  2573          BISB    #MSCP$M_MD_IMMED,-      ; If NOWAIT, then set proper TMSCP
                      092E  2574                  MSCP$W_MODIFIER(R2)     ; modifier in command message.
                      092E  2575
                      092E  2576 10$:     IF_IVCMD then=REWIND_IVCMD_END  ; Branch if invalid command processing.
                      0932  2577
                      0932  2578          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                      0935  2579
                      0935  2580          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                      0938  2581          ACTION_ENTRY    SUCC,  SS$_NORMAL,      REWIND_SUCC
                      093D  2582          ACTION_ENTRY    ABRTD, SS$_ABORT,       REWIND_ABORT
                      0942  2583          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, REWIND_PRESE
                      0947  2584          ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  REWIND_OFFLINE
                      094C  2585          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      REWIND_AVAIL
                      0951  2586          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     REWIND_CTRLERR
                      0956  2587          ACTION_ENTRY    FMTER, SS$_CTRLERR,     REWIND_FMTER
                      095B  2588          ACTION_ENTRY    DRIVE, SS$_DRVERR,      REWIND_DRVERR
                      0960  2589          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     REWIND_IVCMD
                      0965  2590          ACTION_ENTRY    END_TABLE
                      0967  2591
        070F    31    0967  2592          BRW     INVALID_STS             ; Unexpected MSCP end status.
                      096A  2593
                      096A  2594 REWIND_IVCMD:
                      096A  2595          IVCMD_BEGIN                     ; Begin invalid command processing.
        FC91    31    096D  2596          BRW     TU_BEGIN_IVCMD          ; Rebuild fatal MSCP command.
                      0970  2597 REWIND_IVCMD_END:
                      0970  2598          IVCMD_END                       ; Complete invalid command processing.
          10    11    0972  2599          BRB     REWIND_END              ; Branch around to end.
                      0974  2600
```

```
                     0974  2601 REWIND_SUCC:
          1C A2  D0  0974  2602          MOVL    MSCP$L_POSITION(R2),-   ; Update positon on tape.
      00B0 C3        0977  2603                  UCB$L_RECORD(R3)
          08  12    097A  2604          BNEQ    30$                     ; This should be a NOP.
                     097C  2605          ASSUME  MT$V_BOT  GE 16
                     097C  2606          ASSUME  MT$V_EOF  GE 16
                     097C  2607          ASSUME  MT$V_EOT  GE 16
                     097C  2608          ASSUME  MT$V_LOST GE 16
    46 A3  16  8A    097C  2609          BICB    #<<MT$M_EOF ! MT$M_EOT -; Clear position sensitive DEVDEPEND
                     0980  2610                   ! MT$M_LOST> @ -16>, - ; bits.
                     0980  2611                  UCB$L_DEVDEPEND+2(R3)
    46 A3  01  88    0980  2612          BISB    #<MT$M_BOT @ -16>, -    ; Set BOT DEVDEPEND position bit.
                     0984  2613                  UCB$L_DEVDEPEND+2(R3)
                     0984  2614 30$:
                     0984  2615 REWIND_ABORT:
                     0984  2616 REWIND_OFFLINE:
                     0984  2617 REWIND_AVAIL:
                     0984  2618 REWIND_FMTER:
                     0984  2619 REWIND_CTRLERR:
                     0984  2620 REWIND_DRVERR:
                     0984  2621 REWIND_PRESE:
                     0984  2622 REWIND_END:
        0341  31     0984  2623          BRW     FUNCTION_EXIT           ; Branch to common exit.
```

```
                     0987  2625          .SBTTL  Start Space Records and Space Files.
                     0987  2626
                     0987  2627 ;+
                     0987  2628 ; START_SPACEFILE   -
                     0987  2629 ; START_SKIPFILE    - Prepare an MSCP packet to do a REPOSITION command
                     0987  2630 ;                     so as to Skip files.
                     0987  2631 ; START_SPACERECORD -
                     0987  2632 ; START_SKIPRECORD  - Prepare an MSCP packet to do a REPOSITION command
                     0987  2633 ;                     so as to Skip records.
                     0987  2634 ;
                     0987  2635 ; INPUTS:
                     0987  2636 ;       R2 => MSCP buffer
                     0987  2637 ;       R3 => UCB
                     0987  2638 ;       R4 => PDT
                     0987  2639 ;       R5 => CDRP
                     0987  2640 ;       CDRP$L_MEDIA = # of records or files to
                     0987  2641 ;                      skip (word count in longword field).
                     0987  2642 ;
                     0987  2643 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     0987  2644 ;
                     0987  2645
                     0987  2646 START_SKIPFILE:
                     0987  2647 START_SPACEFILE:
                     0987  2648
51   10 A2   9E      0987  2649          MOVAB   MSCP$L_TMGP_CNT(R2),R1  ; R1 => field to fill in for skip files.
     04      11      098B  2650          BRB     SKIP_COMMON             ; Branch around to common code.
                     098D  2651
                     098D  2652 START_SKIPRECORD:
                     098D  2653 START_SPACERECORD:
                     098D  2654
51   0C A2   9E      098D  2655          MOVAB   MSCP$L_REC_CNT(R2),R1   ; R1 => field to fill in for skip records.
                     0991  2656
                     0991  2657 SKIP_COMMON:
        25   90      0991  2658          MOVB    #MSCP$K_OP_REPOS,-      ; Transfer REPOSITION opcode
     08 A2           0993  2659                  MSCP$B_OPCODE(R2)       ;  to packet.
50   D8 A5   32      0995  2660          CVTWL   CDRP$L_MEDIA(R5),R0     ; Pickup # records to skip.
        09   18      0999  2661          BGEQ    10$                     ; GEQ implies positive (forward) movement.
     50 50   CE      099B  2662          MNEGL   R0,R0                   ; Get absolute value of # to skip.
        08   A8      099E  2663          BISW    #MSCP$M_MD_REVRS,-      ; Set modifier to indicate reverse
     0A A2           09A0  2664                  MSCP$W_MODIFIER(R2)     ;  motion.
        14   11      09A2  2665          BRB     17$                     ; If reverse, then do NOT try to detect
                     09A4  2666                                          ;  LEOT, so branch around.
                     09A4  2667
                     09A4  2668 10$:     ; Detect LEOT is performed on all tapes NOT mounted ANSI.  That is,
                     09A4  2669          ; all tapes either NOT mounted or mounted foreign.  The only exception
                     09A4  2670          ; is for physical I/O requests.
                     09A4  2671
0F CA A5    08       E0      09A4  2672          BBS     #IRP$V_PHYSIO, -        ; If physical I/O function, branch
                     09A9  2673                  CDRP$W_STS(R5), 17$     ;  around setting to Detect LEOT.
05 38 A3    13       E1      09A9  2674          BBC     #DEV$V_MNT, -           ; If Tape NOT mounted, go try to Detect
                     09AE  2675                  UCB$L_DEVCHAR(R3), 14$  ; LEOT.
05 38 A3    18       E1      09AE  2676          BBC     #DEV$V_FOR, -           ; If NOT foreign, than ANSI, so branch
                     09B3  2677                  UCB$L_DEVCHAR(R3), 17$  ; around setting to Detect LEOT.
                     09B3  2678 14$:     ASSUME  MSCP$V_MD_DLEOT LE 7
0A A2    80 8F      88      09B3  2679          BISB    #MSCP$M_MD_DLEOT, -     ; Set modifier to ask to Detect LEOT.
                     09B8  2680                  MSCP$W_MODIFIER(R2)
                     09B8  2681
```

```
                     61    50    D0  09B8  2682 17$:    MOVL    R0, (R1)                ; Put #records(files) to skip in packet.
                                 09BB  2683
                                 09BB  2684           IF_IVCMD then=SKIP_IVCMD_END    ; Branch if invalid command processing.
                                 09BF  2685
                                 09BF  2686           SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                                 09C2  2687
                                 09C2  2688           ASSUME  MT$V_BOT  GE 16
                                 09C2  2689           ASSUME  MT$V_EOF  GE 16
                                 09C2  2690           ASSUME  MT$V_EOT  GE 16
                                 09C2  2691           ASSUME  MT$V_LOST GE 16
                  46 A3    17    8A  09C2  2692           BICB    #<<MT$M_BOT ! MT$M_EOF -; Clear position sensitive DEVDEPEND
                                 09C6  2693                     ! MT$M_EOT -            ; bits
                                 09C6  2694                     ! MT$M_LOST> @ -16>, -
                                 09C6  2695                   UCB$L_DEVDEPEND+2(R3)
                                 09C6  2696
                                 09C6  2697           DO_ACTION       TRANSFER        ; Decode MSCP end status.
                                 09C9  2698           ACTION_ENTRY    SUCC,  SS$_NORMAL,      SKIP_SUCC
                                 09CE  2699           ACTION_ENTRY    LED,   SS$_ENDOFVOLUME, SKIP_LEOT
                                 09D3  2700           ACTION_ENTRY    ABRTD, SS$_ABORT,       SKIP_ABORT
                                 09D8  2701           ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, SKIP_PRESE
                                 09DD  2702           ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  SKIP_OFFLINE
                                 09E2  2703           ACTION_ENTRY    AVLBL, SS$_MEDOFL,      SKIP_AVAIL
                                 09E7  2704           ACTION_ENTRY    CNTLR, SS$_CTRLERR,     SKIP_CTRLERR
                                 09EC  2705           ACTION_ENTRY    FMTER, SS$_CTRLERR,     SKIP_FMTER
                                 09F1  2706           ACTION_ENTRY    DRIVE, SS$_DRVERR,      SKIP_DRVERR
                                 09F6  2707           ACTION_ENTRY    BOT,   SS$_NORMAL,      SKIP_BOT
                                 09FB  2708           ACTION_ENTRY    TAPEM, SS$_ENDOFFILE,   SKIP_EOF
                                 0A00  2709           ACTION_ENTRY    PLOST, SS$_CTRLERR,     SKIP_PLOST
                                 0A05  2710           ACTION_ENTRY    ICMD,  SS$_CTRLERR,     SKIP_IVCMD
                                 0A0A  2711           ACTION_ENTRY    END_TABLE
                                 0A0C  2712
                        066A     31    0A0C  2713           BRW     INVALID_STS             ; Unexpected MSCP end status.
                                 0A0F  2714
                                 0A0F  2715 SKIP_IVCMD:
                                 0A0F  2716           IVCMD_BEGIN                     ; Begin invalid command processing.
                           FBEC  31    0A12  2717           BRW     TU_BEGIN_IVCMD          ; Rebuild fatal MSCP command.
                                 0A15  2718 SKIP_IVCMD_END:
                                 0A15  2719           IVCMD_END                       ; Complete invalid command processing.
                                 0A17  2720 ; ----- BRB     SKIP_ABORT              ; Fall through to finish skip operation.
                                 0A17  2721 SKIP_PRESE:
                                 0A17  2722 SKIP_ABORT:
                                 0A17  2723 SKIP_OFFLINE:
                                 0A17  2724 SKIP_AVAIL:
                  50    50    10    9C  0A17  2725           ROTL    #16,R0,R0               ; Move SS$_ code into low order.
                              34    11  0A1B  2726           BRB     SKIP_END                ; Branch around to end.
                                 0A1D  2727
                                 0A1D  2728 SKIP_PLOST:
                                 0A1D  2729           ASSUME  MT$V_LOST GE 16
                     46 A3    10    88  0A1D  2730           BISB    #<MT$M_LOST @ -16>, -   ; Set position LOST DEVDEPEND bit.
                                 0A21  2731                   UCB$L_DEVDEPEND+2(R3)
                              0A    11  0A21  2732           BRB     SKIP_SUCC               ; Rejoin common code.
                                 0A23  2733 SKIP_EOF:
                                 0A23  2734           ASSUME  MT$V_EOF  GE 16
                     46 A3    02    88  0A23  2735           BISB    #<MT$M_EOF @ -16>, -    ; Set EOF DEVDEPEND position bit.
                                 0A27  2736                   UCB$L_DEVDEPEND+2(R3)
                              04    11  0A27  2737           BRB     SKIP_SUCC               ; Rejoin common code.
                                 0A29  2738 SKIP_BOT:
```

```
                        0A29  2739          ASSUME  MT$V_BOT  GE 16
        46 A3   01   88  0A29  2740          BISB    #<MT$M_BOT @ -16>, -    ; Set BOT DEVDEPEND position bit.
                        0A2D  2741                  UCB$L_DEVDEPEND+2(R3)
                        0A2D  2742 ; ----- BRB     SKIP_SUCC               ; Rejoin common code.
                        0A2D  2743 SKIP_FMTER:
                        0A2D  2744 SKIP_CTRLERR:
                        0A2D  2745 SKIP_DRVERR:
                        0A2D  2746 SKIP_SUCC:
                        0A2D  2747 SKIP_LEOT:
     04 09 A2   03   E1 0A2D  2748          BBC     #MSCP$V_EF_EOT, -       ; Is tape in the EOT region?
                        0A32  2749                  MSCP$B_FLAGS(R2), 10$   ; Branch if tape not in EOT.
                        0A32  2750          ASSUME  MT$V_EOT  GE 16
        46 A3   04   88 0A32  2751          BISB    #<MT$M_EOT @ -16>, -    ; Else, set EOT DEVDEPEND position bit.
                        0A36  2752                  UCB$L_DEVDEPEND+2(R3)
                        0A36  2753
        00B0 C3   D5    0A36  2754 10$:     TSTL    UCB$L_RECORD(R3)        ; Previously at BOT?
             04   12    0A3A  2755          BNEQ    15$                     ; Branch if not previously at BOT.
        40 A5   20   88 0A3C  2756          BISB    #CDRP$M_DENSCK, -       ; Else, set density check required flag.
                        0A40  2757                  CDRP$L_DUTUFLAGS(R5)
  00B0 C3   1C A2   D0  0A40  2758 15$:     MOVL    MSCP$L_POSITION(R2), -  ; Update tape position information.
                        0A46  2759                  UCB$L_RECORD(R3)
           0C A2   C1  0A46  2760          ADDL3   MSCP$L_RCSKIPED(R2),-   ; Add records and tapemarks skipped
     51    10 A2       0A49  2761                  MSCP$L_TMSKIPED(R2),R1   ;  so as to return to user.
50   50    F0 8F   79  0A4C  2762          ASHQ    #-16,R0,R0              ; Shift count and SS$_ code into position.
                        0A51  2763 SKIP_END:
            0274   31  0A51  2764          BRW     FUNCTION_EXIT           ; Branch to common exit.
```

```
                                        .SBTTL  Start a SETCHAR or a SETMODE function
                                ;
                                ; START_SETCHAR and START_SETMODE
                                ;       The quad-word of data for the operation is contained in IRP$L_MEDIA.
                                ;       This "PHYSICAL" I/O function and the "LOGICAL" I/O function
                                ;       SET MODE are almost identical.  The only difference is that while
                                ;       both allow for the setting of:
                                ;
                                ;               1. Default buffer size
                                ;               2. Tape density (1600 BPI or 6250 BPI).
                                ;               3. Tape format
                                ;               4. Serious Exception mode
                                ;
                                ;       the former function (i.e. SET CHARACTERISTICS) also allows for
                                ;       the resetting of the DEVICE CLASS and the DEVICE TYPE fields in
                                ;       the UCB.
                                ;
                                ;       The first two bytes of the QUADWORD of data at IRP$L_MEDIA contain
                                ;       the DEVICE CLASS and DEVICE TYPE respectively for a SETCHAR.
                                ;       The next word of the QUADWORD contains the new buffer size.  The
                                ;       third word contains new density and format information.  The fourth
                                ;       word of the QUADWORD is reserved.
                                ;
                                ; INPUTS:
                                ;       R2 => MSCP buffer
                                ;       R3 => UCB
                                ;       R4 => PDT
                                ;       R5 => CDRP
                                ;
                                START_SETCHAR:
                                        ASSUME  UCB$B_DEVTYPE EQ UCB$B_DEVCLASS+1
40 A3   D8 A5   B0              MOVW    CDRP$L_MEDIA(R5),UCB$B_DEVCLASS(R3)     ; Reset CLASS and TYPE.
                                START_SETMODE:
42 A3   DA A5   B0              MOVW    CDRP$L_MEDIA+2(R5),UCB$W_DEVBUFSIZ(R3)  ; Copy new buffer size.
                                        START_SEQNOP                            ; Synchronize class driver - server
                                                                                ; communications so that only this
                                                                                ; thread is sending commands to the
                                                                                ; server.
                                        ASSUME  CDRP$V_CAND EQ 0
     22 40 A5   E8              BLBS    CDRP$L_DUTUFLAGS(R5), -                 ; Was I/O request canceled?
                                        SETMODE_CANCEL                          ; Branch if request was canceled.
        03      90              MOVB    #MSCP$K_OP_GTUNT,-                      ; Opcode is for GET UNIT STATUS.
           08 A2                        MSCP$B_OPCODE(R2)
                                        ASSUME  MSCP$V_MD_CLSEX GE 8
        20      8A              BICB    #<MSCP$M_MD_CLSEX@-8>,-                 ; The clear serious execption modifier
           0@ A2                        MSCP$W_MODIFIER+1(R2)                   ; is illegal on get unit status cmds.
                                        SEND_MSCP_MSG                           ; Send message to remote MSCP server.
                                ;
                                        IF_MSCP SUCCESS, then=SETMODE_ONLINE    ; Branch if GTUNT successful.
                                        .IF     DF      TU_SEQCHK               ; Override sequence checking and
                                        BSBW    OVERRIDE_SEQCHK                 ; remove sequence number from array.
                                        .ENDC
 50   01A4 8F   3C              MOVZWL  #SS$_MEDOFL, R0                         ; Setup final I/O status.
```

```
                     0A8E  2823 SETMODE_ABORT:
                     0A8E  2824 SETMODE_OFFLINE:
                     0A8E  2825 SETMODE_CTRLERR:
                     0A8E  2826 SETMODE_DRVERR:
        08    EF     0A8E  2827          EXTZV   #MT$V_DENSITY,-
        05           0A90  2828                  #MT$S_DENSITY,-
51    DC A5          0A91  2829                  CDRP$L_MEDIA+4(R5),R1   ; Extract user designated DENSITY parameter.
        51    F0     0A94  2830          INSV    R1,-                    ; And insure that UCB$L_DEVDEPEND winds
                     0A96  2831                  #MT$V_DENSITY,-         ;  up with the correct value for DENSITY
   05    08          0A96  2832                  #MT$S_DENSITY,-
     44 A3           0A98  2833                  UCB$L_DEVDEPEND(R3)
                     0A9A  2834
                     0A9A  2835 SETMODE_CANCEL:
      00B0   31      0A9A  2836          BRW     SETMODE_RETURN          ; And branch around.
                     0A9D  2837
                     0A9D  2838 SETMODE_ONLINE:
                     0A9D  2839
                     0A9D  2840          ASSUME  CDRP$V_CAND EQ 0
  ED 40 A5    E8     0A9D  2841          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                     0AA1  2842                  SETMODE_ABORT           ; Branch if request was canceled.
        02    E0     0AA1  2843          BBS     #MT$V_ENSEREXCP,-       ; Branch if Serious Exception explicitly
06 DC A5             0AA3  2844                  CDRP$L_MEDIA+4(R5),10$  ;  enabled.
        04    CA     0AA6  2845          BICL    #MT$M_ENSEREXCP,-       ; Else clear Serious Exception mode.
     44 A3           0AA8  2846                  UCB$L_DEVDEPEND(R3)
        04    11     0AAA  2847          BRB     20$                     ; And branch around.
                     0AAC  2848 10$:
        04    C8     0AAC  2849          BISL    #MT$M_ENSEREXCP,-       ; Enable Serious Exception mode.
     44 A3           0AAE  2850                  UCB$L_DEVDEPEND(R3)
                     0AB0  2851 20$:
     20 A2    B0     0AB0  2852          MOVW    MSCP$W_FORMAT(R2),-     ; Copy format to UCB before recycling
   00F0 C3          0AB3  2853                  UCB$W_TU_FORMAT(R3)     ;  end message.
                     0AB6  2854
                     0AB6  2855          RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
                     0AB9  2856
                     0AB9  2857 SETMODE_BEGIN_IVCMD:
                     0AB9  2858
        0A    90     0AB9  2859          MOVB    #MSCP$K_OP_STUNT,-      ; Transfer Set Unit Characteristics
     08 A2           0ABB  2860                  MSCP$B_OPCODE(R2)       ;  opcode to packet.
                     0ABD  2861
   00E0 C3    B0     0ABD  2862          MOVW    UCB$W_UNIT_FLAGS(R3),-  ; Copy unit flags to MSCP packet.
     0E A2           0AC1  2863                  MSCP$W_UNT_FLGS(R2)
                     0AC3  2864
   00D8 C3    D0     0AC3  2865          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Copy Device dependent parameters to
     1C A2           0AC7  2866                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                     0AC9  2867
   00B0 C3    D5     0AC9  2868          TSTL    UCB$L_RECORD(R3)        ; Is tape at BOT?
        19    12     0ACD  2869          BNEQ    35$                     ; Skip density setup if not at BOT.
        08    EF     0ACF  2870          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
        05           0AD1  2871                  #MT$S_DENSITY,-         ;  specified for this unit
50    DC A5          0AD2  2872                  CDRP$L_MEDIA+4(R5),R0   ;  and put into R0.
                     0AD5  2873
      F934    30     0AD5  2874          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
     09 50    E8     0AD8  2875          BLBS    R0,30$                  ; LBS means successful conversion.
        08    EF     0ADB  2876          EXTZV   #MT$V_DENSITY,-         ; Determine density that the user has
        05           0ADD  2877                  #MT$S_DENSITY,-         ;  last established for this unit
50    44 A3          0ADE  2878                  UCB$L_DEVDEPEND(R3),R0  ;  and put into R0.
      F928    30     0AE1  2879          BSBW    VMSTOMSCP_DENS          ; Convert VMS density to MSCP format.
```

