```
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 RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     RRRRRRRRRRRR
DDD          DDD RRRRRRRRRRRR           III      VVV         VVV  EEEEEEEEEEEE     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
```

FILEID**DUDRIVER

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

Table of contents

```
0000     1          .TITLE  DUDRIVER - DISK CLASS DRIVER
0000     2          .IDENT  'V04-001'
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  11-May-1981
0000    29  ;
0000    30  ;  DISK CLASS DRIVER
0000    31  ;
0000    32  ; MODIFIED BY
0000    33  ;
0000    34  ;       V04-001 ROW0413         Ralph O. Weber          6-SEP-1984
0000    35  ;               Add check in START_MOUNT_VER which causes SS$_VOLINV to be
0000    36  ;               returned when a mount verification request is presented for a
0000    37  ;               device which does not have UCB$V_VALID set.  Mount
0000    38  ;               verification guarantees to have UCB$V_VALID set, even for the
0000    39  ;               first IO$_PACKACK.  Therefore, when such an event occurs,
0000    40  ;               mount verification needs to be informed.
0000    41  ;
0000    42  ;       V03-255 ROW0411         Ralph O. Weber          21-AUG-1984
0000    43  ;               Correct register usage in ROW0360 to reference UCB from R3 not
0000    44  ;               R5.
0000    45  ;
0000    46  ;       V03-254 ROW0410         Ralph O. Weber          6-AUG-1984
0000    47  ;               Setup use of UCB$V_CLUTRAN to guarantee that mount
0000    48  ;               verification always runs for all disks after a VAXcluster
0000    49  ;               state transition.
0000    50  ;
0000    51  ;       V03-253 ROW0400         Ralph O. Weber          21-JUL-1984
0000    52  ;               Eliminate waiting for write access to the I/O database mutex
0000    53  ;               before calling DUTU$FAILOVER_UCB.  That routine will now wait
0000    54  ;               for the mutex in its own fork thread.  This also prevents
0000    55  ;               stalling requests due to lack of access to the mutex.  Since
0000    56  ;               such requests can no longer stall, it is no longer possible to
0000    57  ;               get a deadlock when, for example, a process holding the mutex
```

```
0000     58 ;           must perform a page fault read in order to complete and
0000     59 ;           release the mutex.
0000     60 ;
0000     61 ;     V03-252 ROW0399         Ralph O. Weber          21-JUL-1984
0000     62 ;           Add media-id to device type translation table entries for the
0000     63 ;           RA82, RC26, RCF26, CDR50, RD53, RX31, RX32, and RX18.
0000     64 ;
0000     65 ;     V03-251 ROW0398         Ralph O. Weber          21-JUL-1984
0000     66 ;           Setup use of class driver write-lock bit in UCB$W_DEVSTS.
0000     67 ;           Also eliminate alteration and use of DEV$V_SWL bit in
0000     68 ;           UCB$L_DEVCHAR.  That bit is controlled by the file system.
0000     69 ;
0000     70 ;     V03-250 ROW0395         Ralph O. Weber          21-JUL-1984
0000     71 ;           Make changes which setup "normal" MSCP command timeout
0000     72 ;           algorithm before calls to DUTU$POLL_FOR_UNITS.  Also setup
0000     73 ;           use of DAP CDRP by DUTU$POLL_FOR_UNITS.
0000     74 ;
0000     75 ;     V03-249 ROW0394         Ralph O. Weber          21-JUL-1984
0000     76 ;           Remove DPT_STORE setting of ACL queue present bit in the ORB.
0000     77 ;           This should improve performance on devices which do not really
0000     78 ;           have an ACL queue in their device protection ORB.
0000     79 ;
0000     80 ;     V03-248 ROW0389         Ralph O. Weber           9-JUL-1984
0000     81 ;           Cause both mount verification requests and requests for
0000     82 ;           devices which are not "volume valid" to fail during broken
0000     83 ;           connection cleanup after their resources have been released.
0000     84 ;           This insures the requests from $MOUNT will fail if the
0000     85 ;           connection to the remote MSCP server fails during processing
0000     86 ;           of the $MOUNT command.
0000     87 ;
0000     88 ;     V03-247 ROW0388         Ralph O. Weber           8-JUL-1984
0000     89 ;           Remove wait counter adjustment for possibly failed-over disk
0000     90 ;           from DU_END_MNTVER.  Because of some race conditions, this
0000     91 ;           adjustment must be done in DUTU$FAILOVER_UCB.
0000     92 ;
0000     93 ;     V03-246 ROW0387         Ralph O. Weber           7-JUL-1984
0000     94 ;           Setup use of DUTU$RECONN_LOOKUP and DUTU$DRAIN_CDDB_CDRPQ.
0000     95 ;
0000     96 ;     V03-245 ROW0381         Ralph O. Weber          22-JUN-1984
0000     97 ;           Close a small window in volume not software enabled processing
0000     98 ;           so that if there is any hope of getting an invalid volume back
0000     99 ;           requests will wait until the volume returns.  Also add RD52 to
0000    100 ;           media-id conversion table.
0000    101 ;
0000    102 ;     V03-244 ROW0377         Ralph O. Weber           7-JUN-1984
0000    103 ;           Change order of bit test on DEV$V_2P and call to
0000    104 ;           DUTU$DEALLOC_ALL in PACKACK_OFFLINE so that R0 (the return
0000    105 ;           status) is not corrupted when the bit test branch is taken.
0000    106 ;
0000    107 ;           Correct branch destination in DU_REVALIDATE_DISKS to eliminate
0000    108 ;           infinite loop when performing out-of-quorum processing for a
0000    109 ;           non-multihost single pathed device.
0000    110 ;
0000    111 ;     V03-243 ROW0371         Ralph O. Weber          29-MAY-1984
0000    112 ;           Put a band-aid on problems related to the need to hold the I/O
0000    113 ;           database mutex during a failover.  The band-aid is to
0000    114 ;           eliminate waiting when there is not alternate path to which
```

```
0000   115 ;                one can failover.
0000   116 ;
0000   117 ;        V03-242 ROW0369         Ralph O. Weber          22-MAY-1984
0000   118 ;                Change DU$RE_SYNCH to not do MRESET/MSTART to MSCP servers and
0000   119 ;                then wait for something to happen.  Quite possibly, nothing
0000   120 ;                ever will happen in such cases.  Proceeding directly to the
0000   121 ;                DISCONNECT is the correct action.
0000   122 ;
0000   123 ;        V03-241 ROW0365         Ralph O. Weber          17-MAY-1984
0000   124 ;                Correct method for counting four passes through START_PACKACK
0000   125 ;                when failover operations are attempted.
0000   126 ;
0000   127 ;        V03-240 ROW0361         Ralph O. Weber          5-MAY-1984
0000   128 ;                Setup use of new class driver common DAP processing in
0000   129 ;                DUTU$DODAP.  The new routine is designed to eliminate multiple
0000   130 ;                concurrent DAP threads which are known to crash systems.
0000   131 ;
0000   132 ;        V03-239 ROW0360         Ralph O. Weber          4-MAY-1984
0000   133 ;                Cause I/O requests to stall while a IO$_PACKACK is in progress
0000   134 ;                by bumping RWAITCNT.  This corrects system crashes during
0000   135 ;                booting where a paging I/O request gets SS$_VOLINV because the
0000   136 ;                packack for SYSINIT's mounting of the system disk is in
0000   137 ;                progress.
0000   138 ;
0000   139 ;                Alter DU_REVALIDATE_DISKS for use in the stalling of all disk
0000   140 ;                I/O activity whenever quorum is lost.  Change the routine
0000   141 ;                itself to correctly handle being entered for quorum lost
0000   142 ;                processing.  Cause the routine to be pointed to as the
0000   143 ;                unsolicited interrupt routine.  Until a new DDT entry can be
0000   144 ;                properly defined, this will allow the routine to be located by
0000   145 ;                the connection manager quorum lost algorithm.
0000   146 ;
0000   147 ;        V03-238 ROW0359         Ralph O. Weber          4-MAY-1984
0000   148 ;                Correct the $DISPATCH after PACKACK_OFFLINE to preserve R0
0000   149 ;                which contains SS$_MEDOFL, a possible completion status.
0000   150 ;
0000   151 ;        V03-237 ROW0354         Ralph O. Weber          30-APR-1984
0000   152 ;                Add setting for DEV$M_NNM in DEVCHAR2 to indicate that tape
0000   153 ;                class driver devices use NODENAME$DDCN device names.
0000   154 ;
0000   155 ;        V03-236 ROW0351         Ralph O. Weber          24-APR-1984
0000   156 ;                Correct MSCP offline processing in PACKACK so that spun-down
0000   157 ;                drives return SS$_MEDOFL.
0000   158 ;
0000   159 ;        V03-235 ROW0350         Ralph O. Weber          23-APR-1984
0000   160 ;                Correct more problems causing multiple trips through
0000   161 ;                END_SINGLE_STREAM, with the attendent bugchecks.  First, clear
0000   162 ;                CDDB$V_SNGLSTRM upon entry to DU$CONNECT_ERR.  Second, protect
0000   163 ;                the SCS$UNSTALLUCB loop in END_SINGLE_STREAM from possible
0000   164 ;                connection failures during execution of the loop.
0000   165 ;
0000   166 ;        V03-234 LMP0237         L. Mark Pilant,         19-Apr-1984  11:23
0000   167 ;                Also initialize the size and type fields of the template ORB.
0000   168 ;
0000   169 ;        V03-233 LMP0235         L. Mark Pilant,         17-Apr-1984  13:54
0000   170 ;                Add a template ORB to be hooked up to the template UCB when
0000   171 ;                a new unit is initialized.
```

```
0000    172 ;
0000    173 ;       V03-232 ROW0345         Ralph O. Weber          11-APR-1984
0000    174 ;               Change replace logic so that a replace is not even attempted
0000    175 ;               when the DEV$V_SWL bit is set in UCB$L_DEVCHAR.
0000    176 ;
0000    177 ;       V03-231 ROW0339         Ralph O. Weber          8-APR-1984
0000    178 ;               Setup use of common invalid command processing routines
0000    179 ;               (macros).  This replaces the old "form the original MSCP
0000    180 ;               command packet by hand" algorithm with a "repeat the code
0000    181 ;               which formed the original MSCP command" algorithm.  The cost
0000    182 ;               is a single, hardly ever taken BLBS in the mainline read/write
0000    183 ;               code path.  The savings are elimination of having to duplicate
0000    184 ;               command packet setup changes in the invalid command case,
0000    185 ;               hundreds of bytes of code, and a not inconsequential amount of
0000    186 ;               static storage.
0000    187 ;
0000    188 ;       V03-230 ROW0338         Ralph O. Weber          7-APR-1984
0000    189 ;               Setup use of DO_ACTION macro to replace INTERPRET_ACTION_TABLE.
0000    190 ;               Start using IF_MSCP where only success or failure of an MSCP
0000    191 ;               command is being tested.  Setup use of ACTION_ENTRY END to end
0000    192 ;               action tables.  Remove action table interpretation routines;
0000    193 ;               they are now in DUTUSUBS.
0000    194 ;
0000    195 ;       V03-229 ROW0335         Ralph O. Weber          4-APR-1984
0000    196 ;               Perform numerous class driver fixups and cleanups:
0000    197 ;               > Correct positioning of DPT_STORE REINIT and add note that
0000    198 ;                 reinit is not significant because driver is not reloadable.
0000    199 ;               > Remove unitinit routine which was causing crashes after
0000    200 ;                 power failures and replace it with a pointer to the new
0000    201 ;                 routine in DUTUSUBS which does not have the power failure
0000    202 ;                 problems.