```
                     0AE4  2880 30$:
20 A2   51   B0      0AE4  2881          MOVW    R1,MSCP$W_FORMAT(R2)    ; Copy MSCP density to packet.
                     0AE8  2882
                     0AE8  2883 35$:     ASSUME  MT$K_SPEED_DEF EQ 0
        18   EF      0AE8  2884          EXTZV   #MT$V_SPEED,-           ; Extract user specified speed.
        08           0AEA  2885                  #MT$S_SPEED,-
50   DC A5           0AEB  2886                  CDRP$L_MEDIA+4(R5),R0
        09   13      0AEE  2887          BEQL    40$                     ; EQL implies default.
      F93D   30      0AF0  2888          BSBW    SPEEDTOMSCP             ; Convert speed to MSCP format.
        20   A8      0AF3  2889          BISW    #MSCP$M_UF_VSMSU,-      ; Enable variable speed mode suppression.
     0E A2           0AF5  2890                  MSCP$W_ONT_FLGS(R2)
        04   11      0AF7  2891          BRB     50$                     ; And branch around.
                     0AF9  2892 40$:
        20   AA      0AF9  2893          BICW    #MSCP$M_UF_VSMSU,-      ; Disable variable speed mode suppression.
     0E A2           0AFB  2894                  MSCP$W_ONT_FLGS(R2)
                     0AFD  2895 50$:
22 A2   50   B0      0AFD  2896          MOVW    R0,MSCP$W_SPEED(R2)     ; Place speed value into packet.
                     0B01  2897
      F966   30      0B01  2898          BSBW    SET_CLEAR_SEX           ; Set SEX if called for.
                     0B04  2899
                     0B04  2900          IF_IVCMD then=SETMODE_IVCMD_END ; Branch if invalid command processing.
                     0B08  2901
                     0B08  2902          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     0B0B  2903
                     0B0B  2904          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                     0B0E  2905          ACTION_ENTRY    SUCC,  SS$_NORMAL,      SETMODE_SUCC
                     0B13  2906          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, SETMODE_RETURN
                     0B18  2907          ACTION_ENTRY    ABRTD, SS$_ABORT,       SETMODE_ABORT
                     0B1D  2908          ACTION_ENTRY    ICMD,  SS$_BUGCHECK,    SETMODE_IVCMD
                     0B22  2909          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      SETMODE_OFFLINE
                     0B27  2910          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      SETMODE_OFFLINE
                     0B2C  2911          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     SETMODE_CTRLERR
                     0B31  2912          ACTION_ENTRY    FMTER, SS$_CTRLERR,     SETMODE_CTRLERR
                     0B36  2913          ACTION_ENTRY    DRIVE, SS$_DRVERR,      SETMODE_DRVERR
                     0B3B  2914          ACTION_ENTRY    END_TABLE
                     0B3D  2915
0539   31            0B3D  2916          BRW     INVALID_STS             ; Unexpected MSCP end status.
                     0B40  2917
                     0B40  2918 SETMODE_IVCMD:
                     0B40  2919          IVCMD_BEGIN                     ; Begin invalid command processing.
FF73   31            0B43  2920          BRW     SETMODE_BEGIN_IVCMD     ; Rebuild fatal MSCP command.
                     0B46  2921 SETMODE_IVCMD_END:
                     0B46  2922          IVCMD_END                       ; Complete invalid command processing.
  03   11            0B48  2923          BRB     SETMODE_RETURN          ; Complete setmode operation.
                     0B4A  2924
                     0B4A  2925 SETMODE_SUCC:
                     0B4A  2926
FC48   30            0B4A  2927          BSBW    RECORD_SETUNIT_CHAR     ; Record data from End Message in UCB.
                     0B4D  2928
                     0B4D  2929 SETMODE_RETURN:
                     0B4D  2930          END_SEQNOP                      ; End synchronized class driver -
                     0B63  2931                                          ; server communications.
0162   31            0B63  2932          BRW     FUNCTION_EXIT           ; Terminate I/O request.
```

```
                     0B66  2934          .SBTTL  Start SENSECHAR and SENSEMODE functions.
                     0B66  2935
                     0B66  2936 ; START_SENSECHAR and START_SENSEMODE.
                     0B66  2937 ;
                     0B66  2938 ; INPUTS:
                     0B66  2939 ;       R2 => MSCP buffer
                     0B66  2940 ;       R3 => UCB
                     0B66  2941 ;       R4 => PDT
                     0B66  2942 ;       R5 => CDRP
                     0B66  2943 ;
                     0B66  2944
                     0B66  2945 START_SENSECHAR:
                     0B66  2946 START_SENSEMODE:
                     0B66  2947
          03   90    0B66  2948          MOVB    #MSCP$K_OP_GTUNT,-      ; Opcode is for GET UNIT STATUS.
        08 A2         0B68  2949                  MSCP$B_OPCODE(R2)
                     0B6A  2950          ASSUME  MSCP$V_MD_CLSEX GE 8
          20   8A    0B6A  2951          BICB    #<MSCP$M_MD_CLSEX@-8>,- ; The clear serious execption modifier
        0B A2         0B6C  2952                  MSCP$W_MODIFIER+1(R2)   ; is illegal on get unit status cmds.
                     0B6E  2953          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                     0B71  2954
                     0B71  2955          IF_MSCP_SUCCESS, then=SENSEMODE_ONLINE  ; Branch if GTUNT successful.
    50   01A4 8F    3C    0B77  2956          MOVZWL  #SS$_MEDOFL,R0          ; Mark final I/O status.
          06   11    0B7C  2957          BRB     SENSEMODE_RETURN        ; And branch around.
                     0B7E  2958
                     0B7E  2959 SENSEMODE_ONLINE:
                     0B7E  2960
        FC22   30    0B7E  2961          BSBW    RECORD_GETUNIT_CHAR     ; Copy data from End Message to UCB.
    50    01   3C    0B81  2962          MOVZWL  #SS$_NORMAL, R0         ; Setup successful completion status.
                     0B84  2963
                     0B84  2964 SENSEMODE_RETURN:
        0141   31    0B84  2965          BRW     FUNCTION_EXIT
```

```
                     0B87  2967          .SBTTL  START_READPBLK and START_WRITEPBLK and START_WRITECHECK
                     0B87  2968
                     0B87  2969 ; START_READPBLK - Prepare an MSCP packet to do a READ command.
                     0B87  2970 ;
                     0B87  2971 ; START_WRITEPBLK - Prepare an MSCP packet to do a WRITE command.
                     0B87  2972 ;
                     0B87  2973 ; START_WRITECHECK - Prepare an MSCP packet to do a COMPARE HOST DATA command.
                     0B87  2974 ;
                     0B87  2975 ; INPUTS:
                     0B87  2976 ;       R2 => MSCP buffer
                     0B87  2977 ;       R3 => UCB
                     0B87  2978 ;       R4 => PDT
                     0B87  2979 ;       R5 => CDRP
                     0B87  2980 ;
                     0B87  2981 ;       MSCP packet is zero except for MSCP$L_CMD_REF and MSCP$W_UNIT fields.
                     0B87  2982 ;
                     0B87  2983
                     0B87  2984          .enable lsb
                     0B87  2985 START_WRITECHECK:
                     0B87  2986
     20   90         0B87  2987          MOVB    #MSCP$K_OP_COMP,-       ; Compare host data opcode
 08 A2                0B89  2988                  MSCP$B_OPCODE(R2)       ;  to packet.
     06   E1         0B8B  2989          BBC     #IO$V_REVERSE,-         ; Branch around if NOT reverse.
23 C0 A5              0B8D  2990                  CDRP$Q_FUNC(R5),20$
     08   A8         0B90  2991          BISW    #MSCP$M_MD_REVRS,-      ; Else set reverse modifier.
 0A A2                0B92  2992                  MSCP$W_MODIFIER(R2)
     1D   11         0B94  2993          BRB     20$                     ; And branch around to join common code
                     0B96  2994
                     0B96  2995 START_WRITEPBLK:
                     0B96  2996
     22   90         0B96  2997          MOVB    #MSCP$K_OP_WRITE,-      ; Transfer WRITE opcode
 08 A2                0B98  2998                  MSCP$B_OPCODE(R2)       ;  to packet.
     0D   11         0B9A  2999          BRB     10$
                     0B9C  3000
                     0B9C  3001 START_READPBLK:
                     0B9C  3002
     21   90         0B9C  3003          MOVB    #MSCP$K_OP_READ,-       ; Transfer READ opcode
 08 A2                0B9E  3004                  MSCP$B_OPCODE(R2)       ;  to packet.
                     0BA0  3005
     06   E1         0BA0  3006          BBC     #IO$V_REVERSE,-         ; Branch around if NOT reverse.
04 C0 A5              0BA2  3007                  CDRP$Q_FUNC(R5),10$
     08   A8         0BA5  3008          BISW    #MSCP$M_MD_REVRS,-      ; Else set reverse modifier.
 0A A2                0BA7  3009                  MSCP$W_MODIFIER(R2)
                     0BA9  3010 10$:
                     0BA9  3011
     0E   E1         0BA9  3012          BBC     #IO$V_DATACHECK,-       ; See if user specified compare in
05 C0 A5              0BAB  3013                  CDRP$Q_FUNC(R5),20$     ;  addition to data transfer. If not, branch
                     0BAE  3014          ASSUME  MSCP$V_MD_COMP GE 8     ; Else, set the read/write with
0B A2   40 8F        0BAE  3015          BISB    #<MSCP$M_MD_COMP@-8>, - ; data compare modifier.
                     0BB3  3016                  MSCP$W_MODIFIER+1(R2)
                     0BB3  3017 20$:
                     0BB3  3018          IF_IVCMD then=70$               ; Branch if invalid command processing.
                     0BB7  3019
     30 A5   9E      0BB7  3020          MOVAB   CDRP$T_LBUFHNDL(R5),-   ; Put address of Local BUFfer HaNDLe
     2C A5           0BBA  3021                  CDRP$L_LBUFH_AD(R5)     ;  field into field that points to it.
                     0BBC  3022          MAP_IRP                         ; Allocate mapping resources and load
                     0BBF  3023                                          ;  them with data from SVAPTE, BOFF,
```

```
                        0BBF  3024                                      ;  and BCNT derived from IRP within
                        0BBF  3025                                      ;  CDRP.
                        0BBF  3026
52   1C A5    D0        0BBF  3027          MOVL    CDRP$L_MSG_BUF(R5),R2   ; Refresh R2 => MSCP packet.
     30 A5    7D        0BC3  3028 70$:     MOVQ    CDRP$T_LBUFHNDL(R5),-   ; Copy contents of buffer handle to
     10 A2              0BC6  3029                  MSCP$B_BUFFER(R2)       ;  MSCP buffer descriptor field.
     38 A5    D0        0BC8  3030          MOVL    CDRP$T_LBUFHNDL+8(R5),- ; Buffer handle is 96 bits (12 bytes)
     18 A2              0BCB  3031                  MSCP$B_BUFFER+8(R2)     ;  in length.
     D2 A5    D0        0BCD  3032          MOVL    CDRP$L_BCNT(R5),-
     0C A2              0BD0  3033                  MSCP$L_BYTE_CNT(R2)     ; Copy byte count of transfer.
                        0BD2  3034
                        0BD2  3035          IF_IVCMD then=XFER_IVCMD_END    ; Branch if invalid command processing.
                        0BD6  3036
                        0BD6  3037          .enable lsb                     ; Start a new local symbol block.
                        0BD6  3038
                        0BD6  3039          SEND_MSCP_MSG                   ; Send message to remote MSCP server.
                        0BD9  3040
                        0BD9  3041          ASSUME  MT$V_BOT  GE 16
                        0BD9  3042          ASSUME  MT$V_EOF  GE 16
                        0BD9  3043          ASSUME  MT$V_EOT  GE 16
                        0BD9  3044          ASSUME  MT$V_LOST GE 16
46 A3    17    8A       0BD9  3045          BICB    #<<MT$M_BOT ! MT$M_EOF -; Clear position sensitive DEVDEPEND
                        0BDD  3046                    ! MT$M_EOT -          ; bits.
                        0BDD  3047                    ! MT$M_LOST> @ -16>, -
                        0BDD  3048                  UCB$L_DEVDEPEND+2(R3)
                        0BDD  3049
                        0BDD  3050          DO_ACTION       TRANSFER        ; Decode MSCP end status.
                        0BE0  3051          ACTION_ENTRY    SUCC,  SS$_NORMAL,      TRANSFER_RTN_RECLEN
                        0BE5  3052          ACTION_ENTRY    PRESE, SS$_SERIOUSEXCP, TRANSFER_PRESE
                        0BEA  3053          ACTION_ENTRY    ABRTD, SS$_ABORT,       TRANSFER_RTN_BCNT
                        0BEF  3054          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     TRANSFER_INVALID_COMMAND
                        0BF4  3055          ACTION_ENTRY    COMP,  SS$_DATACHECK,   TRANSFER_COMPERR
                        0BF9  3056          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      TRANSFER_MEDOFL
                        0BFE  3057          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      TRANSFER_MEDOFL
                        0C03  3058          ACTION_ENTRY    TAPEM, SS$_ENDOFFILE,   TRANSFER_EOF
                        0C08  3059          ACTION_ENTRY    BOT,   SS$_ENDOFFILE,   TRANSFER_BOT
                        0C0D  3060          ACTION_ENTRY    PLOST, SS$_CTRLERR,     TRANSFER_PLOST
                        0C12  3061          ACTION_ENTRY    RDTRN, SS$_DATAOVERUN,  TRANSFER_RTN_RECLEN
                        0C17  3062          ACTION_ENTRY    DATA,  SS$_PARITY,      TRANSFER_DATA_ERROR
                        0C1C  3063          ACTION_ENTRY    HSTBF, SS$_IVBUFLEN,    TRANSFER_HOST_BUFFER_ERROR
                        0C21  3064          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     TRANSFER_CTRLERR
                        0C26  3065          ACTION_ENTRY    FMTER, SS$_CTRLERR,     TRANSFER_RTN_BCNT
                        0C2B  3066          ACTION_ENTRY    DRIVE, SS$_DRVERR,      TRANSFER_RTN_BCNT
                        0C30  3067          ACTION_ENTRY    WRTPR, SS$_WRITLCK,     TRANSFER_RTN_BCNT
                        0C35  3068          ACTION_ENTRY    END_TABLE
                        0C37  3069
           043F  31     0C37  3070          BRW     INVALID_STS             ; Unexpected MSCP end status.
                        0C3A  3071
                        0C3A  3072 XFER_IVCMD_END:
             3A  11     0C3A  3073          BRB     TRANSFER_IVCMD_END      ; Branch assist.
                        0C3C  3074
                        0C3C  3075
                        0C3C  3076 TRANSFER_PLOST:
                        0C3C  3077          ASSUME  MT$V_LOST GE 16
46 A3    10    88       0C3C  3078          BISB    #<MT$M_LOST @ -16>, -   ; Set position LOST DEVDEPEND bit.
                        0C40  3079                  UCB$L_DEVDEPEND+2(R3)
             0A  11     0C40  3080          BRB     300$                    ; Join common code.
```

```
                        0C42  3081 TRANSFER_EOF:
                        0C42  3082         .ASSUME  MT$V_EOF  GE 16
          46 A3   02    88  0C42  3083         BISB    #<MT$M_EOF @ -16>, -    ; Set EOF DEVDEPEND position bit.
                        0C46  3084                 UCB$L_DEVDEPEND+2(R3)
                  04    11  0C46  3085         BRB     300$                    ; Join common code.
                        0C48  3086 TRANSFER_BOT:
                        0C48  3087         .ASSUME  MT$V_BOT  GE 16
          46 A3   01    88  0C48  3088         BISB    #<MT$M_BOT @ -16>, -    ; Set BOT DEVDEPEND position bit.
                        0C4C  3089                 UCB$L_DEVDEPEND+2(R3)
                        0C4C  3090 ; ----- BRB     300$                    ; Join common code.
                        0C4C  3091
                    51  D4  0C4C  3092 300$:   CLRL    R1                      ; Set zero bytes transfered.
                  0049  31  0C4E  3093         BRW     TRANSFER_SHIFT          ; Branch around.
                        0C51  3094
                        0C51  3095 TRANSFER_PRESE:
                        0C51  3096
                    51  D4  0C51  3097         CLRL    R1                      ; R1 = number of bytes transferred.
     50   50    F0 8F   79  0C53  3098         ASHQ    #-16,R0,R0              ; Shift into proper position for IOSB.
                  006D  31  0C58  3099         BRW     FUNCTION_EXIT           ; Complete function immedately.
                        0C5B  3100
                        0C5B  3101 TRANSFER_CTRLERR:
                    05  EF  0C5B  3102         EXTZV   #MSCP$S_ST_MASK,-       ; Extract the sub-code only.
                    0B      0C5D  3103                 #16-MSCP$S_ST_MASK,-
          51    0A A2       0C5E  3104                 MSCP$W_STATUS(R2),R1
             51   01  B1  0C61  3105         CMPW    #MSCP$K_SC_DLATE,R1     ; Compare to Data Late error.
                  07  12  0C64  3106         BNEQ    25$                     ; Branch around if not Data Late.
     50   22740000 8F  D0  0C66  3107         MOVL    #SS$_DATALATE@16,R0     ; Set SS$_DATALATE into high word.
                  002A  31  0C6D  3108 25$:    BRW     TRANSFER_SHIFT          ; Branch to common code.
                        0C70  3109
                        0C70  3110 TRANSFER_INVALID_COMMAND:
                        0C70  3111
                        0C70  3112         IVCMD_BEGIN                     ; Begin invalid command processing.
                  F98B  31  0C73  3113         BRW     TU_BEGIN_IVCMD          ; Rebuild fatal MSCP command.
                        0C76  3114 TRANSFER_IVCMD_END:
                        0C76  3115         IVCMD_END                       ; Complete invalid command processing.
                    D2  11  0C78  3116         BRB     300$                    ; Complete the function.
                        0C7A  3117
                        0C7A  3118 TRANSFER_MEDOFL:
                        0C7A  3119
                    06  E1  0C7A  3120         BBC     #MSCP$V_SC_INOPR,-      ; Branch around if NOT unit inoperative
                0A A2       0C7C  3121                 MSCP$W_STATUS(R2),-     ;  substatus.
                  17        0C7E  3122                 TRANSFER_RTN_BCNT
     50   008C0000 8F  D0  0C7F  3123         MOVL    #SS$_DRVERR@16,R0       ; Else set up R0 with proper SS$_ code
                        0C86  3124                                             ;  in high order word and
                    0E  11  0C86  3125         BRB     TRANSFER_RTN_BCNT       ; Branch around.
                        0C88  3126 TRANSFER_HOST_BUFFER_ERROR:
                        0C88  3127
                    05  EF  0C88  3128         EXTZV   #MSCP$S_ST_MASK,-       ; Extract the sub-code only.
                    0B      0C8A  3129                 #16-MSCP$S_ST_MASK,-
          51    0A A2       0C8B  3130                 MSCP$W_STATUS(R2),R1
             51   02  B1  0C8E  3131         CMPW    #MSCP$K_SC_ODDBC,R1     ; Compare to Odd Byte Count error.
                  03  13  0C91  3132         BEQL    TRANSFER_RTN_BCNT       ; Branch around if Odd BCNT.
                03E3  31  0C93  3133         BRW     INVALID_STS             ; Here we got an invalid MSCP status.
                        0C96  3134
                        0C96  3135 TRANSFER_DATA_ERROR:                        ; TRANSFER action routine for MSCP$K_ST_DATA
                        0C96  3136
                        0C96  3137 TRANSFER_COMPERR:
```

```
                          0C96  3138 TRANSFER_RTN_BCNT:
                          0C96  3139 TRANSFER_RTN_RECLEN:                 ; Common TRANSFER action routine.
                          0C96  3140                                      ; Here R0 contains SS$_ code in hi order..
      51    0C A2    D0   0C96  3141          MOVL    MSCP$L_BYTE_CNT(R2),R1 ; Get # bytes actually transferred.
                          0C9A  3142
                          0C9A  3143 TRANSFER_SHIFT:
                          0C9A  3144
50    50    F0 8F    79   0C9A  3145          ASHQ    #-16,R0,R0             ; Shift into proper position for IOSB.
                          0C9F  3146
                          0C9F  3147 NORMAL_TRANSFEREND:
                          0C9F  3148
   04 09 A2    03    E1   0C9F  3149          BBC     #MSCP$V_EF_EOT, -      ; Is tape in the EOT region?
                          0CA4  3150                  MSCP$B_FLAGS(R2), 65$  ; Branch if tape not in EOT.
                          0CA4  3151          ASSUME  MT$V_EOT  GE 16
      46 A3    04    88   0CA4  3152          BISB    #<MT$M_EOT @ -16>, -   ; Else, set EOT DEVDEPEND position bit.
                          0CA8  3153                  UCB$L_DEVDEPEND+2(R3)
         0D 50    E9   0CA8  3154 65$:      BLBC    R0, 70$                ; Branch if already returning an error.
0A A2    0400 8F    B1   0CAB  3155          CMPW    #<MSCP$M_SC_EOT -      ; Was a EOT subcode returned on a
                          0CB1  3156                   +MSCP$K_ST_SUCC>, -   ; success command status?
                          0CB1  3157                  MSCP$W_STATUS(R2)
            05    12   0CB1  3158          BNEQ    70$                    ; Branch if not EOT.
   50    0878 8F    B0   0CB3  3159          MOVW    #SS$_ENDOFTAPE, R0     ; Else, return EOT status.
                          0CB8  3160
      00B0 C3    D5   0CB8  3161 70$:      TSTL    UCB$L_RECORD(R3)       ; Previously at BOT?
            04    12   0CBC  3162          BNEQ    75$                    ; Branch if not previously at BOT.
      40 A5    20    88   0CBE  3163          BISB    #CDRP$M_DENSCK, -      ; Else, set density check required flag.
                          0CC2  3164                  CDRP$L_DUTUFLAGS(R5)
00B0 C3    1C A2    D0   0CC2  3165 75$:      MOVL    MSCP$L_POSITION(R2), - ; Update tape position information.
                          0CC8  3166                  UCB$L_RECORD(R3)
                          0CC8  3167
                          0CC8  3168 ; ----- BRB     FUNCTION_EXIT          ; Go to common exit code.
                          0CC8  3169
                          0CC8  3170          .disable        lsb
```