0000    203 ;               > Remove usage of allocation class value in the SCS connect
0000    204 ;                 accept message.  All MSCP servers now supply that
0000    205 ;                 information in the Set Controller Characteristics command
0000    206 ;                 end packet.
0000    207 ;               > Do not queue stalled I/O when UCB$V_VALID is clear and the
0000    208 ;                 "stalled" request is not a physical function.
0000    209 ;               > Eliminate call to DU$ONLINE_COMPLETE whenever the ONLINE
0000    210 ;                 command reports that the device was already online when the
0000    211 ;                 command was issued.
0000    212 ;               > Remove unnecessary register saves and restores around calls
0000    213 ;                 to DU$UNLOCK_HIRT.
0000    214 ;               > Add processing for IO$M_INHRETRY and IO$M_EXPRESS.  Also
0000    215 ;                 rework read/write modifer testing so that only one test for
0000    216 ;                 all possible modifers is in-line for the read/write path.
0000    217 ;                 If any modifer is set, an out-of-line code segment
0000    218 ;                 individually tests and processes the modifiers.
0000    219 ;               > Add the multi-host progress counter handling proposed by the
0000    220 ;                 HSC implementors to DU$TMR.  This algorithm simplifies
0000    221 ;                 handling of the case where the MSCP server is busy on an
0000    222 ;                 older command from another host.
0000    223 ;
0000    224 ;       V03-228 ROW0331         Ralph O. Weber          30-MAR-1984
0000    225 ;               Setup use of common cancel support in DUTUSUBS.  Also make
0000    226 ;               functions which use multiple MSCP commands check for cancel
0000    227 ;               after each MSCP command and perform cancel if necessary.
0000    228 ;
```

```
0000    229 ;
0000    230 ;       V03-227 ROW0329         Ralph O. Weber          22-MAR-1984
0000    231 ;               Give up on preventing failover to a non-operational path as
0000    232 ;               attempted in ROW0300, ROW0319, and ROW0328.  It does not work.
0000    233 ;               Replace it with tests in DU_END_MOUNTVER which determine
0000    234 ;               whether or not a pass through RESTART_NEXT_CDRP is in the
0000    235 ;               cards for the current connection (at the time of mount
0000    236 ;               verification completion).
0000    237 ;
0000    238 ;       V03-226 ROW0328         Ralph O. Weber          21-MAR-1984
0000    239 ;               Correct bugs in ROW0319 which caused it to incorrectly miss
0000    240 ;               the end of the CDDB UCB chain.
0000    241 ;
0000    242 ;       V03-225 ROW0323         Ralph O. Weber          12-MAR-1984
0000    243 ;               Rework MSCP offline handling in PACKACK so that both "unknown"
0000    244 ;               and "inoperative" subcodes cause failover attempts.
0000    245 ;
0000    246 ;       V03-224 ROW0319         Ralph O. Weber          28-FEB-1984
0000    247 ;               Attempt to eliminate failover to non-operational path by
0000    248 ;               making clearing of CDDB$V_RECONNECT the last thing done in
0000    249 ;               END_SINGLE_STREAM.  Also add sanity check that CDDB$V_RECONNECT
0000    250 ;               is set before it is cleared.
0000    251 ;
0000    252 ;       V03-223 ROW0306         Ralph O. Weber          13-FEB-1984
0000    253 ;               Change end mount verification processing so that RWAITCNT
0000    254 ;               reduction occurs whether or not mount verification succeeds.
0000    255 ;               This averts hangs produced by mount verification aborting
0000    256 ;               after a failover occurs.
0000    257 ;
0000    258 ;       V03-222 ROW0302         Ralph O. Weber          10-FEB-1984
0000    259 ;               Add clearing of DEV$V_SWL to IO$_AVAILABLE and IO$_UNLOAD
0000    260 ;               function processing.  This counteracts the effects of setting
0000    261 ;               the bit in IO$_PACKACK processing.
0000    262 ;
0000    263 ;       V03-221 ROW0301         Ralph O. Weber          9-FEB-1984
0000    264 ;               Remove clearing CDDB$V_NOCONN bit from MAKE_CONNECTION, this
0000    265 ;               bit cannot be cleared until all UCBs chained to the CDDB have
0000    266 ;               had their connection dependent fields updated to reflect the
0000    267 ;               new connection.  In the re-connect code following the call to
0000    268 ;               MAKE_CONNECTION, move the loop to test RWAITCNT in each
0000    269 ;               chained UCB to before the call to DUTU$POLL_FOR_UNITS and add
0000    270 ;               initialization of the connection dependent UCB fields to the
0000    271 ;               operations performed by this loop.  After that loop, insert
0000    272 ;               clearing the CDDB$V_NOCONN bit followed by the call to
0000    273 ;               DUTU$POLL_FOR_UNITS.
0000    274 ;
0000    275 ;               These changes guarantee: 1. that mount verification thread
0000    276 ;               will not attempt MSCP requests until AFTER the UCB connection
0000    277 ;               dependent fields have be altered to reflect the current
0000    278 ;               connection, 2. that the RWAITCNT test is made at a time when
0000    279 ;               there is not active MSCP traffic on the connection (this is
0000    280 ;               important because the test cannot account for RWAITCNT being
0000    281 ;               bumped due to threads waiting on SCS resources), 3. that both
0000    282 ;               mount verification and poll-for-units thread can proceed
0000    283 ;               concurrently.
0000    284 ;
0000    285 ;               Some comments regarding the testing of for an empty queue of
                            outstanding requests have been removed.  The test itself was
```

```
0000   286 ;               removed in ROW0279 because mount verification threads can very
0000   287 ;               well have outstanding requests at the time the test used to be
0000   288 ;               made.
0000   289 ;
0000   290 ;       V03-220 ROW0300         Ralph O. Weber          9-FEB-1984
0000   291 ;               Enhance DU_END_MNTVER to unbump RWAITCNT if the
0000   292 ;               UCB$V_MSCP_WAITBMP flag is set.  This handles the case where
0000   293 ;               mount verification has successfully failed a drive from a
0000   294 ;               crashed controller to a working controller.
0000   295 ;
0000   296 ;       V03-219 ROW0298         Ralph O. Weber          9-FEB-1984
0000   297 ;               Setup use of CDRP$W_ENDMSGSIZ to hold the size of an incomming
0000   298 ;               sequenced message.  This replaces use of CDRP$L_IOST2+2 whose
0000   299 ;               use causes valuable input information to be overwritten.
0000   300 ;
0000   301 ;       V03-218 ROW0296         Ralph O. Weber          6-FEB-1984
0000   302 ;               Setup use of CDDB$V_RSTRTWAIT status bit in end-of-mount-
0000   303 ;               verification processing to tell whether or not to call
0000   304 ;               RESTART_NEXT_CDRP.
0000   305 ;
0000   306 ;       V03-217 ROW0295         Ralph O. Weber          6-FEB-1984
0000   307 ;               Set use of DUTU$CHECK_RWAITCNT, a routine which validates
0000   308 ;               UCB$W_RWAITCNT and bug checks if it is wrong.
0000   309 ;
0000   310 ;       V03-216 ROW0294         Ralph O. Weber          5-FEB-1984
0000   311 ;               Correct RECORD_STCON setup of allocation class information in
0000   312 ;               the DDBs to use DDB$L_CONLINK so that only those DDBs on this
0000   313 ;               connection are effected.
0000   314 ;
0000   315 ;       V03-215 ROW0293         Ralph O. Weber          4-FEB-1984
0000   316 ;               Cleanup references to INIT_MSG_BUF and RECYCLE_ALL and remove
0000   317 ;               them.  Restructure START_PACKACK so that the normal path is
0000   318 ;               more linear.  In START_PACKACK, remove the notes about over-
0000   319 ;               riding a bad RCT as this can be done by write-locking the
0000   320 ;               device and add a tests which rejects attempts to mount 576
0000   321 ;               byte/sector disks.  Setup return of SS$_SHACHASTA for shadow
0000   322 ;               set state change on all commands.
0000   323 ;
0000   324 ;       V03-214 ROW0287         Ralph O. Weber          22-JAN-1984
0000   325 ;               Make all remaining speed enhancements to mainline READ/WRITE
0000   326 ;               path.  Setup support for IO$M_FORCERR modifier on WRITEs.
0000   327 ;
0000   328 ;       V03-213 ROW0286         Ralph O. Weber          22-JAN-1984
0000   329 ;               Cleanup all references to routines now moved to DUTUSUBS.  Fix
0000   330 ;               usage of CDRP$V_ERLOGIP to be CDRP$V_ERLIP.
0000   331 ;
0000   332 ;       V03-212 ROW0285         Ralph O. Weber          21-JAN-1984
0000   333 ;               Move all host initiated replacement code, etc. to a seperate
0000   334 ;               module, DUHIRT.
0000   335 ;
0000   336 ;       V03-211 ROW0283         Ralph O. Weber          16-JAN-1984
0000   337 ;               Take all reasonable steps to speed-up the mainline READ/WRITE
0000   338 ;               path through the driver.  Eliminate taken branches and slow
0000   339 ;               instructions.  Make I/O success case avoid call to
0000   340 ;               INTERPRET_ACTION_TABLE.
0000   341 ;
0000   342 ;       V03-210 ROW0279         Ralph O. Weber          14-JAN-1984
```

```
0000    343 ;
0000    344 ;               Change mechanism for restoring units to online state after
0000    345 ;               connection failure from use of BRING_UNIT_ONLINE to use of
0000    346 ;               mount verification.  This includes a complete reworking of
0000    347 ;               mount verification processing for the "normal" case too.  Also
0000    348 ;               allow all unit failover to occur due to mount verification
0000    349 ;               sending PACKACK requests to the driver.
0000    350 ;
0000    351 ;       V03-209 ROW0272         Ralph O. Weber          31-DEC-1983
0000    352 ;               Remove block which prevents issuing DAP requests from DU$TMR.
0000    353 ;               Change START_DAP_THREAD to only send Determin Access Paths
0000    354 ;               commands for those UCBs which are UCB$V_VALID.  MSCP servers
0000    355 ;               will ignore DAP commands for units which are not MSCP online,
0000    356 ;               so why should we send them.  Add block which prevents logging
0000    357 ;               errors for DAP attention messages to ACCESS_PATH_ATTN.  This
0000    358 ;               allows the code which logs DAP attention messages to remain
0000    359 ;               and to be patched back into existance should it be needed.
0000    360 ;
0000    361 ;       V03-208 ROW0270         Ralph O. Weber          29-DEC-1983
0000    362 ;               Eliminate DRIVER_SEND_MSG_BUF by replacing all calls to it
0000    363 ;               with SEND_MSCP_MSG_DRIVER.  Change MAKE_CONNECTION to use the
0000    364 ;               larger of HSTIMEOUT_ARRAY[controller_model] and the controller
0000    365 ;               timeout value as the final host timeout value for the MSCP Set
0000    366 ;               Controller Characteristics command.  Setup use of VMS SCS
0000    367 ;               RECYCL_RSPID and FIND_RSPID_RDTE (except in DU$IDR where speed
0000    368 ;               is desired).  Make all permanent/DAP CDRP to CDDB conversions
0000    369 ;               use PERMCDRP_TO_CDDB.
0000    370 ;
0000    371 ;       V03-207 ROW0269         Ralph O. Weber          29-DEC-1983
0000    372 ;               Change DU_CONTROLLER_INIT to use DUTU$CREATE_CDDB.
0000    373 ;
0000    374 ;       V03-206 ROW0267         Ralph O. Weber          28-DEC-1983
0000    375 ;               Add IO$M_SHADOW modifier for IO$_PACKACK.  This is the PACKACK
0000    376 ;               support for volume shadowing.
0000    377 ;
0000    378 ;       V03-205 ROW0262         Ralph O. Weber          24-NOV-1983
0000    379 ;               Move all UCB lookup and creation to DUTUSUBS.  Cleanup
0000    380 ;               ATTN_MSG processing in DU$IDR.  Implement usage of $DUTUDEF,
0000    381 ;               all device independent UCB fields, and the IOC$GL_DU_CDDB
0000    382 ;               listhead.  Replace all DPT_STORE macros which init UCB fields
0000    383 ;               with INIT_UCB macros.  INIT_UCB initializes both the DPT and
0000    384 ;               the template UCB.  Its use eliminates possible mismatch of the
0000    385 ;               two UCB sources as well as some setup code in the controller
0000    386 ;               initialization routine.  Make driver not reloadable.  Change
0000    387 ;               POLL_FOR_UNITS to DUTU$POLL_FOR_UNITS.
0000    388 ;
0000    389 ;       V03-204 ROW0261         Ralph O. Weber          22-NOV-1983
0000    390 ;               Move DUMP_COMMAND and DUMP_ENDMESSAGE to DUTUSUBS.  Change
0000    391 ;               DU$END to DUTU$END so that linking with multiple modules does
0000    392 ;               not involve a hack.  Do some common path cleanup to speed
0000    393 ;               passage through the common code paths.  Change subroutine
0000    394 ;               CALL_SEND_MSG_BUF to SEND_MSCP_MSG macro.  Move INIT_TPLATE_UCB
0000    395 ;               to DOTULIB (macro library).
0000    396 ;
0000    397 ;       V03-203 RLRQBUS         Robert L. Rappaport     16-NOV-1983
0000    398 ;               Change building of transfer commands MSCP packet so that
0000    399 ;               PQDRIVER can alter the mapping information during a map
                            request and have the altered information appear in the MSCP
```

```
0000    400 ;
0000    401 ;               packet.  Also fix a credit bug when invalid media addresses or
0000    402 ;               buffer lengths are encountered.  In such cases, a NOP command
0000    403 ;               must be sent to release the already allocated resources.
0000    404 ;
0000    405 ;       V03-202 ROW0253         Ralph O. Weber          12-NOV-1983
0000    406 ;               Change device dependent UCB definitions to work with globally
0000    407 ;               defined MSCP extension to the UCB.  This change does not make
0000    408 ;               use of all the UCB fields in the new extension.  It simply
0000    409 ;               eliminates interactions which will prevent this module from
0000    410 ;               building in the presence of the new UCB definitions.  The
0000    411 ;               UCB$L_DU_MEDIATYP field, which was changed to UCB$L_MEDIA_ID
0000    412 ;               ages ago, has also been eliminated.
0000    413 ;
0000    414 ;       V03-201 ROW0248         Ralph O. Weber          8-NOV-1983
0000    415 ;               Remove setting of DEV$M_RCT from INIT_CONNDEP_UCB and change
0000    416 ;               RECORD_UNIT_STATUS to set DEV$M_RCT if and only if the number
0000    417 ;               of RCT copies is not zero.  This change causes the class
0000    418 ;               driver to properly record the presence or absence of an RCT.
0000    419 ;               It may break applications which were incorrectly using
0000    420 ;               DEV$M_RCT as a signal that this was/is a MSCP disk.
0000    421 ;
0000    422 ;       V03-200 ROW0247         Ralph O. Weber          2-NOV-1983
0000    423 ;               Correct inverted-sense branches in PACKACK_OFFLINE path added
0000    424 ;               in ROW0231 to switch paths when a IO$_PACKACK fails and there
0000    425 ;               is reason to believe the alternate path will work.
0000    426 ;
0000    427 ;       V03-199 ROW0245         Ralph O. Weber          17-OCT-1983
0000    428 ;               Correct couple of outstanding bugs:
0000    429 ;                - Change DU$IDR to store incomming message size in
0000    430 ;                  CDRP$L_IOST2+2.  This provides the message size to any code
0000    431 ;                  requiring it.  In particular, the INVALID_STS and
0000    432 ;                  RECORD_UNIT_STATUS fixes mentioned below use this feature.
0000    433 ;                - Enhance RECORD_UNIT_STATUS to bugcheck if it produces zero
0000    434 ;                  geometry information.
0000    435 ;                - Fix INVALID_STS to properly place the size of the incomming
0000    436 ;                  MSCP message in R1 before calling ERL$LOG_DMSCP.
0000    437 ;
0000    438 ;       V03-198 ROW0242         Ralph O. Weber          15-OCT-1983
0000    439 ;               Change unit attention processing in DU$IDR to skip altering
0000    440 ;               UCB$M_DU_WAITBMP and UCB$W_RWAITCNT when the CDDB$M_INITING or
0000    441 ;               CDDB$M_RECONNECT is set in CDDB$W_STATUS.  This prevents
0000    442 ;               altering the wait count is such a way that the wait count
0000    443 ;               tests in controller init and reconnection processing fail.
0000    444 ;               Therefore, a spurous disk class driver bugcheck is eliminated.
0000    445 ;
0000    446 ;       V03-197 ROW0231         Ralph O. Weber          1-OCT-1983
0000    447 ;               Fix several bugs:
0000    448 ;               o When a new DDB is created, either in CHAIN_UCB or
0000    449 ;                 CHAIN_DUALPORT_UCB, force the initial UCB links both primary
0000    450 ;                 and secondary to be zero (no chain exists yet).
0000    451 ;               o Change PACKACK function to try secondary path if primary
0000    452 ;                 path ONLINE command fails with "unit unknown or online to
0000    453 ;                 another controller."
0000    454 ;               o Fix PACKACK function to set DEV$M_SWL correctly whether or
0000    455 ;                 not host initiated bad block replacement is in effect.
0000    456 ;               o Correct a couple of typographical errors in comments.
```

```
0000   457 ;
0000   458 ;       V03-196 ROW0229         Ralph O. Weber          24-SEP-1983
0000   459 ;               Make several changes which help SHOW DEVICE display
0000   460 ;               information on dual pathed disks:
0000   461 ;               o Set the DEV$M_2P bit in UCB$L_DEVCHAR2 of all dual pathed
0000   462 ;                 disks.  This allows dual pathed disks to be quickly
0000   463 ;                 identified.
0000   464 ;               o When a local, non-MSCP disk also has a MSCP path, set the
0000   465 ;                 DEV$M_CDP bit only in UCB$L_DEVCHAR2 of the UCB for the MSCP
0000   466 ;                 path.  This allows differentiation between the local-path
0000   467 ;                 and the MSCP-path UCBs.
0000   468 ;               o When a local, non-MSCP disk also has a MSCP path, fillin
0000   469 ;                 UCB$L_DP_DDB in both UCBs.  This allows correct chaining
0000   470 ;                 back to the system block in all cases.
0000   471 ;
0000   472 ;       V03-195 ROW0222         Ralph O. Weber          12-SEP-1983
0000   473 ;               o Change FDT to use EXE$LCLDSKVALID for IO$_PACKACK,
0000   474 ;                 IO$_AVAILABLE, and IO$_UNLOAD processing.
0000   475 ;               o Move IO$_DSE entry higher in FDT to give it faster
0000   476 ;                 processing.
0000   477 ;               o Change base of device dependent UCB to be UCB$K_LCL_DISK.
0000   478 ;               o Move the testing for and setup of a local-dual-path UCB from
0000   479 ;                 INIT_UCB to CHAIN_UCB, where we have a vaild DUDRIVER-
0000   480 ;                 created DDB to test against.
0000   481 ;               o Fix LOOKUP_LOCAL_UCB to use this DDB.
0000   482 ;               o If a local UCB exists for a MSCP served unit, mark the MSCP
0000   483 ;                 served unit offline to make is unusable.  At this point, we
0000   484 ;                 have no plans to allow it to be used, so make the UCB
0000   485 ;                 reflect that fact.
0000   486 ;               o Correct bad stack offset in CHAIN_DUALPORT_UCB.  Also correct
0000   487 ;                 starting DDB address so that a newly created DDB is linked
0000   488 ;                 into the proper place.
0000   489 ;               o Rewrite INIT_CONNDEP_UCB to use only the UCB address input
0000   490 ;                 and to fillin all fields based upon the primary path to the
0000   491 ;                 device.
0000   492 ;               o Move call to INIT_CONNDEP_UCB for available attention
0000   493 ;                 messages (in DU$IDR) to after call to CHAIN_UCB so that UCB
0000   494 ;                 is properly initialized for INIT_CONNDEP_UCB.
0000   495 ;
0000   496 ;       V03-194 ROW0219         Ralph O. Weber           7-SEP-1983
0000   497 ;               Make a couple of fixes for AZTEC.  Correct media-id
0000   498 ;               information in MEDIA_ARRAY to represent RC25 and RCF25.  Fix
0000   499 ;               no-unit-number error logging to "know" about AZTECs.  (This
0000   500 ;               change is ENH and V3.5 BUG.)
0000   501 ;
0000   502 ;       V03-193 ROW0210         Ralph O. Weber          16-AUG-1983
0000   503 ;               Fix several bugs in LOOKUP_LOCAL_UCB.  Add call to
0000   504 ;               IOC$CVTLOGPHY before transfering IRP to local driver at
0000   505 ;               LOCAL_DEVICE.  Change allocation class setup to use VMSD2 as
0000   506 ;               the allocation class for an HSC which has not specified an
0000   507 ;               allocation class.  Prepare to set UCB$L_MAXBCNT, the maximum
0000   508 ;               single transfer byte count, to 2**32-1.  We cannot actually
0000   509 ;               set MAXBCNT now because it breaks MOUNT and INIT.  Add a
0000   510 ;               branch which disables DAP commands but which can easily be
0000   511 ;               patched out once they are working.
0000   512 ;
0000   513 ;       V03-192 ROW0208         Ralph O. Weber          13-AUG-1983
                        Fix IO$_DSE to have an FDT routine which copies P2 into
```

```
0000   514 ;
0000   515 ;               UCB$L_BCNT and copies P3 into IRP$L_MEDIA.  (This change ENH
0000   516 ;               only.)
0000   517 ;
0000   518 ;               Fix literal reference to EMB$C_INVSTS in INVALID_STS.  (This
0000   519 ;               change ENH and V3.5 BUG.)
0000   520 ;
0000   521 ;       V03-191 ROW0205         Ralph O. Weber           8-AUG-1983
0000   522 ;               Add device failover support.  Currently, this support is
0000   523 ;               limited to FILES-11 devices.  This restriction will eventually
0000   524 ;               have to be lifted.  However, work must be done to mount
0000   525 ;               verification before the restriction can be lifted.
0000   526 ;
0000   527 ;               Primary modifications involved in adding failover support are
0000   528 ;               addition of a FAILOVER_DELTA driver parameter to specify the
0000   529 ;               time we wait after a connection fails before attempting to do
0000   530 ;               failover and addition of a call to DUTU$FAILOVER in
0000   531 ;               MAKE_CONNECTION.  It was also necessary to prevent
0000   532 ;               UCB$L_DU_CDDB from getting over written with the secondary
0000   533 ;               CDDB address and to make CDDB$K_PERMCDRP_OFF a global symbol
0000   534 ;               giving the offset from the CDDB base to the permanent CDRP
0000   535 ;               base.
0000   536 ;
0000   537 ;       V03-190 RLRDSE2         Robert L. Rappaport      28-Jul-1982
0000   538 ;               Correct minor bugs in TRANSFER_INVALID_COMMAND that
0000   539 ;               incorrectly reproduced the opcode for IO$_DSE and
0000   540 ;               also incorrectly returned a Success status code.
0000   541 ;
0000   542 ;       V03-189 RLRDSE1         Robert L. Rappaport      26-Jul-1983
0000   543 ;               Add IO$_DSE to FDT tables.