```
                        0CC8  3172          .SBTTL  FUNCTION_EXIT
                        0CC8  3173
                        0CC8  3174 ; FUNCTION_EXIT -
                        0CC8  3175 ;
                        0CC8  3176 ; INPUTS:
                        0CC8  3177 ;       R0 => Final I/O status
                        0CC8  3178 ;       R3 => UCB
                        0CC8  3179 ;       R4 => PDT
                        0CC8  3180 ;       R5 => CDRP
                        0CC8  3181 ;
                        0CC8  3182
                        0CC8  3183
                        0CC8  3184 FUNCTION_EXIT:
                        0CC8  3185
                        0CC8  3186          .IF     DF      TU_TRACE
                        0CC8  3187          BSBW    TRACE_STATUS            ; Trace status.
                        0CC8  3188          .ENDC
                        0CC8  3189
    52    1C A5    D0    0CC8  3190          MOVL    CDRP$L_MSG_BUF(R5),R2   ; R2 => end message.
                 14    13    0CCC  3191          BEQL    20$                     ; EQL implies no buffer.
                 05    E0    0CCE  3192          BBS     #MSCP$V_EF_ERLOG,-      ; Branch around if error log
       05 09 A2             0CD0  3193                  MSCP$B_FLAGS(R2),10$    ;  message generated.
  0A 40 A5       02    E1    0CD3  3194          BBC     #CDRP$V_ERLIP, -        ; If no ERLOG flag in End Message and
                        0CD8  3195                  CDRP$L_DUTUFLAGS(R5), - ;  no remembered ERLIP, branch around.
                        0CD8  3196                  20$
    40 A5       04    AA    0CD8  3197 10$:    BICW    #CDRP$M_ERLIP, -        ; Clear error log in progress bit.
                        0CDC  3198                  CDRP$L_DUTUFLAGS(R5)
 00000000'GF    16    0CDC  3199          JSB     G^ERL$LOGSTATUS         ; Go log software status for errorlog.
                        0CE2  3200
    D8 A5       50    D0    0CE2  3201 20$:    MOVL    R0, CDRP$L_IOST1(R5)    ; Save final I/O status in CDRP.
                        0CE6  3202          .IF     DF      TU_SEQCHK
                        0CE6  3203          BSBB    SEQ_ENDCHECK            ; Check sequence on end.
                        0CE6  3204          .ENDC
 32 40 A5       05    E5    0CE6  3205          BBCC    #CDRP$V_DENSCK, -       ; Branch if density check not required
                        0CEB  3206                  CDRP$L_DUTUFLAGS(R5), - ; and clear required flag.
                        0CEB  3207                  30$
                        0CEB  3208          ; Use a Set Unit Characteristics command to get the current density of
                        0CEB  3209          ; the tape.  SUC is used instead of Get Unit Status because SUC is a
                        0CEB  3210          ; sequential command.  This affords a better chance of coordinating
                        0CEB  3211          ; with controller attempts to determine the density.  (Specificially,
                        0CEB  3212          ; the HSC50 needs a sequential command here.)
                        0CEB  3213          RESET_MSCP_MSG                  ; Else, setup to send another MSCP cmd.
    08 A2       0A    90    0CEE  3214          MOVB    #MSCP$K_OP_STUNT, -     ; Make that command a set unit
                        0CF2  3215                  MSCP$B_OPCODE(R2)       ; characteristics command.
0E A2       00E0 C3   B0    0CF2  3216          MOVW    UCB$W_UNIT_FLAGS(R3), - ; Must provide current unit flags
                        0CF8  3217                  MSCP$Q_UNT_FLGS(R2)     ; for SUC.
        00D8 C3         D0    0CF8  3218          MOVL    UCB$L_MSCPDEVPARAM(R3),-; Must also provide device dependent
          1C A2                 0CFC  3219                  MSCP$L_DEV_PARM(R2)     ; parameters for SUC.
                        0CFE  3220          SEND_MSCP_MSG                   ; Send the command.
                        0D01  3221          IF_MSCP FAILURE, then=30$       ; Skip is get unit status failed.
  11 09 A2       02    E0    0D07  3222          BBS     #MSCP$V_EF_PLS, -       ; Skip if correct tape position is
                        0D0C  3223                  MSCP$B_FLAGS(R2), 30$   ; not known.
                        0D0C  3224          ASSUME  MT$V_DENSITY GE 8       ; Otherwise, clear out previous
    44 A3       1F    8A    0D0C  3225          BICB    #<MT$M_DENSITY @ -8>, - ; density information.
                        0D10  3226                  UCB$L_DEVDEPEND(R3)
    50    20 A2         3C    0D10  3227          MOVZWL  MSCP$Q_FORMAT(R2), R0   ; Get MSCP density value.
             F70E       30    0D14  3228          BSBW    MSCPTOVMS_DENS          ; Convert density to VMS format.
```

```
                44 A3   05   08   50   F0  0D17  3229          INSV    R0, #MT$V_DENSITY, -    ; Store VMS density in UCB.
                                          0D1D  3230                  #MT$S_DENSITY, -
                                          0D1D  3231                  UCB$L_DEVDEPEND(R3)
                                          0D1D  3232
                          F2E0'  30       0D1D  3233 30$:     BSBW    DUTU$DEALLOC_ALL        ; Free resources owned by this CDRP.
                                          0D20  3234
                     50   D8 A5   D0      0D20  3235          MOVL    CDRP$L_IOST1(R5), R0    ; Restore final I/O status.
                     51   44 A3   D0      0D24  3236          MOVL    UCB$L_DEVDEPEND(R3),R1  ; Return to user I/O status block.
                52   00BC C3   D0        0D28  3237          MOVL    UCB$L_CDDB(R3),R2       ; R2 => CDDB.
                           00   E1        0D2D  3238          BBC     #CDDB$V_SNGLSTRM,-      ; See if in one at a time CDRP mode.
                        0A 12 A2         0D2F  3239                  CDDB$W_STATUS(R2),100$  ;  If NOT branch around PUSHAB which
                                         0D32  3240                                          ;  allows us to regain control after
                                         0D32  3241                                          ;  ALT_REQCOM.
                          52   DD        0D32  3242          PUSHL   R2                      ; Save R2 => CDDB for after ALT_REQCOM.
                          54   DD        0D34  3243          PUSHL   R4                      ; Likewise save R4 => PDT.
                 00000D42'EF   9F        0D36  3244          PUSHAB  110$                    ; Push address to which to return after
                                         0D3C  3245                                          ;  ALT_REQCOM.
                                         0D3C  3246 100$:
                                         0D3C  3247          ALT_REQCOM
                                         0D42  3248 110$:
                        54 8ED0          0D42  3249          POPL    R4                      ; Restore R4 => PDT.
                        53 8ED0          0D45  3250          POPL    R3                      ;  And R3 => CDDB.
                     013B   31           0D48  3251          BRW     RESTART_NEXT_CDRP       ; Branch to code to restart next CDRP.
                                         0D4B  3252
                                         0D4B  3253          .IF     DF      TU_SEQCHK
                                         0D4B  3254 ;+
                                         0D4B  3255 ; SEQ_ENDCHECK - routine to check that commands end in sequence.
                                         0D4B  3256 ;
                                         0D4B  3257 ; Inputs:
                                         0D4B  3258 ; R0 => final I/O status
                                         0D4B  3259 ; R3 => UCB
                                         0D4B  3260 ; R5 => CDRP
                                         0D4B  3261 ;
                                         0D4B  3262 ; Outputs:
                                         0D4B  3263 ;       All registers preserved.
                                         0D4B  3264
                                         0D4B  3265 SEQ_ENDCHECK:
                                         0D4B  3266          PUSHL   R0                      ; Save R0 for later restore.
                                         0D4B  3267          BBSC    #UCB$V_TU_OVRSQCHK,-    ; Branch around and clear bit if
                                         0D4B  3268                  UCB$W_DEVSTS(R3),10$    ;  override specified.
                                         0D4B  3269          EXTZV   #IRP$V_FCODE,-          ; Extract I/O function code.
                                         0D4B  3270                  #IRP$S_FCODE,-
                                         0D4B  3271                  CDRP$W_FUNC(R5),R0
                                         0D4B  3272          BBC     R0,SEQ_MASK,10$         ; If non-Sequential I/O branch around.
                                         0D4B  3273          CMPW    (SP),#SS$_ABORT         ; Is this an aborted command?
                                         0D4B  3274          BEQL    50$                     ; Branch if aborted command.
                                         0D4B  3275          EXTZV   #0,-                    ; Extract six bit index into array of
                                         0D4B  3276                  #6,-                    ;  IRP sequence number slots.  R0 =
                                         0D4B  3277                  UCB$B_TU_OLDINX(R3),R0  ;  index of oldest slot.
                                         0D4B  3278          INCB    UCB$B_TU_OLDINX(R3)     ; Increment index.
                                         0D4B  3279          CMPL    CDRP$L_SEQNUM(R5),-     ; Compare sequence number of this IRP to
                                         0D4B  3280                  UCB$L_TU_SEQARY(R3)[R0] ;  oldest outstanding sequence number.
                                         0D4B  3281          BNEQ    99$                     ; Branch if terminating out of sequence.
                                         0D4B  3282 10$:     POPL    R0                      ; Restore R0.
                                         0D4B  3283          RSB                             ; Return to caller.
                                         0D4B  3284
                                         0D4B  3285          ; Process canceled, aborted command.
```

```
0D4B  3286 50$:   BSBW    REMOVE_SEQARY             ; Remove aborted command from list of
0D4B  3287                                          ;  commands.
0D4B  3288        BRB     10$                       ; Then exit this routine.
0D4B  3289
0D4B  3290 99$:   BUG_CHECK        TAPECLASS,FATAL ; Sequential command has been lost.
0D4B  3291        .ENDC
```

```
0D4B  3293          .SBTTL  re-CONNECTION after VC error or failure
0D4B  3294
0D4B  3295 ; TU$CONNECT_ERR - Block of code invoked during the time that we
0D4B  3296 ;       re-CONNECT to the intelligent controller following some disturbance
0D4B  3297 ;       that caused dismanteling of the logical CONNECTION between the
0D4B  3298 ;       class driver and the controller.  The ultimate purpose of the code
0D4B  3299 ;       here is to locate all CDRP's relevant to this controller and place
0D4B  3300 ;       them in the proper order into CDDB$L_RSTRTQFL.  Once
0D4B  3301 ;       all the CDRP's are on this list we "execute" each of these CDRP's, one
0D4B  3302 ;       by one, until they are all done.  When the last such CDRP is completed
0D4B  3303 ;       we resume normal QIO processing.  This code works in cooperation with
0D4B  3304 ;       code in FUNCTION_EXIT.
0D4B  3305 ;
0D4B  3306 ;       We are invoked here either by the Port Driver calling us at our error
0D4B  3307 ;       entry point or by the Disk Class Driver branching here as a result of
0D4B  3308 ;       deciding that the intelligent controller has gone "insane".
0D4B  3309 ;
0D4B  3310 ;       The actions herein taken are the following:
0D4B  3311 ;
0D4B  3312 ;       1.  We disable the Timeout Mechanism Routine wakeups by placing a
0D4B  3313 ;               longword of all 1's in CRB$L_DUETIME.
0D4B  3314 ;
0D4B  3315 ;       2.  In order to prevent new CDRP's from starting up, we increment
0D4B  3316 ;               UCB$W_RWAITCNT for each UCB associated with this controller.
0D4B  3317 ;               This count is used to count the number of CDRP's associated
0D4B  3318 ;               with a UCB that have run into resource wait situations.
0D4B  3319 ;               Whenever this count is non-zero, new CDRP's are automatically
0D4B  3320 ;               backed up onto the UCB$L_IRPQFL queue.  Incrementing this
0D4B  3321 ;               count here, insures that it will not be run to zero and will
0D4B  3322 ;               cause all new CDRP's to backup.
0D4B  3323 ;
0D4B  3324 ;       3.  We deallocate resources owned by the permanent CDRP used by the
0D4B  3325 ;               Timeout Mechanism Routine.
0D4B  3326 ;
0D4B  3327 ;       4.  At the time that we are called here, our active CDRP's can be
0D4B  3328 ;               found in one of the following places:
0D4B  3329 ;
0D4B  3330 ;                   a) On the HIRT wait Q.  If here note that the associated UCB
0D4B  3331 ;                      RWAITCNT has been bumped due to being on this list in
0D4B  3332 ;                      addition to the bump given in step 2 above.
0D4B  3333 ;
0D4B  3334 ;                   b) On the RDT resource wait Q.  Here also RWAITCNT has been
0D4B  3335 ;                      bumped once to many times.
0D4B  3336 ;
0D4B  3337 ;                   c) On the CDDB$L_CDRPQFL.  Here RWAITCNT is normal except for
0D4B  3338 ;                      the bump given in step 1.
0D4B  3339 ;
0D4B  3340 ;                   d) On some other resource wait Q (Flow control, message buffer,
0D4B  3341 ;                      mapping resources, etc.). Here again RWAITCNT has been bumped
0D4B  3342 ;                      once to much.
0D4B  3343 ;
0D4B  3344 ;                   e) On the CDDB$L_RSTRTQ.  If here, the CONNECTION has failed
0D4B  3345 ;                      while we were in the middle of cleaning up a previous
0D4B  3346 ;                      CONNECTION failure.  The CDRP's here need no further
0D4B  3347 ;                      gathering.
0D4B  3348 ;
0D4B  3349 ;               Our aim here is to gather all the active CDRP's onto the
```

```
0D4B  3350 ;                CDDB$L_RSTRTQ.  To do this we search for them in the above
0D4B  3351 ;                mentioned places in the order in which they were mentioned.
0D4B  3352 ;                This order is important as will be explained below.
0D4B  3353 ;
0D4B  3354 ;        5.  Note here that at the time of the call to TU$CONNECT_ERR, we
0D4B  3355 ;                may have been on the middle of MOUNT VERIFICATION.  In such
0D4B  3356 ;                a case the particular volume would have been marked as
0D4B  3357 ;                invalid and during re-CONNECTION we would not try to bring
0D4B  3358 ;                the unit online.  Also we would have a set of inactive
0D4B  3359 ;                (i.e. no resources allocated for them) CDRP'a (IRP's) on
0D4B  3360 ;                the MOUNT VERIFICATION QUEUE of the UCB and possibly one
0D4B  3361 ;                MOUNT VERIFICATION specific CDRP active.  This all meshes
0D4B  3362 ;                perfectly with our re-CONNECTION design.  The contents of
0D4B  3363 ;                the MOUNT VERIFICATION QUEUE can be ignored.  The active
0D4B  3364 ;                MOUNT VERIFICATION CDRP will be treated normally.  Its
0D4B  3365 ;                I/O will be retried and will probably fail and MOUNT
0D4B  3366 ;                VERIFICATION will re-submit it and it will wind up on the
0D4B  3367 ;                normal UCB I/O QUEUE awaiting the RWAITCNT's going to zero.
0D4B  3368 ;                After re-CONNECTION, it will start up normally and everything
0D4B  3369 ;                should resume transparently.
0D4B  3370 ;
0D4B  3371 ;        6.  First we scan the HIRT wait Q and remove any CDRP's associated
0D4B  3372 ;                with the current CDDB.  We do this first so that if perchance,
0D4B  3373 ;                some of our CDRP's are here, they will not be selected
0D4B  3374 ;                inadvertantly when the current HIRT owner is possibly killed.
0D4B  3375 ;
0D4B  3376 ;                This scan is done by going down the entire HIRT wait Q and
0D4B  3377 ;                removing the 1st entry of ours that we find.  If in a pass
0D4B  3378 ;                we DO remove an entry , then we go bask and scan from the
0D4B  3379 ;                start of the Q.  When we make an entire pass without any hits,
0D4B  3380 ;                we finish.  Note that when we remove an entry, we decrement
0D4B  3381 ;                the RWAITCNT prior to calling INSERT_RSTRTQ to undo the bump
0D4B  3382 ;                we gave in calling LOCK_HIRT.
0D4B  3383 ;
0D4B  3384 ;        7.  We scan the RDT resource wait Q.  Again we scan until we find our
0D4B  3385 ;                first entry and after a removal we begin to scan from the
0D4B  3386 ;                beginning.  Only a clean scan wnds the process.  Also we
0D4B  3387 ;                must decrement RWAITCNT for each removal.
0D4B  3388 ;
0D4B  3389 ;        8.  We REMQUE each entry on CDDB$L_CDRPQFL and call INSERT_RSTRTQ
0D4B  3390 ;                for each one.
0D4B  3391 ;
0D4B  3392 ;        9.  Here we should note that INSERT_RSTRTQ deallocates all resources
0D4B  3393 ;                owned by a CDRP prior to inserting it in CDDB$L_RSTRTQ.
0D4B  3394 ;                Because of this, the only CDRP's belonging to us that still
0D4B  3395 ;                own RSPID's are the CDRP's which are on other resource wait
0D4B  3396 ;                queues.  So here we scan the RDT looking for entries that
0D4B  3397 ;                belong to us.  When we find one we REMQUE it, decrement its
0D4B  3398 ;                RWAITCNT and call INSERT_RSTRTQ for it.  Note that this
0D4B  3399 ;                deallocates its resources and as a result of this could cause
0D4B  3400 ;                another of our CDRP's to receive these resources and proceed
0D4B  3401 ;                up to the CDDB$L_CDRPQFL.  Therefore after a removal here,
0D4B  3402 ;                we branch back to step 7 to safeguard against this possibility.
0D4B  3403 ;                A complete scan of the RDT with no hits implies that we now
0D4B  3404 ;                have gathered all our CDRP's and that we can continue.
0D4B  3405 ;
0D4B  3406 ;*******************************************************************
```

```
0D4B  3407 ;       9.  If the two counts above are equal, then we have all CDRP's on
0D4B  3408 ;           CDDB$L_RSTRTQFL.  No more CDRP's will trickle in so we clear
0D4B  3409 ;           CDDB$M_CDRPTRCKL in CDDB$W_STATUS.
0D4B  3410 ;
0D4B  3411 ;       10. We DISCONNECT the now dead connection and then re-CONNECT to
0D4B  3412 ;           establish a new channel to the MSCP server in the controller.
0D4B  3413 ;
0D4B  3414 ;       11. We are now ready to begin single stream execution of CDRPs, until
0D4B  3415 ;           exhaust the contents of the CDRP$L_RSTRTQFL.  However we
0D4B  3416 ;           want to guard against the possibility that a particular
0D4B  3417 ;           request (i.e. CDRP) may repeatedly hang a controller (i.e.
0D4B  3418 ;           cause a re-CONNECTION) and thereby prevent anything from
0D4B  3419 ;           getting through.  To deal with this we only retry a given
0D4B  3420 ;           request a fixed maximum number of times (MAX_RETRY).  The
0D4B  3421 ;           algorithm which resolves this retry logic dilemma relies
0D4B  3422 ;           on several data items in the CDDB:
0D4B  3423 ;
0D4B  3424 ;                   a) CDDB$L_RSTRTCDRP - the address of the CDRP that is
0D4B  3425 ;                       currently being processed in single stream mode if
0D4B  3426 ;                       we are in single stream mode.
0D4B  3427 ;
0D4B  3428 ;                   b) CDDB$B_RETRYCNT - the number of remaining retries
0D4B  3429 ;                       for the current CDRP being processes in single
0D4B  3430 ;                       stream mode if we are in single stream mode.
0D4B  3431 ;
0D4B  3432 ;                   c) CDDB$V_SNGLSTRM - bit in CDDB$W_STATUS which tells
0D4B  3433 ;                       us if we are in single stream mode.
0D4B  3434 ;
0D4B  3435 ;           The algorithm is as follows: If upon selecting the first CDRP
0D4B  3436 ;           on CDDB$L_RSTRTQFL, we find CDDB$V_SNGLSTRM clear, we merely
0D4B  3437 ;           set it and we can be assurred that this is the first time
0D4B  3438 ;           that we are attempting to retry this request in single stream
0D4B  3439 ;           mode.  This is so because the bit being clear implies either
0D4B  3440 ;           that this is the first re-CONNECTION since the system came up
0D4B  3441 ;           or that the last re-CONNECTION ran to completion thereby leaving
0D4B  3442 ;           the bit clear.  In this case we select this first CDRP, set
0D4B  3443 ;           CDDB$B_RETRYCNT to the maximum and establish this CDRP as the
0D4B  3444 ;           current one by storing its address in CDDB$L_RSTRTCDRP.
0D4B  3445 ;
0D4B  3446 ;           If however CDDB$V_SNGLSTRM is set upon selecting a CDRP, we
0D4B  3447 ;           must compare the CDRP address to the current value of
0D4B  3448 ;           CDDB$L_RSTRTCDRP.  If they are NOT equal, then again this is
0D4B  3449 ;           the first retry attempt for this CDRP and we merely set the
0D4B  3450 ;           CDDB$B_RETRYCNT to the maximum and store the CDRP in
0D4B  3451 ;           CDDB$L_RSTRTCDRP.  If the CDRP has the same address however,
0D4B  3452 ;           we must decrement one from the retry count and if it is not
0D4B  3453 ;           exhausted attempt to process the CDRP again.
0D4B  3454 ;
0D4B  3455 ;           Note this all works even though the address of a CDRP is not
0D4B  3456 ;           necessarily unique. That is, many I/O requests in the life of
0D4B  3457 ;           the system may occupy the same CDRP in virtual space.  However,
0D4B  3458 ;           once re-CONNECTION logic begins, it deals only with the CDRPs
0D4B  3459 ;           on the CDDB$L_RSTRTQFL.  This list never grows until re-
0D4B  3460 ;           CONNECTION is run to completion since all new IRPs are
0D4B  3461 ;           being backed up.  Therefore even though we may run repeated
0D4B  3462 ;           re-CONNECTIONs that do not run to completion but rather each
0D4B  3463 ;           causes the connection to go down, through all this the
```