0000   544 ;
0000   545 ;       V03-188 RLRODDBCNT      Robert L. Rappaport      22-Jul-1983
0000   546 ;               Test for odd bcnt in Host Memory Access Errors and
0000   547 ;               return SS$_IVBUFLEN in that case.
0000   548 ;
0000   549 ;       V03-187 RLRMEDIAID      Robert L. Rappaport      15-Jul-1983
0000   550 ;               Set UCB$L_MEDIA_ID in POLL_FOR_UNITS from data in
0000   551 ;               GET UNIT STATUS End Message.
0000   552 ;
0000   553 ;       V03-186 RLRCREDITBUG    Robert L. Rappaport      12-Jul-1983
0000   554 ;               Correct bug that allowed credits to go to zero.  Bug
0000   555 ;               was that we allocated a message buffer (and therefore
0000   556 ;               a credit) in STARTIO before testing the VOL_VALID bit.
0000   557 ;               We then simply deallocated the buffer without
0000   558 ;               disposing of the credit.
0000   559 ;
0000   560 ;       V03-185 RLRINVSTS       Robert L. Rappaport      1-Jul-1983
0000   561 ;               General set of changes to improve reliability.  These
0000   562 ;               include tolerating invalid MSCP status returns and
0000   563 ;               HOST Memory Buffer errors.  Also incorporate logging
0000   564 ;               of invalid ATTN messages, and RE-SYNCH after invalid
0000   565 ;               status returns.
0000   566 ;
0000   567 ;       V03-184 RLRDUALP1       Robert L. Rappaport      23-Jun-1983
0000   568 ;               First set of changes intended to support dual ported
0000   569 ;               devices.  Here we introduce the DU$_CDDB_CHAIN,
0000   570 ;               subroutines SCAN_UCB_CHAIN, and CHAIN_DUALPORT_UCB,
               and introduce the DAPCDRP, (Determine Access Path).
```

```
0000   571 ;
0000   572 ;       V03-183 RLRCREDITa      Robert L. Rappaport     21-Jun-1983
0000   573 ;               Correct bug, discovered by Mike Masters in CX.  Fix
0000   574 ;               is in POST_CDRP, we must have R3=>UCB before calling
0000   575 ;               INIT_MSG_BUF.
0000   576 ;
0000   577 ;       V03-182 RLRDDB1         Robert L. Rappaport     21-Jun-1983
0000   578 ;               Additional change to previous update (RLRDDB) so
0000   579 ;               that the OrigUCB (Boot UCB) remains linked on its
0000   580 ;               DDB so as to enable it to be found by the ASSIGN
0000   581 ;               system service.
0000   582 ;
0000   583 ;       V03-181 RLRRWAITBUG     Robert L. Rappaport     17-Jun-1983
0000   584 ;               Correct bug in Cancel_IO Routine that decremented
0000   585 ;               RWAITCNT twice when we canceled CDRP that was
0000   586 ;               current HIRT owner.
0000   587 ;       V03-180 RLRALCLS2       Robert L. Rappaport     17-Jun-1983
0000   588 ;               Check for valid protocol type in Connect Data Message.
0000   589 ;
0000   590 ;       V03-179 RLRALCLS1       Robert L. Rappaport     3-Jun-1983
0000   591 ;               Correct typo in previous update.
0000   592 ;
0000   593 ;       V03-178 RLRALOCLS       Robert L. Rappaport     26-May-1983
0000   594 ;               Set CDDB$L_ALLOCLS and DBB$L_ALLOCLS from Connect
0000   595 ;               data.
0000   596 ;
0000   597 ;       V03-177 RLRUCBSZ        Robert L. Rappaport     25-May-1983
0000   598 ;               In ASSUME statement that tests size of UCB to that
0000   599 ;               of size of boot UCB (in DEVICEDAT.MAR), increase limit
0000   600 ;               to reflect growing of boot UCB in DEVICEDAT.MAR.
0000   601 ;
0000   602 ;       V03-176 RLRWRCHK        Robert L. Rappaport     19-May-1983
0000   603 ;               Correct error in handling IO$_WRITECHECK requests that
0000   604 ;               only allowed 512 BCNT and references within RCT.
0000   605 ;
0000   606 ;       V03-175 TCM0001         Trudy C. Matthews       4-May-1983
0000   607 ;               Set the DEV$V_CLU (device is available cluster-wide) bit
0000   608 ;               in INIT_UCB if the device is on a multi-host bus (i.e. CI).
0000   609 ;
0000   610 ;       V03-174 RLRNOLOG        Robert L. Rappaport     15-Apr-1983
0000   611 ;               Don't log Available Attention messages.
0000   612 ;
0000   613 ;       V03-173 RLRCANCELg      Robert L. Rappaport     1-Apr-1983
0000   614 ;               Correct bug in Cancel_IO Routine that affects CDRP's
0000   615 ;               on the HIRT Q and the CDRP that owns the HIRT.  Fix
0000   616 ;               is to DEALLOC_MSG_BUF the buffer owned by these CDRP's
0000   617 ;               prior to calling POST_CDRP.
0000   618 ;
0000   619 ;       V03-172 RLRRWCPTRa      Robert L. Rappaport     29-Mar-1983
0000   620 ;               Correct bug in RLRRWCPTR fix.
0000   621 ;
0000   622 ;       V03-171 RLRCANCELf      Robert L. Rappaport     29-Mar-1983
0000   623 ;               Initialize CDRP fields before deciding whether to
0000   624 ;               start this I/O request or whether to Q to mount
0000   625 ;               verification or UCB I/O Q.  This prevents misusing
0000   626 ;               uninitialized fields from being misinterpreted.  Bug
0000   627 ;               arose when an IRP traveled from I/O Queue to Mount
```

```
0000   628 ;               Verification Queue.  This fix supercedes RLRCANCELc.
0000   629 ;
0000   630 ;       V03-170 RLRDDB          Robert L. Rappaport     18-Mar-1983
0000   631 ;               Add support for multiple DDB's per connection.
0000   632 ;
0000   633 ;       V03-169 RLRRWCPTR       Robert L. Rappaport     4-Mar-1983
0000   634 ;               Test for zero UCB$L_RWCPTR in RDTWAIT_DIS_ACT and
0000   635 ;               in RDT_DIS_ACTION.  Such a situation could occur if
0000   636 ;               no RSPID's were available during a re-Connection and
0000   637 ;               if the re-Connection failed and we had to do a
0000   638 ;               re-re-Connection.  Also use Controller timeout for
0000   639 ;               host timeout value for those controllers for which
0000   640 ;               we care to set a host timeout.
0000   641 ;
0000   642 ;       V03-168 RLRIHIRT        Robert L. Rappaport     3-Mar-1983
0000   643 ;               Eliminate window in Controller initialization wherein
0000   644 ;               a timeout on the Set Controller Characteristics command
0000   645 ;               will cause us to neglect to call INIT_HIRT.
0000   646 ;
0000   647 ;       V03-167 RLRDSE          Robert L. Rappaport     28-Feb-1983
0000   648 ;               Add support for IO$_DSE.
0000   649 ;
0000   650 ;       V03-166 RLRTMUCB        Robert L. Rappaport     25-Feb-1983
0000   651 ;               Revamp Template UCB so as to be automatically compliant
0000   652 ;               with new UCB additions.
0000   653 ;
0000   654 ;       V03-165 RLRRDRX         Robert L. Rappaport     9-Feb-1983
0000   655 ;               Add RD/RX definitions in tables.  Also add patchable
0000   656 ;               extensions to tables.
0000   657 ;
0000   658 ;       V03-164 RLRCREDIT       Robert L. Rappaport     18-Jan-1983
0000   659 ;               Correct esoteric bug that could lose credits when do
0000   660 ;               Cancel I/O.  Basically, CDRP's that are CANCELED
0000   661 ;               (rather than ABORTED), that have already allocated
0000   662 ;               Message Buffers have no way of deallocating the
0000   663 ;               associated CREDIT.  The only solution is to have them
0000   664 ;               send a NOP message.
0000   665 ;               Also use CDDB$W_CNTRLTMO rather than INIT_IMMED_DELTA in
0000   666 ;               BRING_UNIT_ONLINE.
0000   667 ;               Also in REPLACE_LBN, after writting the TEST_PATTERN,
0000   668 ;               explicitly UNMAP and then reMAP before reading it back in.
0000   669 ;               Otherwise on 780's we will have buffered datapath problems.
0000   670 ;
0000   671 ;       V03-163 RLRONLIN        Robert L. Rappaport     28-Oct-1982
0000   672 ;               Add possible status returns from ONLIN command so that
0000   673 ;               unformatted disk does not cause crash.
0000   674 ;
0000   675 ;       V03-162 RLRCLINIT       Robert L. Rappaport     21-Oct-1982
0000   676 ;               Clear CDDB$M_INITING in Reconnection.
0000   677 ;
0000   678 ;       V03-161 RLRWAITBMP      Robert L. Rappaport     13-Oct-1982
0000   679 ;               Add UCB$M_DU_WAITBMP bit in UCB$W_DU_DEVSTS, which is
0000   680 ;               set iff the UCB$W_RWAITCNT has been bumped due to either
0000   681 ;               initialization or re-connection.  This allows to bump
0000   682 ;               (or not bump) the RWAITCNT in re-connection on a unit
0000   683 ;               by unit basis instead of on a connection basis.  This
0000   684 ;               prevents situations where during init or re-connect,
```

```
0000   685 ;
0000   686 ;               certain timeouts result in some but not all of the units
0000   687 ;               having their RWAITCNTs decremented.  Also increment
0000   688 ;               CDDB$W_RSTRTCNT each time we enter re-connection code.
0000   689 ;
0000   690 ;       V03-160 RLRBUNOL         Robert L. Rappaport     11-Oct-1982
0000   691 ;               Correct problem in Bring UNit OnLine, wherein we
0000   692 ;               might use the connection permanent CDRP to do
0000   693 ;               ONLINE_COMPLETE, and therefore bump the RWAITCNT
0000   694 ;               of this UCB.
0000   695 ;
0000   696 ;       V03-159 RLRPOLL          Robert L. Rappaport     8-Oct-1982
0000   697 ;               Make the polling of controller for units a callable
0000   698 ;               function rather than in line code executed at initialization
0000   699 ;               time only.  Also call it after re-connection.
0000   700 ;
0000   701 ;       V03-158 RLRMRESETb       Robert L. Rappaport     8-Oct-1982
0000   702 ;               Correct bug introduced in V03-055 which cleared R3=>CDDB
0000   703 ;               and then tested CDDB$W_STATUS bit.
0000   704 ;
0000   705 ;       V03-157 RLRIATTN         Robert L. Rappaport     5-Oct-1982
0000   706 ;               In ATTN_MSG ignore attentions during initialization.
0000   707 ;
0000   708 ;       V03-156 RLRMERSETa       Robert L. Rappaport     5-Oct-1982
0000   709 ;               Change calls to MRESET and MSTART to pass PB$B_RSTATION
0000   710 ;               instead of remote System ID.
0000   711 ;
0000   712 ;       V03-155 RLRRESYN         Robert L. Rappaport     4-Oct-1982
0000   713 ;               Let DU$RESYNCH setup R4 => PDT.
0000   714 ;
0000   715 ;       V03-154 RLRATTN          Robert L. Rappaport     29-Sep-1982
0000   716 ;               Correct error in Input Dispatcher Routine, wherein after
0000   717 ;               receipt and handling of an Attention message, R3 was no
0000   718 ;               longer pointing to the CDT.