```
                     0D4B  3464 ;                  CDDB$L_RSTRTQFL is always monotonically decreasing and no
                     0D4B  3465 ;                  new CDRPs are entered onto it that were not there at the time
                     0D4B  3466 ;                  that we began to process the first re-CONNECTION.  In a fixed
                     0D4B  3467 ;                  list of CDRPs which all exist at the same time, the address
                     0D4B  3468 ;                  is a unique descriptor.
                     0D4B  3469 ;
                     0D4B  3470 ;           12. Note that CDDB$M_SNGLSTRM in CDDB$W_STATUS acts as a flag to
                     0D4B  3471 ;                  FUNCTION_EXIT so that it can aid in the one at a time re-
                     0D4B  3472 ;                  execution of the CDRP's.
                     0D4B  3473 ;
                     0D4B  3474 ;           13. For debugging sake, we loop thru all UCB's and check that their
                     0D4B  3475 ;                  UCB$W_RWAITCNT values are all equal to 1.
                     0D4B  3476 ;                  Also for debugging sake we check that CDDB$L_CDRPQFL is
                     0D4B  3477 ;                  empty.
                     0D4B  3478 ;
                     0D4B  3479 ;           14. We REMQUE the 1st CDRP on CDDB$L_RSTRTQFL and branch to
                     0D4B  3480 ;                  TU_RESTARTIO to begin its execution.
                     0D4B  3481 ;
                     0D4B  3482 ; Inputs: (for TU$RE_SYNCH)
                     0D4B  3483 ;          R3 => CRB
                     0D4B  3484
                     0D4B  3485
                     0D4B  3486 TU$RE_SYNCH:
                     0D4B  3487
53   10 A3    D0     0D4B  3488          MOVL    CRB$L_AUXSTRUC(R3),R3    ; R3 => CDDB.
54   14 A3    D0     0D4F  3489          MOVL    CDDB$L_PDT(R3),R4        ; R4 => PDT.
26 A3  04     91     0D53  3490          CMPB    #MSCP$K_CM_EMULA,-       ; If this is the MSCP server, the right
                     0D57  3491                  CDDB$B_CNTRLMDL(R3)      ; resynch technique is DISCONNECT.
       0A     13     0D57  3492          BEQL    RECONN_COMMON            ; So, skip the MRESET setup.
       10     A8     0D59  3493          BISW    #CDDB$M_RESYNCH,-        ; Signal that we should reset
     12 A3            0D5B  3494                  CDDB$W_STATUS(R3)        ;  intelligent controller.
       04     11     0D5D  3495          BRB     RECONN_COMMON            ; Branch around to common code.
                     0D5F  3496
                     0D5F  3497 ; Inputs: (for TU$CONNECT_ERR)
                     0D5F  3498 ;          R3 => CDT
                     0D5F  3499 ;          R4 => PDT
                     0D5F  3500 ;
                     0D5F  3501
                     0D5F  3502 TU$CONNECT_ERR:
                     0D5F  3503
     53   5C A3    D0     0D5F  3504          MOVL    CDT$L_AUXSTRUC(R3),R3    ; R3 => CDDB.
                     0D63  3505 RECONN_COMMON:
          3A A3     B6     0D63  3506          INCW    CDDB$W_RSTRTCNT(R3)      ; Count number of times reconnected.
                    AA     0D66  3507          BICW    #<CDDB$M_IMPEND -        ; Signal: no immediate command pending
                     0D67  3508                   !CDDB$M_INITING -          ;         out of initialization
                     0D67  3509                   !CDDB$M_SNGLSTRM -         ;         no single stream in progress
                     0D67  3510                   !CDDB$M_RSTRTWAIT>,-       ;         not waiting to restart CDRPs
 12 A3   0107 8F     0D67  3511                  CDDB$W_STATUS(R3)
                     0D6C  3512
     50   18 A3    D0     0D6C  3513          MOVL    CDDB$L_CRB(R3),R0        ; R0 => CRB.
     18 A0    01     CE     0D70  3514          MNEGL   #1,CRB$L_DUETIME(R0)     ; Prevent Timeout Mechanism wakeups.
                     0D74  3515
          08     A8     0D74  3516          BISW    #CDDB$M_RECONNECT,-      ; Set bit meaning that we are in
          12 A3            0D76  3517                  CDDB$W_STATUS(R3)        ;  the re-CONNECTING state.
                     0D78  3518
53   0000007C 8F    C3     0D78  3519          SUBL3   #<UCB$L_CDDB_LINK -      ; Get "previous" UCB address in R1.
                     0D7F  3520                   -CDDB$L_UCBCHAIN>, -
```

```
                51        0D7F  3521                    R3, R1
                          0D80  3522
 51    00C4 C1    D0      0D80  3523 10$:     MOVL     UCB$L_CDDB_LINK(R1), R1 ; Chain to next UCB (if any).
                0A  13     0D85  3524              BEQL     20$                     ; EQL implies no more UCB's here.
F4 68 A1        0A  E2     0D87  3525              BBSS     #UCB$V_MSCP_WAITBMP, -  ; Only bump RWAITCNT once.  If already
                          0D8C  3526                       UCB$W_DEVSTS(R1), 10$   ;  bumped, branch back.
              56 A1  B6     0D8C  3527              INCW     UCB$W_RWAITCNT(R1)      ; Prevent new CDRP's from starting up.
                EF  11     0D8F  3528              BRB      10$                     ; Go look for more UCB's.
                          0D91  3529 20$:
                          0D91  3530
                          0D91  3531 ;
                          0D91  3532 ; Now we are sure that no new CDRP's will start.
                          0D91  3533 ;
                          0D91  3534
              F26C'  30     0D91  3535              BSBW     DUTU$DISCONNECT_CANCEL  ; Perform disconnect cancel cleanup.
                          0D94  3536
                          0D94  3537 ; Deallocate RSPID & message buffer on each of the CDDB perm. IRP/CDRP pairs.
                          0D94  3538
 55    0194 C3    9E      0D94  3539              MOVAB    CDDB$A_DAPCDRP(R3), R5  ; Get DAP permanent CDRP address.
                F264'  30     0D99  3540              BSBW     DUTU$DEALLOC_RSPID_MSG  ; Deallocate its RSPID & msg. buf.
 55    00D0 C3    9E      0D9C  3541              MOVAB    CDDB$A_PRMCDRP(R3), R5  ; Get permanent CDRP address.
                F25C'  30     0DA1  3542              BSBW     DUTU$DEALLOC_RSPID_MSG  ; Deallocate its RSPID & msg. buf.
                          0DA4  3543
                          0DA4  3544 ;
                          0DA4  3545 ;       Registers here are:
                          0DA4  3546 ;               R3 => CDDB
                          0DA4  3547 ;               R4 => PDT.
                          0DA4  3548 ;
                          0DA4  3549
                          0DA4  3550 ; Locate and prepare for restarting all CDRPs currently waiting for a RSPID.
                          0DA4  3551 ; Since the class driver allocates a RSPID as the first step in any function,
                          0DA4  3552 ; CDRPs found now will not be holding any resources and will not be active.
                          0DA4  3553 ; Since these CDRPs hold no resources, their cleanup will not cause any other
                          0DA4  3554 ; waiting requests to becom active.  (This fact is not currently used, but it
                          0DA4  3555 ; might be useful.)
                          0DA4  3556
 53    00F4 C3    D0      0DA4  3557              MOVL     CDDB$L_CDT(R3), R3              ; Get CDT address.
                          0DA9  3558
                51  D4     0DA9  3559              CLRL     R1                              ; Set SCAN_RSPID_WAIT flag.
                          0DAB  3560              SCAN_RSPID_WAIT -                       ; Use SCS service to scan RSPID
                          0DAB  3561                       action = DUTU$RECONN_LOOKUP    ; wait queue.
                          0DB8  3562                                                      ; DUTU$RECONN_LOOKUP is in
                          0DB8  3563                                                      ; DUTUSUBS.
                          0DB8  3564
                          0DB8  3565 ; Remove all CDRPs on the active requests queue.  These CDRPs:
                          0DB8  3566 ;   a. have outstanding requests in the irtelligent controller,
                          0DB8  3567 ;   b. suffered allocation failures due to a broken connection,
                          0DB8  3568 ;   c. represent the request during which an "insane" controller was detected.
                          0DB8  3569 ; In any case, these CDRPs are not on any resource wait queue and do not have
                          0DB8  3570 ; their associated resource wait count bumped due to need for a resource.
                          0DB8  3571
                F245'  30     0DB8  3572              BSBW     DUTU$DRAIN_CDDB_CDRPQ           ; Cleanup active requests.
                          0DBB  3573
                          0DBB  3574 ; Now scan the entire Response-id Descriptor Table for any remaining CDRPs
                          0DBB  3575 ; belonging to this connection.  Presumably these CDRPs are on a resource wait
                          0DBB  3576 ; queue somewhere.  In addtion, releasing whatever resources such CDRPs hold
                          0DBB  3577 ; may cause other waiting CDRPs to become active.  Therefore, after every CDRP
```

```
                     0DBB  3578 ; is located and processed, the active CDRP queue must be scanned again.
                     0DBB  3579
           51   D6   0DBB  3580          INCL    R1                              ; Set SCAN_RDT flag.
                     0DBD  3581          SCAN_RDT -                              ; Use SCS service to scan RDT.
                     0DBD  3582                  action = DUTU$RECONN_LOOKUP     ; DUTU$RECONN_LOOKUP is in
                     0DCA  3583                                                  ; DUTUSUBS.
                     0DCA  3584
     53    5C A3   D0   0DCA  3585          MOVL    CDT$L_AUXSTRUC(R3), R3          ; Restore the CDDB address.
                     0DCE  3586
                     0DCE  3587 RESTART_FIRST_CDRP:
                     0DCE  3588
                     0DCE  3589 ;
                     0DCE  3590 ; We come here either by falling thru from above code or by branching here
                     0DCE  3591 ;       from CALL_SEND_MSG_BUF when the last CDRP has trickled in.
                     0DCE  3592 ;
                     0DCE  3593 ;
                     0DCE  3594 ; If here all CDRP's are in CDDB$L_RSTRTQFL.  So no more will trickle.
                     0DCE  3595 ;       Clear bit that prevents CALL_SEND_MSG_BUF from doing its job.
                     0DCE  3596 ;
                     0DCE  3597 ; INPUTS:
                     0DCE  3598 ;       R3 => CDDB
                     0DCE  3599 ;       R4 => PDT
                     0DCE  3600 ;
                     0DCE  3601
                     0DCE  3602
                     0DCE  3603 ;
                     0DCE  3604 ; Here we DISCONNECT the old connection.
                     0DCE  3605 ;
                     0DCE  3606
     55    00D0 C3   9E   0DCE  3607          MOVAB   CDDB$A_PRMCDRP(R3),R5   ; Put R5 => CDRP for coming BSBWs.
           50    53   D0   0DD3  3608          MOVL    R3,R0                   ; R0 => CDDB.
        53    24 A5   D0   0DD6  3609          MOVL    CDRP$L_CDT(R5),R3       ; Set R3 => CDT.
  12 A0    0080 8F   A8   0DDA  3610          BISW    #CDDB$M_NOCONN,-        ; Set no connection active flag.
                     0DE0  3611                  CDDB$W_STATUS(R0)
                04   E5   0DE0  3612          BBCC    #CDDB$V_RESYNCH,-       ; Do NOT branch around if we were called
           1C 12 A0        0DE2  3613                  CDDB$W_STATUS(R0),2$    ;  in order to re-synchronize.
        53    1C A3   D0   0DE5  3614          MOVL    CDT$L_PB(R3),R3         ; R3 => Path Block for MRESET, etc.
                     0DE9  3615          MRESET  PB$B_RSTATION(R3),#1    ; Force controller to reset itself.
                     0DF3  3616          MSTART  PB$B_RSTATION(R3)       ; And force controller to restart itself.
                05   0E00  3617          RSB                             ; Kill this thread. Rely on Port
                     0E01  3618                                          ;  Driver calling error routine as
                     0E01  3619                                          ;  a result of MRESET to accomplish
                     0E01  3620                                          ;  DISCONNECT and subsequent logic.
                     0E01  3621 2$:
                     0E01  3622          DISCONNECT      #DISCONNECT_REASON
                     0E0A  3623
                     0E0A  3624          PERMCDRP_TO_CDDB -                      ; Get CDDB address in R3.
                     0E0A  3625                  cdrp=R5, cddb=R3
                     0E11  3626
                     0E11  3627 ;
                     0E11  3628 ; Deallocate mapping resources
                     0E11  3629 ; and queue mount verification requests for post processing
                     0E11  3630 ; <<< The mount verification references have been commented out in the  >>>
                     0E11  3631 ; <<< following lines.  This driver does not do mount verification.     >>>
                     0E11  3632 ; <<< When it is taught to do mount verification, however, the comment- >>>
                     0E11  3633 ; <<< ed lines MUST be restored.                                        >>>
                     0E11  3634 ;
```

```
                     0E11  3635
                     0E11  3636          ; Any mapping resources still owned by CDRPs on the restart queue are
                     0E11  3637          ; deallocated here.  This deallocation is delayed until after the
                     0E11  3638          ; DISCONNECT (and possible MRESET) to prevent an "insane" controller
                     0E11  3639          ; from continuing to transfer via possibly re-allocated mapping
                     0E11  3640          ; resources.  The mount verification queueing is delayed because the
                     0E11  3641          ; mount verification operation may be holding mapping resources.
                     0E11  3642
          3C A3  9F  0E11  3643          PUSHAB  CDDB$L_RSTRTQFL(R3)     ; Setup listhead address.
          3C A3  DD  0E14  3644          PUSHL   CDDB$L_RSTRTQFL(R3)     ; Setup first CDRP address.
                     0E17  3645
       55 8ED0       0E17  3646 4$:     POPL    R5                      ; Get next CDRP address.
    6E    55  D1     0E1A  3647          CMPL    R5, (SP)                ; Is it the listhead?
       07  13        0E1D  3648          BEQL    6$                      ; If yes, all deallocations are done.
   F1DE'  30        0E1F  3649          BSBW    DUTU$DEALLOC_ALL        ; Free MAP resources owned by this CDRP.
       65  DD        0E22  3650          PUSHL   (R5)                    ; Push next CDRP address.
                     0E24  3651 ;<<<     BBC     #IRP$V_MVIRP, -         ; Is this a mount verification IRP?
                     0E24  3652 ;<<<             CDRP$W_STS(R5), 4$      ; Branch if not an MV IRP.
                     0E24  3653 ;<<<     REMQUE  (R5), R0                ; Else, remove IRP/CDRP from restart
                     0E24  3654 ;<<<     POST_CDRP status=SS$_MEDOFL     ; queue and send it to post processing.
       F1  11        0E24  3655          BRB     4$                      ; Loop till all restart CDRPs are done.
                     0E26  3656
       8E  D5        0E26  3657 6$:     TSTL    (SP)+                   ; Clear listhead pointer from stack.
                     0E28  3658
                     0E28  3659          ; Deallocate mapping resources whose description is stored in the
                     0E28  3660          ; CDDB permanent CDRP.  This information was placed there by
                     0E28  3661          ; DUTU$INSERT_RESTARTQ when it discovered that the HIRT permanent CDRP
                     0E28  3662          ; owned mapping resources.  In this way, another thread is allowed to
                     0E28  3663          ; use the HIRT permanent CDRP while this connection is broken.
                     0E28  3664
 55  00D0 C3  9E     0E28  3665          MOVAB   CDDB$A_PRMCDRP(R3), R5  ; Get CDRP in R5.
        F1D0'  30     0E2D  3666          BSBW    DUTU$DEALLOC_ALL        ; Free old HIRT MAP resources.
                     0E30  3667                                          ;  the HIRT CDRP and whose ownership
                     0E30  3668                                          ;  has been transferred here.
                     0E30  3669
                     0E30  3670
                     0E30  3671 ;
                     0E30  3672 ; re-CONNECT - Here we call an internal subroutine which:
                     0E30  3673 ;
                     0E30  3674 ;       1. Makes a connection to the MSCP server in the intelligent
                     0E30  3675 ;               controller.
                     0E30  3676 ;
                     0E30  3677 ;       2. Sends an MSCP command to SET CONTROLLER CHARACTERISTICS.
                     0E30  3678 ;
                     0E30  3679 ;       3. Allocates an MSCP buffer and RSPID for our future use in
                     0E30  3680 ;               connection management.
                     0E30  3681 ;
                     0E30  3682 ; Upon return R4 => PDT and R5 => CDRP.
                     0E30  3683 ;
                     0E30  3684
        F34E  30     0E30  3685          BSBW    MAKE_CONNECTION         ; Call subroutine to connect.
                     0E33  3686
                     0E33  3687          PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                     0E33  3688                  cdrp=R5, cddb=R3
    50   18 A3  D0     0E3A  3689          MOVL    CDDB$L_CRB(R3),R0       ; Get CRB address.
1C A0   0EF0'CF  9E     0E3E  3690          MOVAB   W^TU$TMR, -             ; Establish permanent timeout routine.
                     0E44  3691                  CRB$L_TOUTROUT(R0)
```

```
                51   2A A3    3C  0E44  3692          MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Get controller timeout interval.
      18 A0  00000000'GF  51    C1  0E48  3693          ADDL3   R1, G^EXE$GL_ABSTIM, - ; Use that to set next timeout
                                    0E51  3694                  CRB$L_DUETIME(R0)       ; wakeup time.
                                    0E51  3695
                                    0E51  3696          ; The normal MSCP timeout mechanism is now in effect.  Henceforth,
                                    0E51  3697          ; no fork thread may use the CDDB permanent CDRP as a fork block.
                                    0E51  3698
                                    0E51  3699          ASSUME  CDDB$V_DAPBSY GE 8
                13 A3    04    88  0E51  3700          BISB    #<CDDB$M_DAPBSY@-8>, -; Set DAP CDRP in use flag.
                                    0E55  3701                  CDDB$W_STATUS+1(R3)
                55    54 A3    D0  0E55  3702          MOVL    CDDB$L_DAPCDRP(R3), R5  ; Get DAP CDRP address.
                      F1A4'    30  0E59  3703          BSBW    DUTU$POLL_FOR_UNITS     ; Interrogate controller, poll for units.
                                    0E5C  3704                                          ;   Returns R3 => CDDB, R5 => CDRP.
                                    0E5C  3705
                                    0E5C  3706          ; Now it is necessary to propogate all the connection dependent
                                    0E5C  3707          ; information regarding the newly formed connection to the MSCP server
                                    0E5C  3708          ; to all the UCB's in the primary chain for this CDDB.  At the same
                                    0E5C  3709          ; time, every RWAITCNT value is tested to insure that it is consistant
                                    0E5C  3710          ; with what would be expected based upon the various possible reasons
                                    0E5C  3711          ; which cause it to be bumped.  This is merely a debugging exercise.
                                    0E5C  3712          ; In END_SINGLE_STREAM, RWAITCNT will be reduced by one and the wait
                                    0E5C  3713          ; count bumped flag will be cleared.
                                    0E5C  3714
                                    0E5C  3715          ; This loop also brings previously valid units online, an activity
                                    0E5C  3716          ; which would be performed by mount verification if it existed.
                                    0E5C  3717
                                    0E5C  3718          ; This loop also initializes previously uninitialized trace tables.
                                    0E5C  3719          ; This must be performed after the call to DUTU$POLL_FOR_UNITS.
                                    0E5C  3720
                55    84 A3    9E  0E5C  3721          MOVAB   <CDDB$L_UCBCHAIN -      ; Setup "previous" UCB address.
                                    0E60  3722                  -UCB$L_CDDB_LINK>(R3), -
                                    0E60  3723                  R5
              55    00C4 C5    D0  0E60  3724 15$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Link to next UCB.
                        10    13  0E65  3725          BEQL    30$                     ; Branch if no more UCBs to test.
                           F196' 30  0E67  3726          BSBW    DUTU$INIT_CONN_UCB      ; Setup connection dep. UCB fields.
                                    0E6A  3727          .IF     DEFINED TU_TRACE
                                    0E6A  3728          BSBW    TRACE_INIT              ; Init IRP trace table.
                                    0E6A  3729          .ENDC
                           F193' 30  0E6A  3730          BSBW    DUTU$CHECK_RWAITCNT     ; Validate the wait count value.
            EE 64 A5    0B    E1  0E6D  3731          BBC     #UCB$V_VALID, -         ; If unit is not valid, all done
                                    0E72  3732                  UCB$L_STS(R5), 15$      ; for now.
                           F4CB 30  0E72  3733          BSBW    BRING_UNIT_ONLINE       ; Else, bring the unit back online.
                             E9 11  0E75  3734          BRB     15$                     ; Loop through all UCBs.
                                    0E77  3735
                                    0E77  3736 30$:     ; If this driver performed mount verification, it would now be
                                    0E77  3737          ; possible to execute requests on behalf of any pending mount
                                    0E77  3738          ; verification threads.  Therefore, the CDDB$V_NOCONN bit is
                                    0E77  3739          ; cleared here.
                                    0E77  3740          ;
                                    0E77  3741          ; Since all threads which use the DAP CDRP as a fork block are now
                                    0E77  3742          ; completed, that block may now be used for DAP operations.
                                    0E77  3743          ; Therefore, the DAP CDRP busy flags is cleared too.
                                    0E77  3744
          12 A3    0480 8F    AA  0E77  3745          BICW    #<CDDB$M_NOCONN -       ; Clear no-connection and
                                    0E7D  3746                   !CDDB$M_DAPBSY>, -     ; DAP-CDRP-busy flags.
                                    0E7D  3747                  CDDB$W_STATUS(R3)
                                    0E7D  3748
```

```
                     0E7D  3749          ; Processing of the first CDRP in the restart queue is about to begin.
                     0E7D  3750          ; The queue of active requests should be empty:  check it.  N.B. if
                     0E7D  3751          ; volume revalidation were being performed by mount verification, the
                     0E7D  3752          ; active request queue might not be empty and it would be necessary to
                     0E7D  3753          ; synchronize with mount verification activities as is done in the
                     0E7D  3754          ; disk class driver.
                     0E7D  3755
                     0E7D  3756          ASSUME  CDDB$L_CDRPQFL EQ 0
53   63   D1         0E7D  3757          CMPL    (R3), R3                ; Empty listheads point to themselves.
     04   13         0E80  3758          BEQL    RESTART_NEXT_CDRP       ; EQL implies that all is correct.
                     0E82  3759          BUG_CHECK       TAPECLASS,FATAL
                     0E86  3760
                     0E86  3761
                     0E86  3762 RESTART_NEXT_CDRP:
                     0E86  3763
                     0E86  3764 ;
                     0E86  3765 ; Here we attempt to initiate the first (i.e. next) CDRP on the restart queue.
                     0E86  3766 ;       In order to prevent getting caught in an infinite loop trying to
                     0E86  3767 ;       initiate an operation that the controller cannot complete for
                     0E86  3768 ;       one reason or another, we maintain a retry count and the address
                     0E86  3769 ;       of the CDRP that we are currently single streaming.
                     0E86  3770 ;
                     0E86  3771 ;       In the normal case this is an isolated re-CONNECTION and the
                     0E86  3772 ;       first CDRP on the restart queue is a random CDRP.  We notice this
                     0E86  3773 ;       by seeing that the address of our first CDRP is not equal to the
                     0E86  3774 ;       current contents of CDDB$L_RSTRTCDRP.
                     0E86  3775 ;
                     0E86  3776 ;       In the other case the connection failed while we were in single
                     0E86  3777 ;       stream mode and the CDRP which we happened to be processing is the
                     0E86  3778 ;       same CDRP that now heads our restart queue.  In this case, before
                     0E86  3779 ;       initiating the processing of this CDRP, we decrement 1 from the
                     0E86  3780 ;       retry count and if it remains non-zero, we restart the CDRP
                     0E86  3781 ;       processing.  If the decrementing results in a zero retry count,
                     0E86  3782 ;       then we log the event and effectively abort the CDRP by branching to
                     0E86  3783 ;       FUNCTION_EXIT with an appropriate error status.  FUNCTION_EXIT, due
                     0E86  3784 ;       to the setting of the CDDB$M_SNGLSTRM bit will then start the
                     0E86  3785 ;       processing of the next CDRP on the restart queue.
                     0E86  3786 ;
                     0E86  3787 ; We can arrive here either by falling through from the above code or via
                     0E86  3788 ;       a branch from FUNCTION_EXIT.  In either case we have:
                     0E86  3789 ;
                     0E86  3790 ; INPUT:
                     0E86  3791 ;       R3 => CDDB
                     0E86  3792 ;
                     0E86  3793
55   3C B3   0F      0E86  3794          REMQUE  @CDDB$L_RSTRTQFL(R3),R5 ; R5 => 1st CDRP on restart queue.
         2F   1D     0E8A  3795          BVS     END_SINGLE_STREAM       ; VS implies restart was empty.
         00   E3     0E8C  3796          BBCS    #CDDB$V_SNGLSTRM,-      ; Set bit and if clear, this is 1st
   1B 12 A3           0E8E  3797                  CDDB$W_STATUS(R3),20$   ;  time here for this CDRP, so branch.
34 A3    55   D1     0E91  3798          CMPL    R5,CDDB$L_RSTRTCDRP(R3) ; See if same CDRP as last time.
         15   12     0E95  3799          BNEQ    20$                     ; NEQ implies not the same.
      38 A3   97     0E97  3800          DECB    CDDB$B_RETRYCNT(R3)     ; If same, decrement 1 from retries.
         18   12     0E9A  3801          BNEQ    30$                     ; NEQ implies retries remaining.
                     0E9C  3802
                     0E9C  3803 ;
                     0E9C  3804 ; ********************************Log this error.********************************
                     0E9C  3805 ;
```

```
                     0E9C  3806
          50 00000054 8F  D0 0E9C  3807          MOVL    #SS$_CTRLERR,R0          ; Indicate appropriate error status.
                       51 D4 0EA3  3808          CLRL    R1                       ;  And set second part of I/O status.
                  53 BC A5 D0 0EA5  3809          MOVL    CDRP$L_UCB(R5),R3        ; R3 => UCB.
                    FE1C 31 0EA9  3810          BRW     FUNCTION_EXIT
                         0EAC  3811 20$:
                  34 A3  55 D0 0EAC  3812          MOVL    R5,CDDB$L_RSTRTCDRP(R3) ; Establish new single stream CDRP.
                       02 90 0EB0  3813          MOVB    #MAX_RETRY,-             ; Establish fresh retry count.
                     38 A3       0EB2  3814                  CDDB$B_RETRYCNT(R3)
                         0EB4  3815 30$:
                  53 BC A5 D0 0EB4  3816          MOVL    CDRP$L_UCB(R5),R3        ; R3 => UCB.
                    F710 31 0EB8  3817          BRW     TU_RESTARTIO             ; Restart the CDRP.
                         0EBB  3818
                         0EBB  3819
                         0EBB  3820 END_SINGLE_STREAM:
                         0EBB  3821
                         0EBB  3822 ;
                         0EBB  3823 ; Here we want to resume normal operation and get each unit going.
                         0EBB  3824 ;       To do this we pickup each UCB in turn and call SCS$UNSTALLUCB
                         0EBB  3825 ;       for it.  This has the effect of starting up as many (perhaps all)
                         0EBB  3826 ;       of the IRP's (that's right IRP's) as possible that may have
                         0EBB  3827 ;       backed up on the UCB input queue while we were in single stream mode.
                         0EBB  3828 ;       We then go on to the next UCB until we exhaust all UCB's connected
                         0EBB  3829 ;       to this CDDB.
                         0EBB  3830 ;
                         0EBB  3831
                  12 A3  01  AA 0EBB  3832          BICW    #CDDB$M_SNGLSTRM, -      ; Clear single streaming CDRPs flag.
                         0EBF  3833                  CDDB$W_STATUS(R3)
                  50  3A A3  3C 0EBF  3834          MOVZWL  CDDB$W_RSTRTCNT(R3), R0 ; Get current restart count.
                  55  84 A3  9E 0EC3  3835          MOVAB   <CDDB$L_UCBCHAIN -       ; Setup "previous" UCB address.
                         0EC7  3836                  -UCB$L_CDDB_LINK>(R3), -
                         0EC7  3837                  R5
                         0EC7  3838
               55  00C4 C5  D0 0EC7  3839 10$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Point to next UCB.
                       1D 13 0ECC  3840          BEQL    30$                      ; Branch if no more UCBs to process.
            68 A5  0400 8F  AA 0ECE  3841          BICW    #UCB$M_MSCP_WAITBMP, -   ; Indicate RWAITCNT no longer bumped.
                         0ED4  3842                  UCB$W_DEVSTS(R5)
                     56 A5  B7 0ED4  3843          DECW    UCB$W_RWAITCNT(R5)       ; Unbump wait count.
                      F126' 30 0ED7  3844          BSBW    DUTU$CHECK_RWAITCNT      ; Else, check wait count and
                       09 BB 0EDA  3845          PUSHR   #^M<R0,R3>               ; Save restart cnt. and CDDB address.
                00000000'GF  16 0EDC  3846          JSB     G^SCS$UNSTALLUCB         ; Start up IRPs on UCB.
                       09 BA 0EE2  3847          POPR    #^M<R0,R3>               ; Restore restart cnt. and CDDB address.
                  3A A3  50 B1 0EE4  3848          CMPW    R0, CDDB$W_RSTRTCNT(R3) ; Did the unstall cause a restart?
                       DD 13 0EE8  3849          BEQL    10$                      ; Branch if no restart was caused.
                       05 0EEA  3850          RSB                              ; Else, discontinue this thread.
                         0EEB  3851
                  12 A3  08  AA 0EEB  3852 30$:     BICW    #CDDB$M_RECONNECT, -     ; Clear reconnect in progress bit.
                         0EEF  3853                  CDDB$W_STATUS(R3)
                       05 0EEF  3854          RSB                              ; Ta De, Ta De, that's all folks.
```

```
0EF0  3856          .SBTTL  TU$TMR - Class Driver Timeout Mechanism Routine
0EF0  3857
0EF0  3858 ;+
0EF0  3859 ; TU$TMR - Time out Mechanism Routine.  This routine is called
0EF0  3860 ;  periodically whenever CRB$L_DUETIME becomes due.  At the time of a
0EF0  3861 ;  periodic call to TU$TMR the Class Driver is in one of three states
0EF0  3862 ;  with respect to the intelligent mass storage controller associated
0EF0  3863 ;  with the CRB pointed at by R3.
0EF0  3864 ;
0EF0  3865 ;       1. State #1, the "normal" state for which this routine is optimized,
0EF0  3866 ;          is characterized by the following two conditions:
0EF0  3867 ;
0EF0  3868 ;              a) One or more MSCP commands are outstanding to the controller.
0EF0  3869 ;                 This is determined by having a NON-empty queue of CDRP's
0EF0  3870 ;                 hanging off the CDDB.
0EF0  3871 ;
0EF0  3872 ;              b) The oldest outstanding command was initiated since the
0EF0  3873 ;                 previous invocation of TU$TMR and is therefore not very
0EF0  3874 ;                 old.  This is determined by comparing the RSPID of the
0EF0  3875 ;                 currently oldest command to the RSPID of the oldest request
0EF0  3876 ;                 at the time of the previous invocation.  If they are not
0EF0  3877 ;                 equal then we are in State #1.
0EF0  3878 ;
0EF0  3879 ;       2. State #2 is characterized by having NO outstanding MSCP commands in
0EF0  3880 ;          the controller.  This is determined by finding an empty CDRP queue
0EF0  3881 ;          in the CDDB.
0EF0  3882 ;
0EF0  3883 ;       3. State #3 is the state where MSCP commands are outstanding and the
0EF0  3884 ;          oldest one has been outstanding for at least one previous TU$TMR
0EF0  3885 ;          invocation.
0EF0  3886 ;
0EF0  3887 ; If we determine that we are in state #1, we simply record the RSPID of the
0EF0  3888 ;  currently oldest outstanding MSCP command in CDDB$L_OLDRSPID and we initial-
0EF0  3889 ;  ize CDDB$L_OLDCMDSTS to all 1's.  We then calculate a new due time,
0EF0  3890 ;  place it in CRB$L_DUETIME and return to our caller, which results
0EF0  3891 ;  in scheduling ourselves for the next invocation of TU$TMR.
0EF0  3892 ;
0EF0  3893 ; States #2 and #3 share some common code.  In both cases we will issue an
0EF0  3894 ;  IMMEDIATE command to the controller but for diverse reasons.  In the case
0EF0  3895 ;  of state #2 it will be an effective NOP command that is only issued to
0EF0  3896 ;  insure against the controller timing out the host (i.e. us) due to lack of
0EF0  3897 ;  activity on our part.  In the case of state #3, the IMMEDIATE command will
0EF0  3898 ;  be a "GET COMMAND STATUS" for the oldest outstanding MSCP command.
0EF0  3899 ;
0EF0  3900 ; The common code they share consists of code to appropriate the pre-allocated
0EF0  3901 ;  MSCP buffer pointed at by CDRP$L_MSG_BUF and to pick up the pre-allocated
0EF0  3902 ;  RSPID identified by CDRP$L_RSPID.  BOth these items are located in
0EF0  3903 ;  the permanent CDRP which is appended to the CDDB of this intelligent
0EF0  3904 ;  controller.  Also at this time a new due time is calculated prior to
0EF0  3905 ;  doing the DRIVER_SEND_MSG so that we will be able to time out the
0EF0  3906 ;  Immediate command.  Then the code for these two states diverges for
0EF0  3907 ;  a while to prepare distinct MSCP packets, do the SEND_MSG_BUF,
0EF0  3908 ;  and in the case of state #3, to do some specific processing upon
0EF0  3909 ;  receipt of the END PACKET for the IMMEDIATE command.  This processing
0EF0  3910 ;  consists of insuring that the command status returned in the END PACKET
0EF0  3911 ;  indicates progress being made on the oldest outstanding command; and also
0EF0  3912 ;  of saving this received command status in the CDDB$L_OLDCMDSTS so as to
```

```
                     0EF0  3913 ;   have it available at the next invocation, if this oldest command is still
                     0EF0  3914 ;   outstanding.  Following this the two code paths converge to recycle the
                     0EF0  3915 ;   received END PACKET for use as the next IMMEDIATE MSCP buffer and to also
                     0EF0  3916 ;   recycle the RSPID by bumping its sequence number.
                     0EF0  3917 ;
                     0EF0  3918 ; INPUTS:
                     0EF0  3919 ;       R3 => CRB of the intelligent disk controller
                     0EF0  3920 ;
                     0EF0  3921 ; OUTPUTS:
                     0EF0  3922 ;       Registers R0 through R5 are all possibly modified.
                     0EF0  3923 ;
                     0EF0  3924
                     0EF0  3925 TU$TMR:
                     0EF0  3926          SETIPL  #IPL$_SCS                   ; After wakeup lower IPL.
51   10 A3   D0     0EF3  3927          MOVL    CRB$L_AUXSTRUC(R3),R1       ; R1 => CDDB.
                     0EF7  3928
                     0EF7  3929          ASSUME  CDDB$L_CDRPQFL  EQ  0
   51   61   D1     0EF7  3930          CMPL    (R1),R1                     ; If =, then list of CDRP's is empty
        21   13     0EFA  3931          BEQL    20$                         ; EQL means empty list of CDRP's,
                     0EFC  3932                                              ;  which implies we are in State #2.
   50   61   D0     0EFC  3933          MOVL    (R1),R0                     ; R0 => CDRP associated with "oldest"
                     0EFF  3934                                              ;  outstanding MSCP command.
                     0EFF  3935
     20 A0   D1     0EFF  3936          CMPL    CDRP$L_RSPID(R0),-          ; Compare RSPID of oldest request to
     2C A1          0F02  3937                  CDDB$L_OLDRSPID(R1)         ;  that of request current at time of
                     0F04  3938                                              ;  previous invocation of TU$TMR.
        1C   13     0F04  3939          BEQL    30$                         ; EQL implies State #3, i.e. current
                     0F06  3940                                              ;  oldest has been around for awhile.
                     0F06  3941
     20 A0   D0     0F06  3942          MOVL    CDRP$L_RSPID(R0),-          ; State #1, we have a NEW oldest request
     2C A1          0F09  3943                  CDDB$L_OLDRSPID(R1)         ;  so record its PSPID in CDDB field.
30 A1    01   CE     0F0B  3944          MNEGL   #1,CDDB$L_OLDCMDSTS(R1) ; And initialize its associated status.
                     0F0F  3945 10$:
7E   2A A1   3C     0F0F  3946          MOVZWL  CDDB$W_CNTRLTMO(R1),-(SP); Pickup controller delta.
        8E   C1     0F13  3947          ADDL3   (SP)+,-                     ; Calculate delta time for next
00000000'GF         0F15  3948                  G^EXE$GL_ABSTIM,-           ;  periodic invocation of TU$TMR.
     18 A3          0F1A  3949                  CRB$L_DUETIME(R3)
                05   0F1C  3950          RSB                                 ; And return to caller.
                     0F1D  3951
                     0F1D  3952 20$:                                         ; If we are here, there are NO outstand-
                     0F1D  3953                                              ;  ing requests in the controller since
                     0F1D  3954                                              ;  CDRP list is empty.
        50   D4     0F1D  3955          CLRL    R0                          ; R0 flagged to indicate State #2.
     2C A1   D4     0F1F  3956          CLRL    CDDB$L_OLDRSPID(R1)         ; Set to impossible value to prevent
                     0F22  3957                                              ;  inadvertent comparison error.
                     0F22  3958
                     0F22  3959 30$:                                         ; Common State #2, State #3 code path.
                     0F22  3960                                              ; If here, for sure we will be issuing
                     0F22  3961                                              ;  an immediate command to the controller.
                     0F22  3962                                              ;  If we are in State #2, it will be a
                     0F22  3963                                              ;  "GET UNIT STATUS" (NOP) command but
                     0F22  3964                                              ;  if we are in State #3, it will be
                     0F22  3965                                              ;  a "GET COMMAND STATUS" command.  For
                     0F22  3966                                              ;  either case we begin the common setup.
                     0F22  3967
                     0F22  3968
54   14 A1   D0     0F22  3969          MOVL    CDDB$L_PDT(R1),R4           ; Setup for SEND_MSG_BUF, R4=>PDT.
```

```
                55   00D0 C1   9E   0F26  3970          MOVAB   CDDB$A_PRMCDRP(R1),R5   ; R5 => CDRP appended to CDDB.
                          01   E3   0F2B  3971          BBCS    #CDDB$V_IMPEND,-        ; Branch if an immediate command is NOT
                     03 12 A1        0F2D  3972                  CDDB$W_STATUS(R1),40$   ;  pending. Also set bit to show that
                                     0F30  3973                                          ;  one WILL be pending momentarily.
                         FE18   31   0F30  3974          BRW     TU$RE_SYNCH             ; Bit set implies that an immediate
                                     0F33  3975                                          ;  "GET STATUS" type command has not
                                     0F33  3976                                          ;  completed in the timeout interval.
                                     0F33  3977                                          ;  So we goto resynchronization logic.
                                     0F33  3978
                                     0F33  3979 40$:
                          7E   50    7D   0F33  3980          MOVQ    R0, -(SP)           ; Save valuable registers.
                                     0F36  3981          INIT_MSCP_MSG                   ; Initalize buffer for MSCP message.
                          50   8E    7D   0F39  3982          MOVQ    (SP)+, R0           ; Restore valuable registers.
                                     0F3C  3983
                               D1    10   0F3C  3984          BSBB    10$                 ; Establish due time so as to be able
                                     0F3E  3985                                          ;  to timeout Immediate command.
                               50    D5   0F3E  3986          TSTL    R0                  ; Test for State #2 or State #3.
                               09    12   0F40  3987          BNEQ    50$                 ; NEQ implies State #3. Branch to handle it.
                                     0F42  3988
                                     0F42  3989 ; State #2 specific code.
                                     0F42  3990 ; Here we prepare the MSCP packet for the "GET UNIT STATUS" command for
                                     0F42  3991 ;       unit #0, which is an effective NOP command.  This is done to
                                     0F42  3992 ;       maintain minimum activity so that the controller will not time
                                     0F42  3993 ;       out the host (i.e. us). NOTE that since the MSCP buffer has been
                                     0F42  3994 ;       cleared above, there is no need to specify unit #0 in the command
                                     0F42  3995 ;       buffer.
                                     0F42  3996 ;
                                     0F42  3997
                               03    90   0F42  3998          MOVB    #MSCP$K_OP_GTUNT,-      ; Move in "GET UNIT STATUS" opcode.
                          08 A2          0F44  3999                  MSCP$B_OPCODE(R2)
                                     0F46  4000
                                     0F46  4001          SEND_MSCP_MSG DRIVER            ; Here we call to send the MSCP packet
                                     0F49  4002                                          ;  to the intelligent disk controller.
                                     0F49  4003
                                     0F49  4004                                          ; Return is experienced here after
                                     0F49  4005                                          ;  receipt of the END PACKET correspond-
                                     0F49  4006                                          ;  ing to the MSCP NOP sent above.  We
                                     0F49  4007                                          ;  regain control due to a callback
                                     0F49  4008                                          ;  from our own INPUT DISPATCHER
                                     0F49  4009                                          ;  ROUTINE.  Passed to us at this call-
                                     0F49  4010                                          ;  back are R2 => END PACKET, R3 => CRB,
                                     0F49  4011                                          ;  R4 => PDT and R5 => CDRP.
                                     0F49  4012                                          ; All we want to do is recycle the
                                     0F49  4013                                          ;  END PACKET for use as our next MSCP
                                     0F49  4014                                          ;  packet and recycle the RSPID.
                                     0F49  4015                                          ;  To do this we branch to common code.
                               35    11   0F49  4016          BRB     70$
                                     0F4B  4017
                                     0F4B  4018 50$:
                                     0F4B  4019
                                     0F4B  4020 ; State #3 specific code.
                                     0F4B  4021 ; Here we prepare the MSCP packet for a "GET COMMAND STATUS" command.
                                     0F4B  4022
                50   BC A0    D0   0F4B  4023          MOVL    CDRP$L_UCB(R0),R0       ; R0 => UCB for oldest outstanding request.
                                     0F4F  4024
                     00D4 C0    B0   0F4F  4025          MOVW    UCB$W_MSCPUNIT(R0),-    ; Setup UNIT field.
                       04 A2          0F53  4026                  MSCP$Q_UNIT(R2)
```

```
                     02   90   0F55  4027          MOVB    #MSCP$K_OP_GTCMD,-      ; Setup OPCODE field.
                 08  A2        0F57  4028                  MSCP$B_OPCODE(R2)
                                0F59  4029
                 2C  A1   D0    0F59  4030          MOVL    CDDB$L_OLDRSPID(R1),-   ; Setup OUTSTANDING COMMAND REFERENCE
                 0C  A2         0F5C  4031                  MSCP$L_OUT_REF(R2)      ;  field.
                                0F5E  4032
                                0F5E  4033          SEND_MSCP_MSG DRIVER            ; Here we call to send the MSCP packet
                                0F61  4034                                          ;  to the intelligent disk controller.
                                0F61  4035
                                0F61  4036                                          ; We experience return here upon receipt
                                0F61  4037                                          ;  of the END PACKET for the above "GET
                                0F61  4038                                          ;  COMMAND STATUS" command. We must make
                                0F61  4039                                          ;  sure that progress has indeed been
                                0F61  4040                                          ;  made on the outstanding command.  We
                                0F61  4041                                          ;  therefore compare the outstanding
                                0F61  4042                                          ;  command status returned in the END
                                0F61  4043                                          ;  PACKET to the previous value in CDDB
                                0F61  4044                                          ;  field CDDB$L_OLDCMDSTS.
                                0F61  4045                                          ; Here R2=>END PACKET, R3=>CRB, R4=>PDT
                                0F61  4046                                          ;  and R5=>CDRP.
                                0F61  4047
           51   10  A3   D0    0F61  4048          MOVL    CRB$L_AUXSTRUC(R3),R1   ; R1 => CDDB.
                10  A2   D1    0F65  4049          CMPL    MSCP$L_CMD_STS(R2),-    ; Compare received outstanding command
                30  A1         0F68  4050                  CDDB$L_OLDCMDSTS(R1)    ;  status to previous value.
                   0F   1F     0F6A  4051          BLSSU   60$                     ; LSSU implies progress made so branch.
                   0A   12     0F6C  4052          BNEQ    55$                     ; If not equal, progress went the
                                0F6E  4053                                          ;  wrong direction; a sure sign of
                                0F6E  4054                                          ;  an insane controller.
     10  A2   FFFFFFFF  8F    D1    0F6E  4055          CMPL    #-1, MSCP$L_CMD_STS(R2) ; If equal to last time, is this the
                                0F76  4056                                          ;  multi-host busy somewhere else value?
                   03   13     0F76  4057          BEQL    60$                     ; Branch if it is busy somewhere else.
                 FDD0   31     0F78  4058  55$:    BRW     TU$RE_SYNCH             ; Anything else, implies no progress
                                0F7B  4059                                          ;  has been made.  So we goto
                                0F7B  4060                                          ;  re-synchronize with the intelligent
                                0F7B  4061                                          ;  disk controller and re-issue all
                                0F7B  4062                                          ;  outstanding commands.
                                0F7B  4063
                                0F7B  4064  60$:
                10  A2   D0    0F7B  4065          MOVL    MSCP$L_CMD_STS(R2),-    ; Remember this received outstanding
                30  A1         0F7E  4066                  CDDB$L_OLDCMDSTS(R1)    ;  command status for next time.
                                0F80  4067
                                0F80  4068  70$:                                   ; States #2 and #3 code paths merge here.
                                0F80  4069          RECYCH_MSG_BUF          ; Recycle END PACKET.
                                0F83  4070          RECYCL_RSPID                    ; Likewise the RSPID.
                                0F89  4071
           51   10  A3   D0    0F89  4072          MOVL    CRB$L_AUXSTRUC(R3),R1   ; R1 => CDDB.
                   02   AA     0F8D  4073          BICW    #CDDB$M_IMPEND,-        ; Indicate that immediate command is
                12  A1         0F8F  4074                  CDDB$W_STATUS(R1)       ;  no longer pending.
                 FD6C'  31     0F91  4075          BRW     DUTU$DODAP              ; Continue by doing DAP processing.
```