0000   719 ;
0000   720 ;       V03-153 RLRUNLCK         Robert L. Rappaport     24-Sep-1982
0000   721 ;               Correct bug in UNLOCK_HIRT that attempts to return
0000   722 ;               to caller's caller when original CDRP has been
0000   723 ;               canceled.  Problem is that caller may have left data
0000   724 ;               on stack.  Solution is to have caller to UNLOCK_HIRT
0000   725 ;               check for zero in R5 and cleanup its own stack before
0000   726 ;               returning to its caller.
0000   727 ;               Note- version numbers incremented by 100 to distinguish
0000   728 ;               from .BUG version.
0000   729 ;
0000   730 ;       V03-052 ROW0128          Ralph O. Weber          22-SEP-1982
0000   731 ;               Modify template UCB to conform with new UCB structure.
0000   732 ;               WARNING:  this makes the enhancement version of this driver
0000   733 ;               completely incompatible with V3.0.
0000   734 ;
0000   735 ;       V03-051 RLRPURGE         Robert L. Rappaport     17-Sep-1982
0000   736 ;               In WRITE_RCT_BLOCK and READ_RCT_BLOCK, do an explicit
0000   737 ;               MAP and UNMAP before each read or write so as to
0000   738 ;               force a needed datapath purge on UDA's on 780's.
0000   739 ;
0000   740 ;       V03-050 RLRTIMOUT        Robert L. Rappaport     10-Sep-1982
0000   741 ;               Allow for timing out of MAKE_CONNECTION.  That is,
                        if the number of seconds specified in SGN$GL_VMSD3
```

```
0000   742 ;           has past since we began to try to CONNECT, then
0000   743 ;           terminate all pending I/O with SS$_CTRLERR.  If this
0000   744 ;           SYSGEN parameter is zero, then such timeouts are
0000   745 ;           disabled and we do not terminate any I/O but rather
0000   746 ;           keep them pending until the CONNECTION is established.
0000   747 ;
0000   748 ;       V03-049 RLRMRESET       Robert L. Rappaport     8-Sep-1982
0000   749 ;           Changed order of MRESET and DISCONNECT so that an
0000   750 ;           explicit DISCONNECT is NOT done if we decide to MRESET.
0000   751 ;           We instead rely on the port drivers to call our error
0000   752 ;           entrypoint in order for the DISCONNECT to be done.
0000   753 ;           Therefore after MRESET we now RSB.
0000   754 ;
0000   755 ;       V03-048 RLRCANCELe      Robert L. Rappaport     27-Aug-1982
0000   756 ;           Modify TST_CANCEL_CDRP so as to NOT test for virtual
0000   757 ;           I/O requests.
0000   758 ;
0000   759 ;       V03-047 RLRCANCELd      Robert L. Rappaport     24-Aug-1982
0000   760 ;           Correct bug in CANCEL I/O wherein two cancels, in quick
0000   761 ;           succession, for a channel that had a CDRP that was doing
0000   762 ;           a replacement caused a crash.  Fix is to check for zero
0000   763 ;           in HIRT$L_SAVCDRP of a busy HIRT.  This could occur if
0000   764 ;           we had already posted the owner but allowed the replacement
0000   765 ;           to run to completion.
0000   766 ;
0000   767 ;       V03-046 RLR0046         Robert L. Rappaport     22-July-1982
0000   768 ;           Changed references to UCB$L_DU_MEDIAID to UCB$L_MEDIA_ID.
0000   769 ;           Also changed references to DU_FORK_IPL to IPL$_SCS.
0000   770 ;
0000   771 ;       V03-045 RLRCANCELc      Robert L. Rappaport     7-July-1982
0000   772 ;           Correct bug in CANCEL I/O wherein we take an IRP off
0000   773 ;           the Mount Verification Q and call POST_IRP and the
0000   774 ;           IRP had never had its CDRP portion initialized.
0000   775 ;           Fix entails clearing CDRP$W_CDRPSIZE when INSQUEing
0000   776 ;           an uninitialized IRP onto Mount verification Q and
0000   777 ;           testing for this condition in POST_CDRP and then NOT
0000   778 ;           calling DEALLOC_RESOURCES for such a CDRP.
0000   779 ;
0000   780 ;       V03-044 RLRCANCELb      Robert L. Rappaport     7-July-1982
0000   781 ;           Correct re-CONNECT bug that left UCB$W_RWAITCNT zero.
0000   782 ;           Added new bit, CDRP$M_CANCELED, in CDRP$W_STS that
0000   783 ;           distinguishes all CDRP's that are in the process of
0000   784 ;           being canceled.  These CDRP's are necessarily queued
0000   785 ;           on either the CDRP$L_IOQFL or CDRP$L_ABCNT of some
0000   786 ;           Cancel CDRP.  In RDT_DIS_ACTION we check for this bit.
0000   787 ;
0000   788 ;       V03-043 RLRDEFS         Robert L. Rappaport     29-June-1982
0000   789 ;           Add $DCDEF, $IPLDEF, $PCBDEF, $PRDEF, $SSDEF, and $VADEF.
0000   790 ;
0000   791 ;       V03-042 RLRCANCELa      Robert L. Rappaport     24-June-1982
0000   792 ;           Correct minor bugs in CANCEL_IO.
0000   793 ;
0000   794 ;       V03-041 RLRECO13        Robert L. Rappaport     16-June-1982
0000   795 ;           Allow for long (not infinite) host timeouts on UDA.
0000   796 ;
0000   797 ;       V03-040 RLRCANCEL       Robert L. Rappaport     10-June-1982
0000   798 ;           Add CANCEL_IO functionality.
```

```
0000    799 ;
0000    800 ;       V03-039 RLREC012        Robert L. Rappaport     7-June-1982
0000    801 ;               Eliminate extra UCB created by SYSGEN.  This entails
0000    802 ;               adding a bogus UNITINIT routine to be called that
0000    803 ;               only looks out for bogus UCB's and eliminates them.
0000    804 ;
0000    805 ;       V03-038 RLREC011        Robert L. Rappaport     4-June-1982
0000    806 ;               Two corrections:
0000    807 ;               1. BRW to DU$RE_SYNCH as a result of an INIT_TIMEOUT.
0000    808 ;               2. Return proper success status if disk write-locked
0000    809 ;                       on a PACKACK operation.  We were returning
0000    810 ;                       SS$_WRITLCK.  ALso if ONLINE succeeds but
0000    811 ;                       ONLINE_COMPLETE fails due to something other
0000    812 ;                       than write-lock, make disk AVAILABLE.
0000    813 ;
0000    814 ;       V03-037 RLREC010        Robert L. Rappaport     3-June-1982
0000    815 ;               Correct error introduced in RLREC007 wherein MOVZWL
0000    816 ;               to HIRT$W_IOST also cleared next word (HIRT$W_STS).
0000    817 ;
0000    818 ;       V03-036 RLREC009        Robert L. Rappaport     2-June-1982
0000    819 ;               Correct two problems:
0000    820 ;               1. In INSERT_RSTRTQ, do NOT (recursively) insert the
0000    821 ;               HIRT$L_SAVDCDRP if this CDRP is the connection permanent
0000    822 ;               CDRP.
0000    823 ;               2. Lengthen the connection permanent CDRP (appended to CDDB)
0000    824 ;               to the length of a normal CDRP (including an IRP prefix).
0000    825 ;               Also insure that this CDRP has a valid value in CDRP$L_UCB.
0000    826 ;               This is necessary to allow proper deallocation of map resources
0000    827 ;               that may have been put here due to re-CONNECT from HIRTCDRP
0000    828 ;               when we are running on the UDA port.  This port driver needs
0000    829 ;               to have access to a UCB in order to find the CRB.
0000    830 ;
0000    831 ;       V03-035 RLREC008        Robert L. Rappaport     26-May-1982
0000    832 ;               Correct shift of FORCEDERROR subcode in
0000    833 ;               TRANSFER_DATA_ERROR.
0000    834 ;
0000    835 ;       V03-034 RLREC007        Robert L. Rappaport     26-May-1982
0000    836 ;               Allow write locked disks to be MOUNTed.
0000    837 ;
0000    838 ;       V03-033 RLREC006        Robert L. Rappaport     25-May-1982
0000    839 ;               Correct bug introduced in ECO01, wherein resetting
0000    840 ;               of Timeout Routine was done prematurely.
0000    841 ;
0000    842 ;       V03-032 RLREC005        Robert L. Rappaport     20-May-1982
0000    843 ;               To protect against a sick controller, do UNMAPs for
0000    844 ;               collected CDRP's only after controller reset and
0000    845 ;               DISCONNECT done in re-CONNECT logic.  This is
0000    846 ;               done by having INSERT_RSTRTQ now call DEALO_MSG_RSPID,
0000    847 ;               which only deallocates these resources, and then
0000    848 ;               calling DEALLOC_RESOURCES for all collected CDRP's.
0000    849 ;               Also any mapping resources allocated by the HIRTCDRP
0000    850 ;               for this connection are copied to the connection
0000    851 ;               permanent CDRP which also has DEALLOC_RESOURCES called
0000    852 ;               for it at this same time.
0000    853 ;
0000    854 ;       V03-031 RLREC004        Robert L. Rappaport     19-May-1982
0000    855 ;               In UNLOCK_HIRT, after call to GRANT_HIRT, resume thread
```

```
0000    856 ;                   from HIRT$L_SAVDCDRP rather than from R5 => ReplaceCDRP
0000    857 ;                   returned by GRANT_HIRT.
0000    858 ;
0000    859 ;       V03-030 RLREC003        Robert L. Rappaport     19-May-1982
0000    860 ;                   Return SS$_DRVERR to users of PACKACK and transfer
0000    861 ;                   functions if MSCP$K_ST_MEDOFL is returned with
0000    862 ;                   MSCP$V_SC_INOPR subcode also set.
0000    863 ;
0000    864 ;       V03-029 RLREC002        Robert L. Rappaport     13-May-1982
0000    865 ;                   1. Refresh R2 => End Message after ONLINE_COMPLETE in
0000    866 ;                         BRING_UNIT_ONLINE.
0000    867 ;                   2. Call SCS$UNSTALL_UCB in UNLOCK_HIRT.
0000    868 ;                   3. Refresh R2 => End Message after UNMAP in STEP7 of
0000    869 ;                         REPLACE_LBN.
0000    870 ;                   This change tracks patch to DUDRIVER in V3.1
0000    871 ;
0000    872 ;       V03-028 RLREC001        Robert L. Rappaport     26-Apr-1982
0000    873 ;                   In UNLOCK_HIRT,  do not reassign RSPID to user CDRP if
0000    874 ;                   RSPID has been deallocated due to re-CONNECT. Also
0000    875 ;                   re-establish DU$TMR as timeout routine after calling
0000    876 ;                   MAKE_CONNECTION in re-CONNECTION.