```
                     0F94  4077          .SBTTL  TU$IDR - Class Driver Input Dispatch Routine
                     0F94  4078
                     0F94  4079 ;+
                     0F94  4080 ; TU$IDR - Class Driver Input Dispatching Routine.  This routine is to
                     0F94  4081 ;       the class driver what the Interrupt Service Routine is to a
                     0F94  4082 ;       conventional driver.  We are called here by the Port Driver
                     0F94  4083 ;       and we are passed the address of an END PACKET or an ATTENTION
                     0F94  4084 ;       MESSAGE buffer.  By testing a bit in the ENDCODE field of the
                     0F94  4085 ;       received buffer we determine which of the two has been received.
                     0F94  4086 ;       For ATTENTION MESSAGES we immediately branch to ATTN_MSG.
                     0F94  4087 ;
                     0F94  4088 ;       For END PACKETs we first determine if the END PACKET is still of
                     0F94  4089 ;       interest.  This is done by testing whether the COMMAND REFERENCE
                     0F94  4090 ;       NUMBER returned in the END PACKET, interpreted as a RSPID, is
                     0F94  4091 ;       still valid.  If not, we merely deallocate the END PACKET and
                     0F94  4092 ;       return to our caller in the Port Driver.
                     0F94  4093 ;
                     0F94  4094 ;       If the END PACKET is still of interest then before dispatching
                     0F94  4095 ;       to the code that originally issued the MSCP command for which we
                     0F94  4096 ;       have just received the END PACKET, we first remove the
                     0F94  4097 ;       CDRP associated with the command from the list of active CDRP's
                     0F94  4098 ;       defined by the listhead located at CDDB$L_CDRPQFL.
                     0F94  4099 ;
                     0F94  4100 ; INPUTS:
                     0F94  4101 ;       R1 =  Message Length
                     0F94  4102 ;       R2 => END PACKET or ATTENTION MESSAGE BUFFER
                     0F94  4103 ;       R3 => Connection Data Block
                     0F94  4104 ;-
                     0F94  4105
                     0F94  4106 TU$IDR:
          07   E1    0F94  4107          BBC     #MSCP$V_OP_END,-        ; Is this an ATTENTION MESSAGE
        08 A2         0F96  4108                  MSCP$B_OPCODE(R2),-     ;  or an END PACKET;
           4A         0F98  4109                  ATTN_MSG                ;  bit clear implies ATTENTION.
                     0F99  4110
                     0F99  4111
                     0F99  4112 ;
                     0F99  4113 ; Process command END MESSAGES
                     0F99  4114 ;
                     0F99  4115
          51   DD    0F99  4116          PUSHL   R1                      ; Save message size.
     55   62   D0    0F9B  4117          MOVL    MSCP$L_CMD_REF(R2), R5  ; Get RSPID from end message.
                     0F9E  4118          FIND_RSPID_RDTE                 ; Lookup RDTE for RSPID.
          51 8ED0    0FA4  4119          POPL    R1                      ; Restore message size.
        6B 50   E9    0FA7  4120          BLBC    R0, FINISHED_WITH_MESSAGE ; Branch if error in RSPID.
     55   65   D0    0FAA  4121          MOVL    RD$L_CDRP(R5),R5        ; R5 => CDRP.
50     24 A5   D0    0FAD  4122          MOVL    CDRP$L_CDT(R5),R0       ; R0 => CDT.
50     5C A0   D0    0FB1  4123          MOVL    CDT$L_AUXSTRUC(R0),R0   ; R0 => CDDB.
        2C A0   D1    0FB5  4124          CMPL    CDDB$L_OLDRSPID(R0),-   ; See if oldest outstanding command has
           62         0FB8  4125                  MSCP$L_CMD_REF(R2)      ;  this Command Reference Number.
           03   12    0FB9  4126          BNEQ    20$                     ; If not, branch around.
        2C A0   D4    0FBB  4127          CLRL    CDDB$L_OLDRSPID(R0)     ; Prevent inadvertent timeouts due to
                     0FBE  4128                                          ;  reuse of RSPID in error situations.
                     0FBE  4129 20$:     ASSUME  MSCP$K_LEN LT 32767
46 A5   51   B0    0FBE  4130          MOVW    R1, CDRP$W_ENDMSGSIZ(R5); Save length of incomming packet.
1C A5   52   D0    0FC2  4131          MOVL    R2, CDRP$L_MSG_BUF(R5)  ; Save address of incomming packet.
                     0FC6  4132
     55   65   0F   0FC6  4133          REMQUE  (R5),R5                 ; Remove R5=>CDRP from list.
```

```
                     0FC9  4134          ASSUME  CDRP$V_CAND EQ 0
          0C 40 A5   E8   0FC9  4135          BLBS    CDRP$L_DUTUFLAGS(R5), -  ; Has request been canceled?
                     0FCD  4136                  30$                      ;  If so, do cancel completion work.
    0C CA A5    07   E0   0FCD  4137  23$:      BBS     #IRP$V_DIAGBUF, -        ; Branch out of line if a diagnostic
                     0FD2  4138                  CDRP$W_STS(R5), 50$      ;  buffer was supplied.
                     0FD2  4139
       53   10 A5   7D   0FD2  4140  25$:      MOVQ    CDRP$L_FR3(R5), R3       ; Restore fork registers, R3 & R4.
            0C B5   17   0FD6  4141          JMP     @CDRP$L_FPC(R5)          ; Dispatch to issuer of MSCP command
                     0FD9  4142                                           ;  who will return to our caller.
                     0FD9  4143
             F024'  30   0FD9  4144  30$:      BSBW    DUTU$TEST_CANCEL_DONE    ; If this request completes a cancel
                     0FDC  4145                                           ; operation, cleanup that operation.
               EF   11   0FDC  4146          BRB     23$                      ; Branch back to normal flow.
                     0FDE  4147
             F01F'  30   0FDE  4148  50$:      BSBW    DUTU$DUMP_ENDMESSAGE     ; If diagnostic buffer, record MSCP
                     0FE1  4149                                           ; end message sent in the buffer.
               EF   11   0FE1  4150          BRB     25$                      ; Branch back to normal flow.
                     0FE3  4151
                     0FE3  4152
                     0FE3  4153
                     0FE3  4154 ;
                     0FE3  4155 ; Process ATTENTION MESSAGES
                     0FE3  4156 ;
                     0FE3  4157
                     0FE3  4158 ATTN_MSG:
               1E   BB   0FE3  4159          PUSHR   #^M<R1,R2,R3,R4>         ; Save vital registers.
       53   5C A3   D0   0FE5  4160          MOVL    CDT$L_AUXSTRUC(R3), R3   ; Get CDDB address.
            13'AF   9F   0FE9  4161          PUSHAB  B^EXIT_ATTN_MSG          ; Make DISPATCH look like a BSBx.
                     0FEC  4162          DISPATCH -                       ; Dispatch to attention message
                     0FEC  4163                  MSCP$B_OPCODE(R2), -     ;  specific processing:
                     0FEC  4164                  type=B, prefix=MSCP$K_OP_, < -
                     0FEC  4165                  <AVATN, UNIT_AVAILABLE_ATTN>, -
                     0FEC  4166                  <DUPUN, DUPLICATE_UNIT_ATTN>, -
                     0FEC  4167                  <ACPTH, ACCESS_PATH_ATTN>, -
                     0FEC  4168                  >
                     0FF8  4169 INV_ATTN_MSG:                             ; Process invalid ATTENTION MESSAGE.
               8E   D5   0FF8  4170          TSTL    (SP)+                    ; Pop "return" address.
          50   0A   3C   0FFA  4171          MOVZWL  #EMB$C_INVATT, R0        ; Invalid attention message type.
00000000'GF   16   0FFD  4172          JSB     G^ERL$LOG_TMSCP          ; Log incorrect TAPE MSCP message.
               1E   BA   1003  4173          POPR    #^M<R1,R2,R3,R4>         ; Restore vital registers.
                     1005  4174          DEALLOC_MSG_BUF_REG              ; Deallocate ATTN MSG buffer.
       53   5C A3   D0   1008  4175          MOVL    CDT$L_AUXSTRUC(R3), R3   ; Get CDDB again.
       53   18 A3   D0   100C  4176          MOVL    CDDB$L_CRB(R3), R3       ; From that get the CRB address.
             FD38   31   1010  4177          BRW     TU$RE_SYNCH              ; Re-synchronize with controller.
                     1013  4178
                     1013  4179 EXIT_ATTN_MSG:
               1E   BA   1013  4180          POPR    #^M<R1,R2,R3,R4>         ; Restore vital registers.
                     1015  4181 FINISHED_WITH_MESSAGE:
                     1015  4182          DEALLOC_MSG_BUF_REG              ; Deallocate ATTN MSG buffer.
                  05 1018  4183          RSB                              ; Return to SCS caller.
```

```
                     1019  4185          .SBTTL Attention Message Processing
                     1019  4186          .SBTTL   - Process Unit Available Attention Message
                     1019  4187
                     1019  4188 ;++
                     1019  4189 ;
                     1019  4190 ; Functional Description:
                     1019  4191 ;
                     1019  4192 ;       This routine processes unit available attention messages.  If the
                     1019  4193 ;       available unit is already known in the I/O database, no action is
                     1019  4194 ;       taken.  If the available unit represents a second path to an already
                     1019  4195 ;       known unit, the I/O database is altered to show the alternate path
                     1019  4196 ;       availability.  If the available unit represents a totally new device,
                     1019  4197 ;       it is added to the I/O database.
                     1019  4198 ;
                     1019  4199 ; Inputs:
                     1019  4200 ;
                     1019  4201 ;       R1      attention message size
                     1019  4202 ;       R2      attention message address
                     1019  4203 ;       R3      CDDB address
                     1019  4204 ;
                     1019  4205 ; Outputs:
                     1019  4206 ;
                     1019  4207 ;       R0 - R5 destroyed
                     1019  4208 ;       All other registers preserved
                     1019  4209 ;--
                     1019  4210
                     1019  4211 UNIT_AVAILABLE_ATTN:
                     1019  4212
03 12 A3   05   E0   1019  4213          BBS     #CDDB$V_POLLING, -      ; Is a poll for units in progress?
                     101E  4214                  CDDB$W_STATUS(R3), 90$  ; Branch if poll for units active.
        EFDF'   30   101E  4215          BSBW    DUTU$NEW_UNIT           ; Process possible new unit.
                     1021  4216          .IF     DEFINED TU_TRACE
                     1021  4217          MOVL    R2, R5                  ; Copy UCB address.
                     1021  4218          BSBW    TRACE_INIT              ; Initialize IRP trace table.
                     1021  4219          .ENDC
                05   1021  4220 90$:     RSB
```

```
                     1022  4222          .SBTTL   - Process Duplicate Unit Attention Message
                     1022  4223
                     1022  4224 ;++
                     1022  4225 ;
                     1022  4226 ; Functional Description:
                     1022  4227 ;
                     1022  4228 ;       This routine processes duplicate unit attention messages.
                     1022  4229 ;       Notification of the condition is sent to the operator's console and
                     1022  4230 ;       an entry is made in the error log.  If the unit described in the
                     1022  4231 ;       message cannot be found, an invalid MSCP message error log entry is
                     1022  4232 ;       generated.
                     1022  4233 ;
                     1022  4234 ; Inputs:
                     1022  4235 ;
                     1022  4236 ;       R1      attention message size
                     1022  4237 ;       R2      attention message address
                     1022  4238 ;       R3      CDDB address
                     1022  4239 ;
                     1022  4240 ; Outputs:
                     1022  4241 ;
                     1022  4242 ;       R0 - R5 destroyed
                     1022  4243 ;       All other registers preserved
                     1022  4244 ;--
                     1022  4245
                     1022  4246          .ENABLE LSB
                     1022  4247
                     1022  4248 DUPLICATE_UNIT_ATTN:
                     1022  4249
        EFDB'   30   1022  4250          BSBW    DUTU$LOOKUP_UCB         ; Locate UCB for this message.
53       50     D0   1025  4251          MOVL    R0, R3                  ; Setup UCB address.
            0C  13   1028  4252          BEQL    90$                     ; If no UCB found, ignore the message.
        EFD3'   30   102A  4253          BSBW    DUTU$SEND_DUPLICATE_UNIT; Send message to operator.
50       06     3C   102D  4254          MOVZWL  #EMB$C_DUPUN, R0        ; Setup duplicate unit error log code.
                     1030  4255
                     1030  4256 LOG_ATTENTION_MESSAGE:
00000000'EF     16   1030  4257          JSB     ERL$LOGMESSAGE          ; Error log attention message.
                05   1036  4258 90$:     RSB
                     1037  4259
                     1037  4260          .DISABLE LSB
```

```
                     1037  4262          .SBTTL   - Process Access Path Attention Message
                     1037  4263
                     1037  4264 ;++
                     1037  4265 ;
                     1037  4266 ; Functional Description:
                     1037  4267 ;
                     1037  4268 ;       This routine processes access path attention messages.  If the access
                     1037  4269 ;       path represents a second path to an already known unit, the I/O
                     1037  4270 ;       database is altered to show the alternate path availability, and an
                     1037  4271 ;       entry is made in the error log indicating receipt of the message.
                     1037  4272 ;       If the unit described in the message cannot be found, an invalid MSCP
                     1037  4273 ;       message error log entry is generated.
                     1037  4274 ;
                     1037  4275 ; Inputs:
                     1037  4276 ;
                     1037  4277 ;       R1      attention message size
                     1037  4278 ;       R2      attention message address
                     1037  4279 ;       R3      CDDB address
                     1037  4280 ;
                     1037  4281 ; Outputs:
                     1037  4282 ;
                     1037  4283 ;       R0 - R5 destroyed
                     1037  4284 ;       All other registers preserved
                     1037  4285 ;--
                     1037  4286
                     1037  4287 ACCESS_PATH_ATTN:
                     1037  4288
    EFC6'    30      1037  4289          BSBW    DUTU$SETUP_DUAL_PATH    ; Process possible dual path unit.
53   50    D0      103A  4290          MOVL    R0, R3                  ; Get UCB address.
     06    13      103D  4291          BEQL    90$                     ; If no UCB found, ignore the message.
           05      103F  4292          RSB                             ; Return w/o logging message, but
                   1040  4293                                          ; leave message logging code in place
                   1040  4294                                          ; just in case its needed.
50   08    9A      1040  4295          MOVZBL  #EMB$C_ACPTH, R0        ; Setup ERL$LOGMESSAGE code.
     EB    11      1043  4296          BRB     LOG_ATTENTION_MESSAGE   ; Join common log message path.
           05      1045  4297 90$:     RSB                             ; If no UCB, exit.
```

```
                     1046  4299          .SBTTL  TU$DGDR - Data Gram Dispatch Routine
                     1046  4300 ;
                     1046  4301 ; Inputs:
                     1046  4302 ;
                     1046  4303 ;       R1 =  length of datagram
                     1046  4304 ;       R2 => datagram
                     1046  4305 ;       R3 => CDT
                     1046  4306 ;       R4 => PDT
                     1046  4307
                     1046  4308 TU$DGDR:
                     1046  4309
      50   5C A3   D0  1046  4310          MOVL    CDT$L_AUXSTRUC(R3),R0   ; R0 => CDDB
           55   53   D0  104A  4311          MOVL    R3,R5                   ; Save pointer to CDT.
50    0000007C 8F   C3  104D  4312          SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB address in R3.
                     1054  4313                   -CDDB$L_UCBCHAIN>, -
                 53        1054  4314                   R0, R3
                     1055  4315
      53   00C4 C3   D0  1055  4316 10$:    MOVL    UCB$L_CDDB_LINK(R3), R3 ; Chain to next UCB (if any).
                 11   13  105A  4317          BEQL    20$                     ; No more UCBs.
           00D4 C3   B1  105C  4318          CMPW    UCB$W_MSCPUNIT(R3),-    ; See if datagram (error log packet)
             04 A2       1060  4319                   MSCP$W_UNIT(R2)         ;  for this unit.
                 F1   12  1062  4320          BNEQ    10$                     ; If not, branch abck to try next unit.
           50   02   3C  1064  4321          MOVZWL  #EMB$C_TM,R0            ; Put type of message into R0.
 00000000'GF    16  1067  4322          JSB     G^ERL$LOGMESSAGE        ; And call to log message.
                     106D  4323 20$:
           53   55   D0  106D  4324          MOVL    R5,R3                   ; Restore R3 => CDT.
      52   00B8 C4   C2  1070  4325          SUBL    PDT$L_DGOVRHD(R4),R2    ; R2 => SCS header of datagram.
                     1075  4326          QUEUE_DG_BUF                    ; Requeue datagram buffer.
                 05  1078  4327          RSB                             ; Return to port.
```

```
                     1079  4329          .SBTTL  INVALID_STS
                     1079  4330
                     1079  4331 ;+
                     1079  4332 ; We come here if we get an invalid MSCP status.  We log the MSCP message
                     1079  4333 ; and then RE-SYNCH the controller.
                     1079  4334 ;
                     1079  4335 ; Inputs:
                     1079  4336 ;       R2 => MSCP packet
                     1079  4337 ;       R3 => UCB
                     1079  4338 ;       R4 => PDT
                     1079  4339 ;       R5 => CDRP
                     1079  4340 ;       CDRP$W_ENDMSGSIZ(R5) => length of MSCP packet with invalid status
                     1079  4341 ;
                     1079  4342
                     1079  4343 INVALID_STS:
                     1079  4344
     50   09   3C   1079  4345          MOVZWL  #EMB$C_INVSTS,R0          ; Indicate type of record to log.
   51   46 A5   3C   107C  4346          MOVZWL  CDRP$W_ENDMSGSIZ(R5), R1; Pickup length of faulty packet.
53   00BC C3   D0   1080  4347          MOVL    UCB$L_CDDB(R3),R3         ; R3 => CDDB for logging error.
 00000000'GF   16   1085  4348          JSB     G^ERL$LOG_TMSCP           ; Log tape MSCP error.
       EF72'   30   108B  4349          BSBW    DUTU$INSERT_RESTARTQ      ; Queue CDRP for retry.
  53   18 A3   D0   108E  4350          MOVL    CDDB$L_CRB(R3),R3         ; R3 => CRB for re-SYNCH.
       FCB6   31   1092  4351          BRW     TU$RE_SYNCH               ; Zap controller.
```

```
1095  4353          .SBTTL  TU_UNSOLNT
1095  4354
1095  4355 TU_UNSOLNT:
1095  4356          BUG_CHECK       TAPECLASS,FATAL
1099  4357
1099  4358
1099  4359          .IIF    DEFINED TU_TRACE, .PAGE
1099  4360          .IF     DEFINED TU_TRACE
1099  4361          .SBTTL  IRP Tracing Routines
1099  4362          .SBTTL  - TRACE_INIT - Initialize trace table
1099  4363 ;++
1099  4364 ;
1099  4365 ; TRACE_INIT - Initialize trace table
1099  4366 ;
1099  4367 ; Functional Description:
1099  4368 ;
1099  4369 ;       If the trace table is not initialized, initialize it.
1099  4370 ;
1099  4371 ; Inputs:
1099  4372 ;
1099  4373 ;       R5      UCB address.
1099  4374 ;
1099  4375 ; Implicit Inputs:
1099  4376 ;
1099  4377 ;       UCB$W_DEVSTS(R5)        UCB$V_TU_TRACEACT set if the trace table is
1099  4378 ;                               initialized
1099  4379 ;
1099  4380 ; Outputs:
1099  4381 ;
1099  4382 ;       All registers preserved.
1099  4383 ;
1099  4384 ; Implicit Outputs:
1099  4385 ;
1099  4386 ;       UCB$W_DEVSTS(R5)        UCB$V_TU_TRACEACT is set if the trace table is
1099  4387 ;                               successfully initialized
1099  4388 ;       UCB$L_TRACEBEG(R5)      address of first IRP trace slot
1099  4389 ;       UCB$L_TRACEPTR(R5)      address of first free IRP trace slot
1099  4390 ;       UCB$L_TRACEND(R5)       address of first byte after IRP trace slots
1099  4391 ;--
1099  4392
1099  4393 TRACE_SLOTS = 50                            ; Number of trace slots
1099  4394 TRACE_SIZE = 96                             ; Size of a trace slot
1099  4395 TRACE_TBLSIZ = TRACE_SLOTS * TRACE_SIZE ; Size of the trace table
1099  4396
1099  4397          ASSUME  IRP$L_ARB+8 LE TRACE_SIZE
1099  4398          ASSUME  <TRACE_SIZE & ^X1F> EQ 0
1099  4399
1099  4400 IRP$L_TU_TRCPTR = IRP$K_CD_LEN             ; Define a place to hold pointer to
1099  4401 CDRP$C_TO_TRCPTR = CDRP$K_CD_LEN           ; trace slot
1099  4402
1099  4403          ASSUME  IRP$L_TU_TRCPTR+4 LE IRP$K_LENGTH
1099  4404          ASSUME  CDRP$C_TO_TRCPTR-CDRP$L_IOQFL EQ IRP$L_TU_TRCPTR
1099  4405
1099  4406 TRACE_INIT:
1099  4407
1099  4408          BBS     #UCB$V_TU_TRACEACT, -   ; Branch if tracing is already
1099  4409                  UCB$W_DEVSTS(R5), 90$   ; initialized.
```

```
1099  4410          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers.
1099  4411          MOVZWL  #<TRACE_TBLSIZ+16>, R1  ; Get size of the trace table w/ header.
1099  4412          JSB     G^EXE$ALONONPAGED       ; Attempt to allocate pool.
1099  4413          BLBC    R0, 80$                 ; Branch if allocation failed.
1099  4414          CLRQ    (R2)+                   ; Initialize trace table header for SDA.
1099  4415          MOVW    R1, (R2)+               ; Save size.
1099  4416          MOVW    #DYN$C_CLASSDRV, (R2)+  ; Type.
1099  4417          CLRL    (R2)+                   ; Round header upto 16 byte boundary.
1099  4418          MOVL    R2, UCB$L_TRACEBEG(R5)  ; Save pointer to base of trace table.
1099  4419          MOVL    R2, UCB$L_TRACEPTR(R5)  ; Pointer to next area to use.
1099  4420          ADDL3   #TRACE_TBLSIZ, R2, -    ; Pointer to beyond end of trace table.
1099  4421                  UCB$L_TRACEND(R5)
1099  4422          BISW    #UCB$M_TU_TRACEACT, -   ; Indicate Trace table inited.
1099  4423                  UCB$W_DEVSTS(R5)
1099  4424          MOVC5   #0, (SP), #0, -         ; Zero trace table.
1099  4425                  #TRACE_TBLSIZ, (R2)
1099  4426
1099  4427 80$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers.
1099  4428 90$:    RSB                             ; Return
1099  4429          .PAGE
1099  4430          .SBTTL  - TRACE_IRP - Trace incomming IRP
1099  4431 ;++
1099  4432 ;
1099  4433 ; TRACE_IRP - Trace incomming IRP
1099  4434 ;
1099  4435 ; Functional Description:
1099  4436 ;
1099  4437 ;       Called as a part of start I/O processing, this routine allocates a new
1099  4438 ;       IRP trace slot and copies starting IRP contents into that slot.
1099  4439 ;
1099  4440 ;       IRP trace slots are 96 bytes long so that they line up nicely in
1099  4441 ;       a dump.
1099  4442 ;
1099  4443 ; Inputs:
1099  4444 ;
1099  4445 ;       R3      IRP address
1099  4446 ;       R5      UCB address
1099  4447 ;
1099  4448 ; Implicit Inputs:
1099  4449 ;
1099  4450 ;       UCB$W_DEVSTS(R5)        UCB$V_TU_TRACEACT set if IRP trace slots have
1099  4451 ;                               been allocated
1099  4452 ;       UCB$L_TRACEPTR(R5)      address of first free IRP trace slot
1099  4453 ;       UCB$L_TRACEND(R5)       address of first byte after IRP trace slots
1099  4454 ;       UCB$L_TRACEBEG(R5)      address of first IRP trace slot
1099  4455 ;
1099  4456 ; Outputs:
1099  4457 ;
1099  4458 ;       All registers preserved.
1099  4459 ;
1099  4460 ; Implicit Outputs:
1099  4461 ;
1099  4462 ;       UCB$L_TRACEPTR(R5)      updated
1099  4463 ;       IRP$L_TU_TRCPTR(R3)     Address of IRP trace slot (for TRACE_STATUS)
1099  4464 ;--
1099  4465
1099  4466 TRACE_IRP:
```

```
1099  4467
1099  4468          BBC     #UCB$V_TU_TRACEACT, -   ; If trace table not intialized,
1099  4469                  UCB$W_DEVSTS(R5), 20$   ; exit immediately.
1099  4470          MOVQ    R0, -(SP)               ; Save R0 and R1.
1099  4471          MOVL    R3, R0                  ; Get IRP to trace in R0.
1099  4472          MOVL    UCB$L_TRACEPTR(R5), R1  ; Get address of next free trace slot.
1099  4473          CMPL    UCB$L_TRACEND(R5), R1   ; Check for end of trace table.
1099  4474          BGTR    10$                     ; Branch if not overflowed trace tbl.
1099  4475          MOVL    UCB$L_TRACEBEG(R5), R1  ; Else, reset to base of trace table.
1099  4476 10$:     ADDL3   #TRACE_SIZE, R1, -      ; Setup next entry pointer.
1099  4477                  UCB$L_TRACEPTR(R5)
1099  4478
1099  4479          MOVL    R1, IRP$L_TU_TRCPTR(R3) ; Save trace slot addr at end of CDRP.
1099  4480          ASSUME  <TRACE_SIZE & 7> EQ 0
1099  4481          .REPEAT TRACE_SIZE / 8
1099  4482          MOVQ    (R0)+, (R1)+            ; Copy input IRP.
1099  4483          .ENDR
1099  4484          MOVL    IRP$L_TU_TRCPTR(R3), R1 ; Refresh R1 to trace slot beginning.
1099  4485          MOVL    R3, (R1)                ; Put IRP address in trace slot.
1099  4486          CLRL    4(R1)                   ; Clear field that will contain RSPID.
1099  4487          MNEGL   #1, IRP$L_ARB(R1)       ; Init field for I/O Status #1.
1099  4488          MNEGL   #1, IRP$L_ARB+4(R1)     ; Init field for I/O Status #2.
1099  4489
1099  4490          MOVQ    (SP)+,R0                ; Restore R0 and R1.
1099  4491 20$:     RSB
1099  4492          .PAGE
1099  4493          .SBTTL  - TRACE_STATUS - Trace final I/O request status
1099  4494 ;++
1099  4495 ;
1099  4496 ; TRACE_STATUS - Trace final I/O request status
1099  4497 ;
1099  4498 ; Functional Description:
1099  4499 ;
1099  4500 ;       Copy final I/O status and RSPID into trace slot.
1099  4501 ;
1099  4502 ; Inputs:
1099  4503 ;
1099  4504 ;       R0      I/O status first longword
1099  4505 ;       R3      UCB address
1099  4506 ;       R5      CDRP address
1099  4507 ;
1099  4508 ; Implicit Inputs:
1099  4509 ;
1099  4510 ;       UCB$W_DEVSTS(R3)        UCB$V_TU_TRACEACT set if IRP trace slots have
1099  4511 ;                               been allocated
1099  4512 ;       CDRP$L_TU_TRCPTR(R5)    Address of IRP trace slot
1099  4513 ;       UCB$L_DEVDEPEND(R3)     I/O status second longword
1099  4514 ;
1099  4515 ; Outputs:
1099  4516 ;
1099  4517 ;       All registers preserved.
1099  4518 ;
1099  4519 ; Implicit Outputs:
1099  4520 ;
1099  4521 ;       RSPID and final I/O status copies to IRP trace slot.
1099  4522 ;--
1099  4523
```

```
1099  4524 TRACE_STATUS:
1099  4525
1099  4526          BBC     #UCB$V_TU_TRACEACT, -   ; If trace table not initialized
1099  4527                  UCB$W_DEVSTS(R3), 30$   ; exit immediately.
1099  4528          PUSHL   R2                      ; Save register.
1099  4529          MOVL    CDRP$L_TU_TRCPTR(R5), R2; Get IRP trace slot address.
1099  4530          MOVL    CDRP$L_RSPID(R5), 4(R2) ; Save RSPID in trace.
1099  4531          MOVL    R0, IRP$L_ARB(R2)       ; Save I/O status.
1099  4532          MOVL    UCB$L_DEVDEPEND(R3), -  ;       "
1099  4533                  IRP$L_ARB+4(R2)
1099  4534          POPL    R2                      ; Restore register.
1099  4535 30$:     RSB                             ; Return to caller.
1099  4536
1099  4537          .ENDC
1099  4538
1099  4539          .END
```

```
$$$                             = 00000020 R      04
$$BASE                          = 00000040
$$BEGIN$$                       = 00000002
$$DISPL                         = 00000043
$$GENSW                         = 00000001
$$HIGH                          = 00000042
$$LIMIT                         = 00000002
$$LOW                           = 00000040
$$MEDIA$$                       = 69A9504E
$$MNSW                          = 00000001
$$MXSW                          = 00000001
$$N$$                           = 0000004E
$$OP                            = 00000002
$$S$$                           = 00000002
$$TEMP$$                        = FFFFFFF7
ACCESS_PATH_ATTN                  00001037 R      05
ACP$ACCESS                        ******** X    05
ACP$DEACCESS                      ******** X    05
ACP$MODIFY                        ******** X    05
ACP$MOUNT                         ******** X    05
ACP$READBLK                       ******** X    05
ACP$WRITEBLK                      ******** X    05
ALLOC_DELTA                     = 00000001
AT$_NOLL                        = 00000005
ATE_MSCPCODE                      00000002
ATE_OFFSET                        00000000
ATE_SSCODE                        00000003
ATTN_MSG                          00000FE3 R      05
AUTO_PACKACK                      0000048A R      05
AVAILABLE_ABORT                   0000085F R      05
AVAILABLE_CTRLERR                 0000085F R      05
AVAILABLE_DRVERR                  0000085F R      05
AVAILABLE_MEDOFL                  0000085F R      05
AVAILABLE_SEREX                   0000087E R      05
AVAILABLE_SUCC                    0000085F R      05
AVAIL_IVCMD                       00000857 R      05
AVAIL_IVCMD_END                   0000085D R      05
BRING_UNIT_ONLINE                 00000340 R      05
BUG$_TAPECLASS                    ******** X    05
CDDB$A_2PFKB                      00000174
CDDB$A_DAPCDRP                    00000194
CDDB$A_DAPIRP                     00000134
CDDB$A_PRMCDRP                    000000D0
CDDB$A_PRMIRP                     00000070
CDDB$B_CNTRLMDL                 = 00000026
CDDB$B_RETRYCNT                 = 00000038
CDDB$B_SYSTEMID                 = 0000000C
CDDB$K_LENGTH                   = 00000070
CDDB$L_ALLOCLS                  = 00000050
CDDB$L_CANCLQBL                   000000B4
CDDB$L_CANCLQFL                   000000B0
CDDB$L_CDRPQFL                  = 00000000
CDDB$L_CDT                        000000F4
CDDB$L_CRB                      = 00000018
CDDB$L_DAPCDRP                  = 00000054
CDDB$L_DAPCDT                     000001B8
CDDB$L_DAPUCB                     00000150
CDDB$L_DDB                      = 0000001C
CDDB$L_OLDCMDSTS                = 00000030
CDDB$L_OLDRSPID                 = 0000002C
CDDB$L_PDT                      = 00000014
CDDB$L_PRMUCB                     0000008C
CDDB$L_RSTRTCDRP                = 00000034
CDDB$L_RSTRTQFL                 = 0000003C
CDDB$L_SAVED_PC                 = 00000044
CDDB$L_UCBCHAIN                 = 00000048
CDDB$M_DAPBSY                   = 00000400
CDDB$M_IMPEND                   = 00000002
CDDB$M_INITING                  = 00000004
CDDB$M_NOCONN                   = 00000080
CDDB$M_RECONNECT                = 00000008
CDDB$M_RESYNCH                  = 00000010
CDDB$M_RSTRTWAIT                = 00000100
CDDB$M_SNGLSTRM                 = 00000001
CDDB$Q_CNTRLID                  = 00000020
CDDB$V_ALCLS_SET                = 00000006
CDDB$V_DAPBSY                   = 0000000A
CDDB$V_IMPEND                   = 00000001
CDDB$V_INITING                  = 00000002
CDDB$V_POLLING                  = 00000005
CDDB$V_RESYNCH                  = 00000004
CDDB$V_SNGLSTRM                 = 00000000
CDDB$W_CNTRLFLGS                = 00000028
CDDB$W_CNTRLTMO                 = 0000002A
CDDB$W_RSTRTCNT                 = 0000003A
CDDB$W_STATUS                   = 00000012
CDRP$B_CARCON                   = FFFFFFDC
CDRP$B_CD_TYPE                  = 0000000A
CDRP$B_EFN                      = FFFFFFC2
CDRP$B_FIPL                     = 0000000B
CDRP$B_IRP_TYPE                 = FFFFFFAA
CDRP$B_PRI                      = FFFFFFC3
CDRP$B_RMOD                     = FFFFFFAB
CDRP$L_ABCNT                    = FFFFFFE0
CDRP$L_ARB                      = FFFFFFF8
CDRP$L_AST                      = FFFFFFB0
CDRP$L_ASTPRM                   = FFFFFFB4
CDRP$L_BCNT                     = FFFFFFD2
CDRP$L_CDT                      = 00000024
CDRP$L_DIAGBUF                  = FFFFFFEC
CDRP$L_DUTUFLAGS                = 00000040
CDRP$L_EXTEND                   = FFFFFFF4
CDRP$L_FPC                      = 0000000C
CDRP$L_FR3                      = 00000010
CDRP$L_IOQBL                    = FFFFFFA4
CDRP$L_IOQFL                    = FFFFFFA0
CDRP$L_IOSB                     = FFFFFFC4
CDRP$L_IOST1                    = FFFFFFD8
CDRP$L_IOST2                    = FFFFFFDC
CDRP$L_JNL_SEQNO                = FFFFFFE8
CDRP$L_LBUFH_AD                 = 0000002C
CDRP$L_MEDIA                    = FFFFFFD8
CDRP$L_MSG_BUF                  = 0000001C
CDRP$L_OBCNT                    = FFFFFFE4
```

```
CDRP$L_PID                      = FFFFFFAC
CDRP$L_RSPID                    = 00000020
CDRP$L_RWCPTR                   = 00000028
CDRP$L_SEGVBN                   = FFFFFFE8
CDRP$L_SEQNUM                   = FFFFFFF0
CDRP$L_SVAPTE                   = FFFFFFCC
CDRP$L_TT_TERM                  = FFFFFFDC
CDRP$L_UCB                      = FFFFFFBC
CDRP$L_WIND                     = FFFFFFB8
CDRP$M_DENSCK                   = 00000020
CDRP$M_ERLIP                    = 00000004
CDRP$Q_NT_PRVMSK                = FFFFFFE0
CDRP$T_LBUFHNDL                 = 00000030
CDRP$V_CAND                     = 00000000
CDRP$V_DENSCK                   = 00000005
CDRP$V_ERLIP                    = 00000002
CDRP$V_IVCMD                    = 00000008
CDRP$W_ABCNT                    = FFFFFFE0
CDRP$W_BCNT                     = FFFFFFD2
CDRP$W_BOFF                     = FFFFFFD0
CDRP$W_CDRPSIZE                 = 00000008
CDRP$W_CHAN                     = FFFFFFC8
CDRP$W_ENDMSGSIZ                = 00000046
CDRP$W_FUNC                     = FFFFFFC0
CDRP$W_IRP_SIZE                 = FFFFFFA8
CDRP$W_OBCNT                    = FFFFFFE4
CDRP$W_STS                      = FFFFFFCA
CDT$L_AUXSTRUC                  = 0000005C
CDT$L_PB                        = 0000001C
CLASS_DRVR_NAME                   0000015B R      05
CLU$GL_ALLOCLS                    ******** X    05
CONNECT_DELTA                   = 0000000A
CRB$L_AUXSTRUC                  = 00000010
CRB$L_DUETIME                   = 00000018
CRB$L_INTD                      = 00000024
CRB$L_TOUTROUT                  = 0000001C
DC$_TAPE                        = 00000002
DDB$L_ACPD                      = 00000010
DDB$L_ALLOCLS                   = 0000003C
DDB$L_CONLINK                   = 00000038
DDB$L_DDT                       = 0000000C
DDB$L_UCB                       = 00000004
DEV$M_AVL                       = 00040000
DEV$M_CLU                       = 00000001
DEV$M_DIR                       = 00000008
DEV$M_ELG                       = 00400000
DEV$M_FOD                       = 00004000
DEV$M_IDV                       = 04000000
DEV$M_MSCP                      = 00000020
DEV$M_NNM                       = 00000200
DEV$M_ODV                       = 08000000
DEV$M_SDI                       = 00000010
DEV$M_SQD                       = 00000020
DEV$V_CDP                       = 00000003
DEV$V_FOR                       = 00000018
DEV$V_MNT                       = 00000013
DISCONNECT_REASON               = 00000001
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R      04
DPT$M_NOUNLOAD                  = 00000004
DPT$M_SCS                       = 00000008
DPT$REINITAB                      00000078 R      04
DPT$TAB                           00000000 R      04
DT$_TA78                        = 00000006
DT$_TA81                        = 00000009
DT$_TK50                        = 0000000A
DT$_TU78                        = 00000005
DT$_TU81                        = 00000008
DUPLICATE_UNIT_ATTN               00001022 R      05
DUTU$CANCEL                       ********    X   05
DUTU$CHECK_RWAITCNT               ********    X   05
DUTU$CREATE_CDDB                  ********    X   05
DUTU$DEALLOC_ALL                  ********    X   05
DUTU$DEALLOC_RSPID_MSG            ********    X   05
DUTU$DISCONNECT_CANCEL            ********    X   05
DUTU$DODAP                        ********    X   05
DUTU$DRAIN_CDDB_CDRPQ             ********    X   05
DUTU$DUMP_ENDMESSAGE              ********    X   05
DUTU$END                          ********    X   04
DUTU$GET_DEVTYPE                  ********    X   05
DUTU$INIT_CONN_UCB                ********    X   05
DUTU$INIT_MSCP_MSG                ********    X   05
DUTU$INIT_MSCP_MSG_UNIT           ********    X   05
DUTU$INSERT_RESTARTQ              ********    X   05
DUTU$INTR_ACTION_N                ********    X   05
DUTU$INTR_ACTION_XFER             ********    X   05
DUTU$KILL_THIS_THREAD             ********    X   05
DUTU$LOG_IVCMD                    ********    X   05
DUTU$LOOKUP_UCB                   ********    X   05
DUTU$L_CDDB_LISTHEAD              00000000
DUTU$NEW_UNIT                     ********    X   05
DUTU$POLL_FOR_UNITS               ********    X   05
DUTU$POST_CDRP                    ********    X   05
DUTU$RECONN_LOOKUP                ********    X   05
DUTU$RESET_MSCP_MSG               ********    X   05
DUTU$RESTORE_CREDIT               ********    X   05
DUTU$SEND_DRIVER_MSG              ********    X   05
DUTU$SEND_DUPLICATE_UNIT          ********    X   05
DUTU$SEND_MSCP_MSG                ********    X   05
DUTU$SETUP_DUAL_PATH              ********    X   05
DUTU$TEST_CANCEL_DONE             ********    X   05
DUTU$UNITINIT                     ********    X   05
DYN$C_CDRP                      = 00000039
DYN$C_CRB                       = 00000005
DYN$C_DDB                       = 00000006
DYN$C_DPT                       = 0000001E
DYN$C_ORB                       = 00000049
DYN$C_UCB                       = 00000010
EMB$C_ACPTH                     = 00000008
EMB$C_DUPUN                     = 00000006
EMB$C_INVATT                    = 0000000A
EMB$C_INVSTS                    = 00000009
EMB$C_TM                        = 00000002
```

```
END_PACKACK                     00000792 R      05
END_SINGLE_STREAM               00000EBB R      05
ERASEGAP_POST                   00000BF4 R      05
ERL$LOGMESSAGE                  ********    X   05
ERL$LOGSTATUS                   ********    X   05
ERL$LOG_TMSCP                   ********    X   05
EXE$FORK                        ********    X   05
EXE$GL_ABSTIM                   ********    X   05
EXE$GQ_SYSTIME                  ********    X   05
EXE$INSIOQ                      ********    X   05
EXE$ONEPARM                     ********    X   05
EXE$SETMODE                     ********    X   05
EXE$ZEROPARM                    ********    X   05
EXIT_ATTN_MSG                   00001013 R      05
FINISHED_WITH_MESSAGE           00001015 R      05
FKB$K_LENGTH                =   00000018
FUNCTAB_LEN                 =   00000088
FUNCTION_EXIT                   00000CC8 R      05
HOST_TIMEOUT                =   0000001E
HSTIMEOUT_ARRAY                 0000017B R      05
INI$BRK                         ********    X   05
INITIAL_CREDIT              =   0000000A
INITIAL_DG_COUNT            =   00000002
INIT_IMMED_DELTA            =   0000001E
INIT_TIMEOUT                    00000158 R      05
INVALID_STS                     00001079 R      05
INV_ATTN_MSG                    00000FF8 R      05
IO$V_CLSEREXCP              =   00000009
IO$V_DATACHECK              =   0000000E
IO$V_INHRETRY               =   0000000F
IO$V_NOWAIT                 =   00000007
IO$V_REVERSE                =   00000006
IO$_ACCESS                  =   00000032
IO$_ACPCONTROL              =   00000038
IO$_AVAILABLE               =   00000011
IO$_CREATE                  =   00000033
IO$_DEACCESS                =   00000034
IO$_DELETE                  =   00000035
IO$_DSE                     =   00000015
IO$_ERASETAPE               =   00000006
IO$_MODIFY                  =   00000036
IO$_MOUNT                   =   00000039
IO$_NOP                     =   00000000
IO$_PACKACK                 =   00000008
IO$_READLBLK                =   00000021
IO$_READPBLK                =   0000000C
IO$_READVBLK                =   00000031
IO$_RECAL                   =   00000003
IO$_REWIND                  =   00000024
IO$_REWINDOFF               =   00000022
IO$_SENSECHAR               =   0000001B
IO$_SENSEMODE               =   00000027
IO$_SETCHAR                 =   0000001A
IO$_SETMODE                 =   00000023
IO$_SKIPFILE                =   00000025
IO$_SKIPRECORD              =   00000026
IO$_SPACEFILE               =   00000002
IO$_SPACERECORD             =   00000009
IO$_UNLOAD                  =   00000001
IO$_VIRTUAL                 =   0000003F
IO$_WRITECHECK              =   0000000A
IO$_WRITELBLK               =   00000020
IO$_WRITEMARK               =   0000001C
IO$_WRITEOF                 =   00000028
IO$_WRITEPBLK               =   0000000B
IO$_WRITEVBLK               =   00000030
IOC$ALTREQCOM                   ********    X   05
IOC$GL_TU_CDDB                  ********    X   06
IOC$MNTVER                      ********    X   05
IOC$RETURN                      ********    X   05
IPL$_SCS                    =   00000008
IRP$B_CARCON                =   0000003C
IRP$B_EFN                   =   00000022
IRP$B_PRI                   =   00000023
IRP$B_RMOD                  =   0000000B
IRP$B_TYPE                  =   0000000A
IRP$K_LENGTH                =   000000C4
IRP$L_ABCNT                 =   00000040
IRP$L_ARB                   =   00000058
IRP$L_AST                   =   00000010
IRP$L_ASTPRM                =   00000014
IRP$L_BCNT                  =   00000032
IRP$L_CDT                   =   00000084
IRP$L_DIAGBUF               =   0000004C
IRP$L_EXTEND                =   00000054
IRP$L_FQFL                  =   00000060
IRP$L_IOQBL                 =   00000004
IRP$L_IOQFL                 =   00000000
IRP$L_IOSB                  =   00000024
IRP$L_IOST1                 =   00000038
IRP$L_IOST2                 =   0000003C
IRP$L_JNL_SEQNO             =   00000048
IRP$L_MEDIA                 =   00000038
IRP$L_OBCNT                 =   00000044
IRP$L_PID                   =   0000000C
IRP$L_SEGVBN                =   00000048
IRP$L_SEQNUM                =   00000050
IRP$L_SVAPTE                =   0000002C
IRP$L_TT_TERM               =   0000003C
IRP$L_UCB                   =   0000001C
IRP$L_WIND                  =   00000018
IRP$Q_NT_PRVMSK             =   00000040
IRP$S_FCODE                 =   00000006
IRP$V_DIAGBUF               =   00000007
IRP$V_FCODE                 =   00000000
IRP$V_PHYSIO                =   00000008
IRP$W_ABCNT                 =   00000040
IRP$W_BCNT                  =   00000032
IRP$W_BOFF                  =   00000030
IRP$W_CHAN                  =   00000028
IRP$W_FUNC                  =   00000020
IRP$W_OBCNT                 =   00000044
IRP$W_SIZE                  =   00000008
IRP$W_STS                   =   0000002A
```