0000    877 ;                   This change tracks patch to DUDRIVER in V3.0
0000    878 ;
0000    879 ;
0000    880 ; MACRO LIBRARY CALLS
0000    881 ;
0000    882
0000    883         $CDDBDEF                        ;Define CDDB offsets
0000    884         $CDRPDEF                        ;Define CDRP offsets
0000    885         $CDTDEF                         ;Define CDT offsets
0000    886         $CRBDEF                         ;Define CRB offsets
0000    887         $DCDEF                          ;Define Device Classes and Types
0000    888         $DDBDEF                         ;Define DDB offsets
0000    889         $DEVDEF                         ;Define DEVICE CHARACTERISTICS bits
0000    890         $DPTDEF                         ;Define DPT offsets
0000    891         $DYNDEF                         ;Define DYN symbols
0000    892         $EMBLTDEF                       ;Define EMB Log Message Types
0000    893         $FKBDEF                         ;Define FKB offsets
0000    894         $IDBDEF                         ;Define IDB offsets
0000    895         $IODEF                          ;Define I/O FUNCTION codes
0000    896         $IPLDEF                         ;Define IPL levels
0000    897         $IRPDEF                         ;Define IRP offsets
0000    898         $MSCPDEF                        ;Define MSCP packet offsets
0000    899         $MSLGDEF                        ;Define MSCP Error Log offsets
0000    900         $MTXDEF                         ;Define MUTEX offsets
0000    901         $ORBDEF                         ;Define ORB offsets
0000    902         $PBDEF                          ;Define Path Block offsets
0000    903         $PCBDEF                         ;Define PCB offsets
0000    904         $PDTDEF                         ;Define PDT offsets
0000    905         $PRDEF                          ;Define Processor Registers
0000    906         $RCTDEF                         ;Define RCT offsets
0000    907         $RDDEF                          ;Define RDE offsets
0000    908         $RDTDEF                         ;Define RDT offsets
0000    909         $SBDEF                          ;Define System Block Offsets
0000    910         $SCSCMGDEF                      ;Define SCS Connect Message offsets
0000    911         $SSDEF                          ;Define System Status values
0000    912         $UCBDEF                         ;Define UCB offsets
```

```
                     0000   913          $VADEF                          ;Define Virtual Address offsets
                     0000   914          $VECDEF                         ;Define INTERRUPT DISPATCH VECTOR offsets
                     0000   915          $WCBDEF                         ;Define WCB offsets
                     0000   916
                     0000   917
                     0000   918          $DUTUDEF                        ;Define common class driver CDDB
                     0000   919                                          ; extensions and other common symbols
                     0000   920
                     0000   921
                     0000   922 ; Constants
                     0000   923
00000001             0000   924 ALLOC_DELTA=1                           ; Number of seconds to wait to retry pool
                     0000   925                                          ;  allocation that failed.
0000001E             0000   926 INIT_IMMED_DELTA=30                     ; During Controller Initialization, the
                     0000   927                                          ;  timeout DELTA for immediate MSCP commands.
0000000A             0000   928 CONNECT_DELTA=10                       ; During Controller Initialization, the
                     0000   929                                          ;  time interval for retrying failed
                     0000   930                                          ;  CONNECT attempts.
0000001E             0000   931 HOST_TIMEOUT=30                        ; Host timeout value.
                     0000   932
00000001             0000   933 DISCONNECT_REASON=1
0000000A             0000   934 INITIAL_CREDIT=10
00000002             0000   935 INITIAL_DG_COUNT=2
00000002             0000   936 MAX_RETRY=2
00000002             0000   937 MIN_SEND_CREDIT=2
                     0000   938
                     0000   939 ;
                     0000   940 ; ARGUMENT LIST OFFSET DEFINITIONS
                     0000   941 ;
                     0000   942
00000000             0000   943 P1=0                                    ;FIRST FUNCTION DEPENDENT PARAMETER
00000004             0000   944 P2=4                                    ;SECOND FUNCTION DEPENDENT PARAMETER
00000008             0000   945 P3=8                                    ;THIRD FUNCTION DEPENDENT PARAMETER
0000000C             0000   946 P4=12                                   ;FOURTH FUNCTION DEPENDENT PARAMETER
00000010             0000   947 P5=16                                   ;FIFTH FUNCTION DEPENDENT PARAMETER
00000014             0000   948 P6=20                                   ;SIXTH FUNCTION DEPENDENT PARAMETER
```

```
0000   950          .SBTTL  MACRO DEFINITIONS
0000   951
0000   952 ;
0000   953 ; Expanded opcode macros - Branch word conditional psuedo opcodes.
0000   954 ;
0000   955
0000   956 ;
0000   957 ; BWNEQ - Branch (word offset) not equal
0000   958 ;
0000   959
0000   960          .MACRO  BWNEQ   DEST,?L1
0000   961          .SHOW
0000   962          BEQL    L1                      ; Branch around if NOT NEQ.
0000   963          BRW     DEST                    ; Branch to destination if NEQ.
0000   964 L1:                                     ; Around.
0000   965          .NOSHOW
0000   966          .ENDM   BWNEQ
0000   967
0000   968 ;
0000   969 ; BWEQL - Branch (word offset) equal
0000   970 ;
0000   971
0000   972          .MACRO  BWEQL   DEST,?L1
0000   973          .SHOW
0000   974          BNEQ    L1                      ; Branch around if NOT EQL.
0000   975          BRW     DEST                    ; Branch to destination if EQL.
0000   976 L1:                                     ; Around.
0000   977          .NOSHOW
0000   978          .ENDM   BWEQL
0000   979
0000   980 ;
0000   981 ; BWBS - Branch (word offset) bit set.
0000   982 ;
0000   983
0000   984          .MACRO  BWBS    BIT,FIELD,DEST,?L1
0000   985          .SHOW
0000   986          BBC     BIT,FIELD,L1            ; Branch around if bit NOT set.
0000   987          BRW     DEST                    ; Branch to destination if bit set.
0000   988 L1:                                     ; Around.
0000   989          .NOSHOW
0000   990          .ENDM   BWBS
0000   991
0000   992 ;
0000   993 ; BWBC - Branch (word offset) bit clear.
0000   994 ;
0000   995
0000   996          .MACRO  BWBC    BIT,FIELD,DEST,?L1
0000   997          .SHOW
0000   998          BBS     BIT,FIELD,L1            ; Branch around if bit NOT clear.
0000   999          BRW     DEST                    ; Branch to destination if bit clear.
0000  1000 L1:                                     ; Around.
0000  1001          .NOSHOW
0000  1002          .ENDM   BWBC
```

```
0000  1004          .SBTTL  ASSUMES
0000  1005
0000  1006 ; The following set of ASSUME statements will all be true as long as
0000  1007 ;       the IRP and CDRP definitions remain consistent.
0000  1008
0000  1009          ASSUME  CDRP$L_IOQFL-CDRP$L_IOQFL       EQ      IRP$L_IOQFL
0000  1010          ASSUME  CDRP$L_IOQBL-CDRP$L_IOQFL       EQ      IRP$L_IOQBL
0000  1011          ASSUME  CDRP$W_IRP_SIZE-CDRP$L_IOQFL   EQ      IRP$W_SIZE
0000  1012          ASSUME  CDRP$B_IRP_TYPE-CDRP$L_IOQFL   EQ      IRP$B_TYPE
0000  1013          ASSUME  CDRP$B_RMOD-CDRP$L_IOQFL       EQ      IRP$B_RMOD
0000  1014          ASSUME  CDRP$L_PID-CDRP$L_IOQFL        EQ      IRP$L_PID
0000  1015          ASSUME  CDRP$L_AST-CDRP$L_IOQFL        EQ      IRP$L_AST
0000  1016          ASSUME  CDRP$L_ASTPRM-CDRP$L_IOQFL     EQ      IRP$L_ASTPRM
0000  1017          ASSUME  CDRP$L_WIND-CDRP$L_IOQFL       EQ      IRP$L_WIND
0000  1018          ASSUME  CDRP$L_UCB-CDRP$L_IOQFL        EQ      IRP$L_UCB
0000  1019          ASSUME  CDRP$W_FUNC-CDRP$L_IOQFL       EQ      IRP$W_FUNC
0000  1020          ASSUME  CDRP$B_EFN-CDRP$L_IOQFL        EQ      IRP$B_EFN
0000  1021          ASSUME  CDRP$B_PRI-CDRP$L_IOQFL        EQ      IRP$B_PRI
0000  1022          ASSUME  CDRP$L_IOSB-CDRP$L_IOQFL       EQ      IRP$L_IOSB
0000  1023          ASSUME  CDRP$W_CHAN-CDRP$L_IOQFL       EQ      IRP$W_CHAN
0000  1024          ASSUME  CDRP$W_STS-CDRP$L_IOQFL        EQ      IRP$W_STS
0000  1025          ASSUME  CDRP$L_SVAPTE-CDRP$L_IOQFL     EQ      IRP$L_SVAPTE
0000  1026          ASSUME  CDRP$W_BOFF-CDRP$L_IOQFL       EQ      IRP$W_BOFF
0000  1027          ASSUME  CDRP$L_BCNT-CDRP$L_IOQFL       EQ      IRP$L_BCNT
0000  1028          ASSUME  CDRP$W_BCNT-CDRP$L_IOQFL       EQ      IRP$W_BCNT
0000  1029          ASSUME  CDRP$L_IOST1-CDRP$L_IOQFL      EQ      IRP$L_IOST1
0000  1030          ASSUME  CDRP$L_MEDIA-CDRP$L_IOQFL      EQ      IRP$L_MEDIA
0000  1031          ASSUME  CDRP$L_IOST2-CDRP$L_IOQFL      EQ      IRP$L_IOST2
0000  1032          ASSUME  CDRP$L_TT_TERM-CDRP$L_IOQFL    EQ      IRP$L_TT_TERM
0000  1033          ASSUME  CDRP$B_CARCON-CDRP$L_IOQFL     EQ      IRP$B_CARCON
0000  1034          ASSUME  CDRP$Q_NT_PRVMSK-CDRP$L_IOQFL  EQ      IRP$Q_NT_PRVMSK
0000  1035          ASSUME  CDRP$L_ABCNT-CDRP$L_IOQFL      EQ      IRP$L_ABCNT
0000  1036          ASSUME  CDRP$W_ABCNT-CDRP$L_IOQFL      EQ      IRP$W_ABCNT
0000  1037          ASSUME  CDRP$L_OBCNT-CDRP$L_IOQFL      EQ      IRP$L_OBCNT
0000  1038          ASSUME  CDRP$W_OBCNT-CDRP$L_IOQFL      EQ      IRP$W_OBCNT
0000  1039          ASSUME  CDRP$L_SEGVBN-CDRP$L_IOQFL     EQ      IRP$L_SEGVBN
0000  1040          ASSUME  CDRP$L_JNL_SEQNO-CDRP$L_IOQFL  EQ      IRP$L_JNL_SEQNO
0000  1041          ASSUME  CDRP$L_DIAGBUF-CDRP$L_IOQFL    EQ      IRP$L_DIAGBUF
0000  1042          ASSUME  CDRP$L_SEQNUM-CDRP$L_IOQFL     EQ      IRP$L_SEQNUM
0000  1043          ASSUME  CDRP$L_EXTEND-CDRP$L_IOQFL     EQ      IRP$L_EXTEND
0000  1044          ASSUME  CDRP$L_ARB-CDRP$L_IOQFL        EQ      IRP$L_ARB
```

```
                        0000  1046          .SBTTL  DISK CLASS DRIVER DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS
                        0000  1047
                        0000  1048          $DEFINI UCB,GLOBAL                  ; The GLOBAL is for DUMIRT.
                        0000  1049
                        0000  1050
000000EC                0000  1051  . = UCB$K_MSCP_DISK_LENGTH
                        00EC  1052  $DEF     UCB$L_DU_VOLSER .BLKL   1           ; Volume Serial number as returned
                        00F0  1053                                              ;  in ONLINE end packet.
                        00F0  1054
                        00F0  1055  $DEF     UCB$L_DU_USIZE  .BLKL   1           ; Size of user visible area of unit
                        00F4  1056                                              ;  in logical blocks.
                        00F4  1057  $DEF     UCB$L_DU_TOTSZ  .BLKL   1           ; Size of unit including RCT area in
                        00F8  1058                                              ;  logical blocks.
                        00F8  1059  $DEF     UCB$W_DU_RCTSIZE
000000FA                00F8  1060                          .BLKW   1           ; Size of the RCT in blocks.
                        00FA  1061  $DEF     UCB$B_DU_RCTCPYS
000000FB                00FA  1062                          .BLKB   1           ; Number of RCT copies on the unit.
                        00FB  1063  $DEF     UCB$B_DU_RBNPTRK
000000FC                00FB  1064                          .BLKB   1           ; RBN's per track.