```
LOCAL_DEVICE                    0000056D R      05
LOG_ATTENTION_MESSAGE           00001030 R      05
MAKE_CONNECTION                 00000181 R      05
MASKA                       =   00000008
MASKL                       =   04000000
MAX_RETRY                   =   00000002
MIN_SEND_CREDIT             =   00000002
MSCP$B_BUFFER               =   00000010
MSCP$B_CNT_ALCS             =   00000004
MSCP$B_FLAGS                =   00000009
MSCP$B_OPCODE               =   00000008
MSCP$K_CM_EMULA             =   00000004
MSCP$K_CM_HSC50             =   00000001
MSCP$K_CM_RC25              =   00000003
MSCP$K_CM_TU81              =   00000005
MSCP$K_CM_UDA50             =   00000002
MSCP$K_CM_UDA52             =   00000006
MSCP$K_LEN                  =   00000030
MSCP$K_MXCMDLEN             =   00000024
MSCP$K_OP_ACPTH             =   00000042
MSCP$K_OP_AVAIL             =   00000008
MSCP$K_OP_AVATN             =   00000040
MSCP$K_OP_COMP              =   00000020
MSCP$K_OP_DUPUN             =   00000041
MSCP$K_OP_ERASE             =   00000012
MSCP$K_OP_ERGAP             =   00000016
MSCP$K_OP_GTCMD             =   00000002
MSCP$K_OP_GTUNT             =   00000003
MSCP$K_OP_ONLIN             =   00000009
MSCP$K_OP_READ              =   00000021
MSCP$K_OP_REPOS             =   00000025
MSCP$K_OP_STCON             =   00000004
MSCP$K_OP_STUNT             =   0000000A
MSCP$K_OP_WRITE             =   00000022
MSCP$K_OP_WRITM             =   00000024
MSCP$K_SC_DLATE             =   00000001
MSCP$K_SC_ODDBC             =   00000002
MSCP$K_ST_ABRTD             =   00000002
MSCP$K_ST_AVLBL             =   00000004
MSCP$K_ST_BOT               =   0000000D
MSCP$K_ST_CNTLR             =   0000000A
MSCP$K_ST_COMP              =   00000007
MSCP$K_ST_DATA              =   00000008
MSCP$K_ST_DRIVE             =   0000000B
MSCP$K_ST_FMTER             =   0000000C
MSCP$K_ST_HSTBF             =   00000009
MSCP$K_ST_ICMD              =   00000001
MSCP$K_ST_LED               =   00000013
MSCP$K_ST_OFFLN             =   00000003
MSCP$K_ST_PLOST             =   00000011
MSCP$K_ST_PRESE             =   00000012
MSCP$K_ST_RDTRN             =   00000010
MSCP$K_ST_SUCC              =   00000000
MSCP$K_ST_TAPEM             =   0000000E
MSCP$K_ST_WRTPR             =   00000006
MSCP$L_BYTE_CNT             =   0000000C
MSCP$L_CMD_REF              =   00000000
MSCP$L_CMD_STS              =   00000010
MSCP$L_DEV_PARM             =   0000001C
MSCP$L_MAXWTREC             =   00000024
MSCP$L_MEDIA_ID             =   0000001C
MSCP$L_OUT_REF              =   0000000C
MSCP$L_POSITION             =   0000001C
MSCP$L_RCSKIPED             =   0000000C
MSCP$L_REC_CNT              =   0000000C
MSCP$L_TMGP_CNT             =   00000010
MSCP$L_TMSKIPED             =   00000010
MSCP$M_MD_CLSEX             =   00002000
MSCP$M_MD_COMP              =   00004000
MSCP$M_MD_DLEOT             =   00000080
MSCP$M_MD_EXCLU             =   00000020
MSCP$M_MD_IMMED             =   00000040
MSCP$M_MD_OBJCT             =   00000004
MSCP$M_MD_REVRS             =   00000008
MSCP$M_MD_REWND             =   00000002
MSCP$M_MD_SEREC             =   00000100
MSCP$M_MD_UNLOD             =   00000010
MSCP$M_SC_EOT               =   00000400
MSCP$M_ST_MASK              =   0000001F
MSCP$M_TF_800               =   00000001
MSCP$M_TF_GCR               =   00000004
MSCP$M_TF_PE                =   00000002
MSCP$M_UF_VSMSU             =   00000020
MSCP$M_UF_WRTPH             =   00002000
MSCP$M_UF_WRTPS             =   00001000
MSCP$G_CNT_ID               =   00000014
MSCP$Q_TIME                 =   00000014
MSCP$Q_UNIT_ID              =   00000014
MSCP$S_ST_MASK              =   00000005
MSCP$V_CF_MLTHS             =   00000002
MSCP$V_EF_EOT               =   00000003
MSCP$V_EF_ERLOG             =   00000005
MSCP$V_EF_PLS               =   00000002
MSCP$V_MD_CLSEX             =   0000000D
MSCP$V_MD_COMP              =   0000000E
MSCP$V_MD_DLEOT             =   00000007
MSCP$V_MD_IMMED             =   00000006
MSCP$V_MD_SEREC             =   00000008
MSCP$V_OP_END               =   00000007
MSCP$V_SC_ALONL             =   00000008
MSCP$V_SC_DUPUN             =   00000007
MSCP$V_SC_INOPR             =   00000006
MSCP$V_ST_MASK              =   00000000
MSCP$V_TF_800               =   00000000
MSCP$V_TF_GCR               =   00000002
MSCP$V_TF_PE                =   00000001
MSCP$V_UF_VSMSU             =   00000005
MSCP$V_UF_WRTPH             =   0000000D
MSCP$V_UF_WRTPS             =   0000000C
MSCP$W_CNT_FLGS             =   0000000E
MSCP$W_CNT_TMO              =   00000010
MSCP$W_FORMAT               =   00000020
MSCP$W_FORMENU              =   00000024
MSCP$W_HST_TMO              =   00000010
```

```
MSCP$W_MODIFIER                 =  0000000A
MSCP$W_NOISEREC                 =  00000028
MSCP$W_SPEED                    =  00000022
MSCP$W_STATUS                   =  0000000A
MSCP$W_UNIT                     =  00000004
MSCP$W_UNT_FLGS                 =  0000000E
MSCPTOSPEED                        00000445 R      05
MSCPTOVMS_DENS                     00000425 R      05
MSCP_SRVR_NAME                     0000016B R      05
MSG_BUF_FAILURE                    00000595 R      05
MT$CHECK_ACCESS                    ******** X    05
MT$K_GCR_6250                   =  00000005
MT$K_NORMAL11                   =  0000000C
MT$K_NRZI_800                   =  00000003
MT$K_PE_1600                    =  00000004
MT$K_SPEED_DEF                  =  00000000
MT$M_BOT                        =  00010000
MT$M_DENSITY                    =  00001F00
MT$M_ENSEREXCP                  =  00000004
MT$M_EOF                        =  00020000
MT$M_EOT                        =  00040000
MT$M_HWL                        =  00080000
MT$M_LOST                       =  00100000
MT$M_SEREXCP                    =  00000001
MT$S_DENSITY                    =  00000005
MT$S_SPEED                      =  00000008
MT$V_BOT                        =  00000010
MT$V_DENSITY                    =  00000008
MT$V_ENSEREXCP                  =  00000002
MT$V_EOF                        =  00000011
MT$V_EOT                        =  00000012
MT$V_FORMAT                     =  00000004
MT$V_HWL                        =  00000013
MT$V_LOST                       =  00000014
MT$V_SPEED                      =  00000018
MT$V_SUP_GCR                    =  00000017
MT$V_SUP_NRZI                   =  00000015
MT$V_SUP_PE                     =  00000016
NOP_AVAIL                          000006B3 R      05
NOP_CTRLERR                        000006B3 R      05
NOP_DRVERR                         000006B3 R      05
NOP_IVCMD                          000006AB R      05
NOP_IVCMD_END                      000006B1 R      05
NOP_OFFLINE                        000006B3 R      05
NOP_SUCC                           000006B3 R      05
NORMAL_TRANSFEREND                 00000C9F R      05
ORB$B_FLAGS                     =  0000000B
ORB$B_TYPE                      =  0000000A
ORB$C_LENGTH                    =  00000058
ORB$L_OWNER                     =  00000000
ORB$M_PROT_16                   =  00000001
ORB$W_PROT                      =  00000018
ORB$W_SIZE                      =  00000008
PACKACK_CANCEL                     0000077F R      05
PACKACK_GTUNT_SUCC                 0000074B R      05
PACKACK_IVCMD                      00000752 R      05
PACKACK_IVCMD_END                  00000758 R      05
PACKACK_OFFLINE                    0000075C R      05
PACKACK_SUCC                       00000719 R      05
PB$B_RSTATION                   =  0000000C
PDT$C_ALLOCMSG                  =  00000014
PDT$L_DEALRGMSG                 =  00000024
PDT$L_DGOVRHD                   =  000000B8
PDT$L_MAPIRP                    =  00000034
PDT$L_MRESET                    =  00000070
PDT$L_MSTART                    =  00000074
PDT$L_QUEUEDG                   =  0000003C
PDT$L_RCHMSGBUF                 =  00000044
PHYIO_VOLINV                       000005DE R      05
PR$_IPL                         =  00000012
PRP_STCON_MSG                      0000028B R      05
RD$C_CDRP                       =  00000000
RECONN_COMMON                      00000D63 R      05
RECORD_COMMON                      000007AA R      05
RECORD_GETUNIT_CHAR                000007A3 R      05
RECORD_ONLINE                      00000795 R      05
RECORD_SETUNIT_CHAR                00000795 R      05
RECORD_STCON                       000002BF R      05
RESTART_FIRST_CDRP                 00000DCE R      05
RESTART_NEXT_CDRP                  00000E86 R      05
REWIND_ABORT                       00000984 R      05
REWIND_AVAIL                       00000984 R      05
REWIND_CTRLERR                     00000984 R      05
REWIND_DRVERR                      00000984 R      05
REWIND_END                         00000984 R      05
REWIND_FMTER                       00000984 R      05
REWIND_IVCMD                       0000096A R      05
REWIND_IVCMD_END                   00000970 R      05
REWIND_OFFLINE                     00000984 R      05
REWIND_PRESE                       00000984 R      05
REWIND_SUCC                        00000974 R      05
SCS$ALLOC_RSPID                    ********   X    05
SCS$CONNECT                        ********   X    05
SCS$DISCONNECT                     ********   X    05
SCS$FIND_RDTE                      ********   X    05
SCS$LKP_RDTCDRP                    ********   X    05
SCS$LKP_RDTWAIT                    ********   X    05
SCS$RECVL_RSPID                    ********   X    05
SCS$UNSTALLUCB                     ********   X    05
SENSEMODE_ONLINE                   00000B7E R      05
SENSEMODE_RETURN                   00000B84 R      05
SETMODE_ABORT                      00000A8E R      05
SETMODE_BEGIN_IVCMD                00000AB9 R      05
SETMODE_CANCEL                     00000A9A R      05
SETMODE_CTRLERR                    00000A8E R      05
SETMODE_DRVERR                     00000A8E R      05
SETMODE_IVCMD                      00000B40 R      05
SETMODE_IVCMD_END                  00000B46 R      05
SETMODE_OFFLINE                    00000A8E R      05
SETMODE_ONLINE                     00000A9D R      05
SETMODE_RETURN                     00000B4D R      05
SETMODE_SUCC                       00000B4A R      05
SET_CLEAR_SEX                      0000046A R      05
SGN$GL_VMSD3                       ********   X    05
```

```
SKIP_ABORT                      00000A17 R      05
SKIP_AVAIL                      00000A17 R      05
SKIP_BOT                        00000A29 R      05
SKIP_COMMON                     00000991 R      05
SKIP_CTRLERR                    00000A2D R      05
SKIP_DRVERR                     00000A2D R      05
SKIP_END                        00000A51 R      05
SKIP_EOF                        00000A23 R      05
SKIP_FMTER                      00000A2D R      05
SKIP_IVCMD                      00000A0F R      05
SKIP_IVCMD_END                  00000A15 R      05
SKIP_LEOT                       00000A2D R      05
SKIP_OFFLINE                    00000A17 R      05
SKIP_PLOST                      00000A1D R      05
SKIP_PRESE                      00000A17 R      05
SKIP_SUCC                       00000A2D R      05
SPEEDTOMSCP                     00000430 R      05
SS$_ABORT                   =   0000002C
SS$_BUGCHECK                =   000002A4
SS$_CTRLERR                 =   00000054
SS$_DATACHECK               =   0000005C
SS$_DATALATE                =   00002274
SS$_DATAOVERUN              =   00000838
SS$_DEVOFFLINE              =   00000084
SS$_DRVERR                  =   0000008C
SS$_DUPUNIT                 =   000021C4
SS$_ENDOFFILE               =   00000870
SS$_ENDOFTAPE               =   00000878
SS$_ENDOFVOLUME             =   000009A0
SS$_ILLIOFUNC               =   000000F4
SS$_IVBUFLEN                =   0000034C
SS$_MEDOFL                  =   000001A4
SS$_NORMAL                  =   00000001
SS$_PARITY                  =   000001F4
SS$_SERIOUSEXCP             =   000021D4
SS$_VOLINV                  =   00000254
SS$_WRITLCK                 =   0000025C
START_AVAILABLE                 00000818 R      05
START_DSE                       00000887 R      05
START_ERASETAPE                 00000881 R      05
START_NOP                       00000676 R      05
START_PACKACK                   000006B8 R      05
START_READPBLK                  00000B9C R      05
START_RECAL                     0000091C R      05
START_REWIND                    0000091C R      05
START_REWINDOFF                 00000814 R      05
START_SENSECHAR                 00000B66 R      05
START_SENSEMODE                 00000B66 R      05
START_SETCHAR                   00000A54 R      05
START_SETMODE                   00000A59 R      05
START_SKIPFILE                  00000987 R      05
START_SKIPRECORD                0000098D R      05
START_SPACEFILE                 00000987 R      05
START_SPACERECORD               0000098D R      05
START_UNLOAD                    00000814 R      05
START_WRITECHECK                00000B87 R      05
START_WRITEMARK                 00000897 R      05
START_WRITEOF                   00000897 R      05
START_WRITEPBLK                 00000B96 R      05
TERMINATE_PENDING               000002FD R      05
TRANSFER_BOT                    00000C48 R      05
TRANSFER_COMPERR                00000C96 R      05
TRANSFER_CTRLERR                00000C5B R      05
TRANSFER_DATA_ERROR             00000C96 R      05
TRANSFER_EOF                    00000C42 R      05
TRANSFER_HOST_BUFFER_ERROR      00000C88 R      05
TRANSFER_INVALID_COMMAND        00000C70 R      05
TRANSFER_IVCMD_END              00000C76 R      05
TRANSFER_MEDOFL                 00000C7A R      05
TRANSFER_PLOST                  00000C3C R      05
TRANSFER_PRESE                  00000C51 R      05
TRANSFER_RTN_BCNT               00000C96 R      05
TRANSFER_RTN_RECLEN             00000C96 R      05
TRANSFER_SHIFT                  00000C9A R      05
TU$CONNECT_ERR                  00000D5F R      05
TU$DDT                          00000000 RG     05
TU$DGDR                         00001046 R      05
TU$IDR                          00000F94 R      05
TU$RE_SYNCH                     00000D4B R      05
TU$TMR                          00000EF0 R      05
TU_ABSDENS                      00000400 R      05
TU_ABSPEED                      00000408 R      05
TU_BEGIN_IVCMD                  00000601 R      05
TU_CONTROLLER_INIT              000000C0 R      05
TU_FUNCTABLE                    00000038 R      05
TU_MSCPDENS                     000003FD R      05
TU_REAL_STARTIO                 000005C5 R      05
TU_REDO_IO                      00000601 R      05
TU_RESTARTIO                    000005CB R      05
TU_STARTIO                      00000598 R      05
TU_UNSOLNT                      00001095 R      05
TU_VMSDENS                      000003F9 R      05
UCB$B_DEVCLASS              =   00000040
UCB$B_DEVTYPE               =   00000041
UCB$B_DIPL                  =   0000005E
UCB$B_FIPL                  =   0000000B
UCB$B_TYPE                  =   0000000A
UCB$K_MSCP_TAPE_LENGTH      =   000000EC
UCB$K_TU_LENGTH             =   000000F8
UCB$L_2P_ALTUCB             =   000000A8
UCB$L_CDDB                  =   000000BC
UCB$L_CDDB_LINK             =   000000C4
UCB$L_CDT                   =   000000C8
UCB$L_DEVCHAR               =   00000038
UCB$L_DEVCHAR2              =   0000003C
UCB$L_DEVDEPEND             =   00000044
UCB$L_IOQBL                 =   00000050
UCB$L_IOQFL                 =   0000004C
UCB$L_LINK                  =   00000030
UCB$L_MEDIA_ID              =   0000008C
UCB$L_MSCPDEVPARAM          =   000000D8
UCB$L_PDT                   =   00000084
UCB$L_RECORD                =   000000B0
UCB$L_STS                   =   00000064
```

```
UCB$L_TU_MAXWRCNT               000000EC
UCB$M_BSY                     = 00000100
UCB$M_MSCP_INITING            = 00000200
UCB$M_MSCP_WAITBMP            = 00000400
UCB$M_MSCP_WRTP               = 00002000
UCB$M_ONLINE                  = 00000010
UCB$M_TU_SEQNOP               = 00000004
UCB$M_VALID                   = 00000800
UCB$Q_UNIT_ID                 = 000000CC
UCB$V_BSY                     = 00000008
UCB$V_MSCP_WAITBMP            = 0000000A
UCB$V_MSCP_WRTP               = 0000000D
UCB$V_TU_SEQNOP               = 00000002
UCB$V_VALID                   = 0000000B
UCB$W_DEVBUFSIZ               = 00000042
UCB$W_DEVSTS                  = 00000068
UCB$W_MSCPUNIT                = 000000D4
UCB$W_RWAITCNT                = 00000056
UCB$W_SIZE                    = 00000008
UCB$W_STS                     = 00000064
UCB$W_TU_FORMAT                 000000F0
UCB$W_TU_NOISE                  000000F4
UCB$W_TU_SPEED                  000000F2
UCB$W_UNIT_FLAGS              = 000000E0
UNIT_AVAILABLE_ATTN             00001019 R      05
VALID_PACKACK                   0000078E R      05
VEC$L_INITIAL                 = 0000000C
VMSTOMSCP_DENS                  0000040C R      05
VOL_INVALID                     00000578 R      05
WRITM_ABORT                     000008F8 R      05
WRITM_AVAIL                     000008F8 R      05
WRITM_CTRLERR                   000008F8 R      05
WRITM_DATA_ERROR                000008F8 R      05
WRITM_DRVERR                    000008F8 R      05
WRITM_END                       00000908 R      05
WRITM_FMTER                     000008F8 R      05
WRITM_IVCMD                     000008EA R      05
WRITM_IVCMD_END                 000008F0 R      05
WRITM_OFFLINE                   000008F8 R      05
WRITM_PRESE                     00000919 R      05
WRITM_SUCC                      000008F8 R      05
WRITM_WRITLCK                   000008F8 R      05
WTM_ERASE_COM                   0000089B R      05
XFER_IVCMD_END                  00000C3A R      05
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000001F8 (504.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$$$200_TEMPLATE_UCB_01	000000F8 (248.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$200_TEMPLATE_ORB_01	00000058 (88.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$105_PROLOGUE	00000083 (131.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00001099 (4249.)	05 (5.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$220_DUTU_DATA_01	00000004 (4.)	06 (6.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$220_DEVTYPE_TABLE_01	00000019 (25.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.04	00:00:01.28
Command processing	109	00:00:00.47	00:00:02.87
Pass 1	1050	00:00:43.71	00:02:52.53
Symbol table sort	0	00:00:03.78	00:00:11.25
Pass 2	411	00:00:10.19	00:00:37.49
Symbol table output	1	00:00:00.40	00:00:02.65
Psect synopsis output	0	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1603	00:00:58.62	00:03:48.10

The working set limit was 3000 pages.
322530 bytes (630 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3488 non-local and 113 local symbols.
4539 source lines were read in Pass 1, producing 42 object records in Pass 2.
97 pages of virtual memory were used to define 89 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]DUTULIB.MLB;1	16
_255DUA28:[SYS.OBJ]LIB.MLB;1	50
_255DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	78

3948 GETS were required to define 78 macros.

There were no errors, warnings or information messages.

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

TUDRIVER
LIS

XADRIVER
LIS

TSDRIVER
LIS