                        00FC  1065  $DEF     UCB$W_DU_LBNPTRK
000000FE                00FC  1066                          .BLKW   1           ; LBN's per track.
                        00FE  1067  $DEF     UCB$W_DU_TRKPGRP
00000100                00FE  1068                          .BLKW   1           ; Tracks per group.
                        0100  1069  $DEF     UCB$W_DU_GRPPCYL
00000102                0100  1070                          .BLKW   1           ; Groups per cylinder.
                        0102  1071
00000102                0102  1072  UCB$K_DU_LENGTH=.
                        0102  1073
                        0102  1074          $DEFEND UCB
                        0000  1075
                        0000  1076          ASSUME  UCB$K_DU_LENGTH         LE      UCB$K_LENGTH+160
                        0000  1077
                        0000  1078  ;
                        0000  1079  ; If the preceeding ASSUME macro breaks it is a signal that the Disk Class
                        0000  1080  ;       Driver UCB has grown larger than the space allocated to the Boot
                        0000  1081  ;       Device UCB in [SYS.SRC]DEVICEDAT.MAR.  The Boot Device UCB allocated
                        0000  1082  ;       in DEVICEDAT is 40 longwords longer than the nominal UCB (whose length
                        0000  1083  ;       is known symbolically as UCB$K_LENGTH).  This additional length is
                        0000  1084  ;       meant to accomodate all possible idiosyncracies that individual disk
                        0000  1085  ;       drivers might have.  To correct such an overgrowth in UCB size entails
                        0000  1086  ;       enlarging the expansion area in the Boot Device UCB (beyond the 40
                        0000  1087  ;       longwords) and the relevant correction to the above ASSUME macro.
                        0000  1088  ;
                        0000  1089
                        0000  1090
                        0000  1091          .SBTTL  Allocate Space for Template UCB
                        0000  1092
                        0000  1093  ; Allocate zeroed space for template UCB and ORB.
                        0000  1094
                        0000  1095          INIT_UCB size=UCB$K_DU_LENGTH
                        0000  1096          INIT_ORB size=ORB$C_LENGTH
```

```
                        .SBTTL  DRIVER PROLOGUE AND DISPATCH TABLES (and UCB Initialization)
                ;
                ; LOCAL DATA
                ;
                ; DRIVER PROLOGUE TABLE
                ;

                DPTAB   -                               ;DEFINE DRIVER PROLOGUE TABLE
                        END=DUTU$END,-                  ; End of driver
                        ADAPTER=NULL,-                  ; No Adapter
                        FLAGS=<DPT$M_SCS -              ; Driver requires that SCS be loaded
                              !DPT$M_NOUNLOAD>,-        ; Driver cannot be reloaded
                        UCBSIZE=UCB$K_DU_LENGTH,-       ;Sysgen insists on making a UCB
                        MAXUNITS=1,-                    ;Sysgen insists on making a UCB
                        NAME=DUDRIVER                   ; Driver name
                DPT_STORE INIT                          ; Control block init values
                DPT_STORE DDB,DDB$L_ACPD,L,<^A\F11\>    ; Default ACP name
                DPT_STORE DDB,DDB$L_ACPD+3,B,1          ; ACP class

                ; The following UCB initialization requests alter the template UCB
                ; as well as producing equivalent DPT_STORE entries.  Thus both
                ; structures reflect the required initial UCB state and the UCBs
                ; initially processed by this driver are identical whether they are
                ; produced by SYSGEN or by IOC$COPY_UCB.

                INIT_UCB        W_SIZE,WORD,UCB$K_DU_LENGTH
                INIT_UCB        B_TYPE,BYTE,DYN$C_UCB
                INIT_UCB        B_FIPL,BYTE,IPL$_SCS
                INIT_UCB        L_DEVCHAR,LONG,<<DEV$M_FOD!-
                                                DEV$M_DIR!-
                                                DEV$M_AVL!-
                                                DEV$M_ELG!-
                                                DEV$M_SHR!-
                                                DEV$M_IDV!-
                                                DEV$M_ODV!-
                                                DEV$M_RND>>
                INIT_UCB        L_DEVCHAR2,LONG,<<DEV$M_CLU!-
                                                  DEV$M_MSCP!-
                                                  DEV$M_NNM>>
                INIT_UCB        B_DEVCLASS,BYTE,DC$_DISK
                INIT_UCB        W_DEVBUFSIZ,WORD,512
                INIT_UCB        W_RWAITCNT,WORD,1
                INIT_UCB        B_DIPL,BYTE,IPL$_SCS
                INIT_UCB        W_DEVSTS,WORD, <<UCB$M_NOCNVRT -
                                                !UCB$M_MSCP_INITING -
                                                !UCB$M_MSCP_WAITBMP>>

                ; Initially the disk is made to look huge.  This prevents comparison
                ; errors and the MSCP server will catch any real problems.  Once the
                ; correct size is known or it has been determined that the size need
                ; not be known immediately, this value will be reduced.
                INIT_UCB        L_MAXBLOCK,LONG,<<^x7F000000>>

                ; The following ORB initialization requests alter the template ORB
                ; as well as producing equivalent DPT_STORE entries.  Thus both
                ; structures reflect the required initial ORB state and the ORBs
```

```
                        ; initially processed by this driver are identical whether they are
                        ; produced by SYSGEN or by IOC$COPY_UCB.

                        INIT_ORB        W_SIZE,WORD,ORB$C_LENGTH
                        INIT_ORB        B_TYPE,BYTE,DYN$C_ORB
                        INIT_ORB        B_FLAGS,BYTE,<< -
                                            ORB$M_PROT_16>>     ; SOGW protection word
                        INIT_ORB        W_PROT,WORD,0           ; default protection
                        INIT_ORB        L_OWNER,LONG,0          ; no owner as yet

                        DPT_STORE REINIT                ; Control block re-initialization values

                        ; N.B. Causing the following values to be setup during re-initializa-
                        ; tion is not significant because this driver cannot be reloaded.
                        ; However, were the driver to be reloadable the following values would
                        ; need to be re-initialized upon each driver reload.

                        DPT_STORE CRB, -                        ; Controller init routine.
                              CRB$L_INTD+VEC$L_INITIAL,D,DU_CONTROLLER_INIT
                        DPT_STORE DDB,DDB$L_DDT,D,DU$DDT        ; DDT address.

                        DPT_STORE END

                        ;
                        ; DRIVER DISPATCH TABLE
                        ;

                        DDTAB   DEVNAM=DU,-             ;DRIVER DISPATCH TABLE
                                START=DU_STARTIO,-      ;START I/O OPERATION
                                FUNCTB=DU_FUNCTABLE,-   ;FUNCTION DECISION TABLE
                                CANCEL=DUTU$CANCEL,-    ;CANCEL I/O ENTRY POINT
                                DIAGBF=MSCP$K_MXCMDLEN+MSCP$K_LEN+20,-  ; DIAG BUFF SIZE
                                UNITINIT=DUTU$UNITINIT,-; Unit initialization routine.
                                MNTVER=DU$MOUNTVER, -   ; Mount Verification routine.
                                UNSOLIC=DU_REVALIDATE_DISKS ; Quorum lost processing.
                                                        ; Use of the unsolicited interrupt
                                                        ; entry as defined above is a temporary
                                                        ; measure.  It will be changed when
                                                        ; a correct DDT entry can be defined.
```

```
                        .SBTTL  DISK CLASS DRIVER FUNCTION DECISION TABLE
                ;+
                ; DISK CLASS DRIVER FUNCTION DECISION TABLE
                ;-

                DU_FUNCTABLE:                                   ;Function Decision Table
                        FUNCTAB ,-                              ; LEGAL FUNCTIONS
                                <NOP,-                          ;  No operation
                                UNLOAD,-                        ;  Unload (make available + spindown)
                                AVAILABLE,-                     ;  Available (no spindown)
                                PACKACK,-                       ;  Pack Acknowledge
                                SENSECHAR,-                     ;  Sense Characteristics
                                SETCHAR,-                       ;  Set Characteristics
                                SENSEMODE,-                     ;  Sense Mode
                                SETMODE,-                       ;  Set Mode
                                WRITECHECK,-                    ;  Write Check
                                READPBLK,-                      ;  Read  PHYSICAL Block
                                READLBLK,-                      ;  Read  LOGICAL  Block
                                WRITELBLK,-                     ;  Write LOGICAL  Block
                                READVBLK,-                      ;  Read  VIRTUAL  Block
                                WRITEVBLK,-                     ;  Write VIRTUAL  Block
                                DSE,-                           ;  Data Security Erase
                                ACCESS,-                        ;  Access file and/or find directory entry
                                ACPCONTROL,-                    ;  ACP Control Function
                                CREATE,-                        ;  Create file and/or create directory entry
                                DEACCESS,-                      ;  Deaccess file
                                DELETE,-                        ;  Delete file and/or directory entry
                                MODIFY,-                        ;  Modify file attributes
                                MOUNT>                          ;  Mount volume
                        FUNCTAB ,-                              ; BUFFERED I/O FUNCTIONS
                                <NOP,-                          ;  No Operation
                                 UNLOAD,-                       ;  Unload (make available + spindown)
                                 AVAILABLE,-                    ;  Available (no spindown)
                                 PACKACK,-                      ;  Pack Acknowledge
                                 DSE,-                          ;  Data Security Erase
                                 SENSECHAR,-                    ;  Sense Characteristics
                                 SETCHAR,-                      ;  Set   Characteristics
                                 SENSEMODE,-                    ;  Sense Mode
                                 SETMODE,-                      ;  Set   Mode
                                 ACCESS,-                       ;  Access file and/or find directory entry
                                 ACPCONTROL,-                   ;  ACP Control Function
                                 CREATE,-                       ;  Create file and/or create directory entry
                                 DEACCESS,-                     ;  Deaccess file
                                 DELETE,-                       ;  Delete file and/or directory entry
                                 MODIFY,-                       ;  Modify file attributes
                                 MOUNT>                         ;  Mount volume
                        FUNCTAB +ACP$READBLK,-                  ; READ FUNCTIONS
                                <READLBLK,-                     ;  Read LOGICAL Block
                                 READPBLK,-                     ;  Read PHYSICAL Block
                                 READVBLK>                      ;  Read VIRTUAL Block
                        FUNCTAB +ACP$WRITEBLK,-                 ; WRITE FUNCTIONS
                                <WRITECHECK,-                   ;  Write Check
                                 WRITELBLK,-                    ;  Write LOGICAL Block
                                 WRITEVBLK>                     ;  Write VIRTUAL Block
                        FUNCTAB +ACP$ACCESS,-                   ;
                                <ACCESS,CREATE>                 ; ACCESS AND CREATE FILE OR DIRECTORY
                        FUNCTAB +ACP$DEACCESS,<DEACCESS>        ; DEACCESS FILE
```

```
0078  1252  FUNCTAB +ACP$MODIFY,-                ;
0078  1253          <ACPCONTROL,-                ;  ACP Control Function
0078  1254           DELETE,-                    ;  Delete file or directory entry
0078  1255           MODIFY>                     ;  Modify File Attributes
0084  1256  FUNCTAB DU$DSE_FDT,<DSE>             ; Data Security Erase
0090  1257  FUNCTAB +ACP$MOUNT,<MOUNT>          ; Mount Volume
009C  1258  FUNCTAB DU$SHADOW_PACKACK_FDT, -     ; Pack Acknowledge processing for
009C  1259          <PACKACK>                    ;  shadowed volume sets.  This must
00A8  1260                                       ;  proceed EXE$LCLDSKVALID processing.
00A8  1261  FUNCTAB +EXE$LCLDSKVALID, -          ; Functions effecting UCB$V_VALID
00A8  1262          <PACKACK, -                  ;  Pack Acknowledge
00A8  1263           AVAILABLE, -                ;  Available (no spin-down)
00A8  1264           UNLOAD>                     ;  Unload
00B4  1265  FUNCTAB +EXE$ZEROPARM, -            ; Zero parameter functions
00B4  1266          <NOP>                        ;  No Operation
00C0  1267  FUNCTAB +EXE$SENSEMODE,-            ;
00C0  1268          <SENSECHAR,-                 ;  Sense Characteristics
00C0  1269           SENSEMODE>                  ;  Sense Mode
00CC  1270  FUNCTAB +EXE$SETCHAR,-              ;
00CC  1271          <SETCHAR,-                   ;  Set Characteristics
00CC  1272           SETMODE>                    ;  Set Mode
```

```
                     00D8  1274          .SBTTL  Static Storage
                     00D8  1275          .SBTTL  - Data Area Shared With Common Subroutines Module
                     00D8  1276 ;++
                     00D8  1277 ;
                     00D8  1278 ; Data Area Shared With Common Subroutines Module
                     00D8  1279 ;
                     00D8  1280 ; Functional Description:
                     00D8  1281 ;
                     00D8  1282 ;       This PSECT contains those constant (link-time) values which would
                     00D8  1283 ;       otherwise be passed as arguments to the disk and tape class driver
                     00D8  1284 ;       common routines in module DUTUSUBS.
                     00D8  1285 ;
                     00D8  1286 ;--
                     00D8  1287
                     00D8  1288          .SAVE
                     00D8  1289
                00000000   1290          .PSECT  $$$220_DUTU_DATA_01 RD,WRT,EXE,LONG
                     0000  1291
                     0000  1292          ASSUME  DUTU$L_CDDB_LISTHEAD EQ 0
                     0000  1293
                     0000  1294 ;base + DUTU$L_CDDB_LISTHEAD                    ; Location containing the
                     0000  1295                                                 ; address of the CDDB listhead
           00000000'  0000  1296          .ADDRESS IOC$GL_DU_CDDB               ; for CDDBs belonging to the
                     0004  1297                                                 ; disk device type
                     0004  1298
                000000D8   1299          .RESTORE
```

```
                     00D8  1301          .SBTTL  - Media-id to Device Type Conversion Table
                     00D8  1302 ;++
                     00D8  1303 ;
                     00D8  1304 ; Media-id to Device Type Conversion Table
                     00D8  1305 ;
                     00D8  1306 ; Functional Description:
                     00D8  1307 ;
                     00D8  1308 ;       This table is used by DUTU$GET_DEVTYPE to convert a MSCP media
                     00D8  1309 ;       identifier to a VMS device type.
                     00D8  1310 ;
                     00D8  1311 ;       Entries are made here in order of expected frequency of use.  This
                     00D8  1312 ;       speeds lookup for the more common cases.
                     00D8  1313 ;
                     00D8  1314 ;--
                     00D8  1315
                     00D8  1316          MEDIA   <DU>, <RA80>
                     0000
 25641050            0000                        .LONG   $$MEDIA$$
       14            0004                        .BYTE   DT$_RA80
                     0005
                     00D8  1317          MEDIA   <DU>, <RA81>
                     0005
 25641051            0005                        .LONG   $$MEDIA$$
       15            0009                        .BYTE   DT$_RA81
                     000A
                     00D8  1318          MEDIA   <DJ>, <RA60>
                     000A
 22A4103C            000A                        .LONG   $$MEDIA$$
       16            000E                        .BYTE   DT$_RA60
                     000F
                     00D8  1319          MEDIA   <DU>, <RA82>
                     000F
 25641052            000F                        .LONG   $$MEDIA$$
       1E            0013                        .BYTE   DT$_RA82
                     0014
                     00D8  1320          MEDIA   <DU>, <RD52>
                     0014
 25644034            0014                        .LONG   $$MEDIA$$
       1B            0018                        .BYTE   DT$_RD52
                     0019
                     00D8  1321          MEDIA   <DU>, <RD53>
                     0019
 25644035            0019                        .LONG   $$MEDIA$$
       1C            001D                        .BYTE   DT$_RD53
                     001E
                     00D8  1322          MEDIA   <DU>, <RD51>
                     001E
 25644033            001E                        .LONG   $$MEDIA$$
       19            0022                        .BYTE   DT$_RD51
                     0023
                     00D8  1323          MEDIA   <DU>, <RX50>
                     0023
 25658032            0023                        .LONG   $$MEDIA$$
       1A            0027                        .BYTE   DT$_RX50
                     0028
                     00D8  1324          MEDIA   <DA>, <RC25>
                     0028
```

```
20643019 0028                          .LONG   $$MEDIA$$
      17 002C                          .BYTE   DT$_RC25
         002D
         00D8  1325          MEDIA     <DA>, <RCF25>
         002D
20643319 002D                          .LONG   $$MEDIA$$
      18 0031                          .BYTE   DT$_RCF25
         0032
         00D8  1326          MEDIA     <DA>, <RC26>
         0032
2064301A 0032                          .LONG   $$MEDIA$$
      1F 0036                          .BYTE   DT$_RC26
         0037
         00D8  1327          MEDIA     <DA>, <RCF26>
         0037
2064331A 0037                          .LONG   $$MEDIA$$
      20 003B                          .BYTE   DT$_RCF26
         003C
         00D8  1328          MEDIA     <DU>, <CDR50>
         003C
25464932 003C                          .LONG   $$MEDIA$$
      22 0040                          .BYTE   DT$_CDR50
         0041
         00D8  1329          MEDIA     <DB>, <RP06>
         0041
20A50006 0041                          .LONG   $$MEDIA$$
      05 0045                          .BYTE   DT$_RP06
         0046
         00D8  1330          MEDIA     <DR>, <RM05>
         0046
24A4D005 0046                          .LONG   $$MEDIA$$
      0F 004A                          .BYTE   DT$_RM05
         004B
         00D8  1331          MEDIA     <DR>, <RP07>
         004B
24A50007 004B                          .LONG   $$MEDIA$$
      07 004F                          .BYTE   DT$_RP07
         0050
         00D8  1332          MEDIA     <DR>, <RM80>
         0050
24A4D050 0050                          .LONG   $$MEDIA$$
      0D 0054                          .BYTE   DT$_RM80
         0055
         00D8  1333          MEDIA     <DR>, <RM03>
         0055
24A4D003 0055                          .LONG   $$MEDIA$$
      06 0059                          .BYTE   DT$_RM03
         005A
         00D8  1334          MEDIA     <DR>, <RP08>, DT$_RP07HT
         005A
24A50008 005A                          .LONG   $$MEDIA$$
      08 005E                          .BYTE   DT$_RP07HT
         005F
         00D8  1335          MEDIA     <DB>, <RP05>
         005F
20A50005 005F                          .LONG   $$MEDIA$$
      04 0063                          .BYTE   DT$_RP05
```

```
                        0064
                        00D8  1336          MEDIA   <DB>, <RP04>
                        0064
20A50004                0064                        .LONG   $$MEDIA$$
      03                0068                        .BYTE   DT$_RP04
                        0069
                        00D8  1337          MEDIA   <DM>, <RK07>
                        0069
2364B007                0069                        .LONG   $$MEDIA$$
      02                006D                        .BYTE   DT$_RK07
                        006E
                        00D8  1338          MEDIA   <DM>, <RK06>
                        006E
2364B006                006E                        .LONG   $$MEDIA$$
      01                0072                        .BYTE   DT$_RK06
                        0073
                        00D8  1339          MEDIA   <DU>, <RX31>
                        0073
2565801F                0073                        .LONG   $$MEDIA$$
      23                0077                        .BYTE   DT$_RX31
                        0078
                        00D8  1340          MEDIA   <DU>, <RX32>
                        0078
25658020                0078                        .LONG   $$MEDIA$$
      24                007C                        .BYTE   DT$_RX32
                        007D
                        00D8  1341          MEDIA   <DU>, <RX18>
                        007D
25658012                007D                        .LONG   $$MEDIA$$
      25                0081                        .BYTE   DT$_RX18
                        0082
                        00D8  1342          MEDIA   <DL>, <RL02>
                        0082
2324C002                0082                        .LONG   $$MEDIA$$
      0A                0086                        .BYTE   DT$_RL02
                        0087
                        00D8  1343          MEDIA   <DL>, <RL01>
                        0087
2324C001                0087                        .LONG   $$MEDIA$$
      09                008B                        .BYTE   DT$_RL01
                        008C
                        00D8  1344          MEDIA   <DY>, <RX04>
                        008C
26658004                008C                        .LONG   $$MEDIA$$
      0C                0090                        .BYTE   DT$_RX04
                        0091
                        00D8  1345          MEDIA   <DY>, <RX02>
                        0091
26658002                0091                        .LONG   $$MEDIA$$
      0B                0095                        .BYTE   DT$_RX02
                        0096
                        00D8  1346          MEDIA   <DX>, <RX01>
                        0096
26258001                0096                        .LONG   $$MEDIA$$
      10                009A                        .BYTE   DT$_RX01
                        009B
                        00D8  1347          MEDIA   <DD>, <TU58>
```

```
                009B
2129503A        009B                    .LONG   $$MEDIA$$
      0E        009F                    .BYTE   DT$_TU58
                00A0
                00D8    1348    MEDIA   <DQ>, <RB02>
                00A0
24642002        00A0                    .LONG   $$MEDIA$$
      12        00A4                    .BYTE   DT$_RB02
                00A5
                00D8    1349    MEDIA   <DQ>, <RB80>
                00A5
24642050        00A5                    .LONG   $$MEDIA$$
      13        00A9                    .BYTE   DT$_RB80
                00AA
                00D8    1350    MEDIA   <ML>, <ML11>
                00AA
6B1AC00B        00AA                    .LONG   $$MEDIA$$
      11        00AE                    .BYTE   DT$_ML11
                00AF
```

```
                        00D8  1352          .SBTTL  Controller Initialization Routine
                        00D8  1353
                        00D8  1354 ;+
                        00D8  1355 ; MSCP speaking intelligent controller initialization routine.
                        00D8  1356 ;
                        00D8  1357 ; INPUTS:
                        00D8  1358 ;       R4 => System ID of intelligent controller.
                        00D8  1359 ;       R5 => IDB
                        00D8  1360 ;       R6 => DDB
                        00D8  1361 ;       R8 => CRB for intelligent controller.
                        00D8  1362 ;
                        00D8  1363
                        00D8  1364 DU_CONTROLLER_INIT:
                  06 11 00D8  1365          BRB     0$                      ; Branch around breakpoint.
00000000'GF       16 00DA  1366          JSB     G^INI$BRK               ; Breakpoint for debugging.
                        00E0  1367 0$:
                        00E0  1368
                        00E0  1369 ; Check for CDDB already present.  If a CDDB is present, this call results
                        00E0  1370 ; from a power failure.  This driver performs power failure recovery as a
                        00E0  1371 ; result of virtual circuit closure notification.  No action need be taken
                        00E0  1372 ; here.
                        00E0  1373
                10 A8   D5 00E0  1374          TSTL    CRB$L_AUXSTRUC(R8)      ; Is there a CDDB present?
                   01   13 00E3  1375          BEQL    5$                      ; Branch if CDDB is not present.
                        05 00E5  1376          RSB                             ; Else, just exit.
                        00E6  1377
                        00E6  1378 ; Check that only one UCB is chained onto the input DDB.  This UCB could be
                        00E6  1379 ; the boot device UCB.  Therefore, make the UCB online so that I/O may be
                        00E6  1380 ; performed on it.  All other initialization of the UCB is performed as the
                        00E6  1381 ; result of DPT_STORE entries place in the INIT section of the DPT by the
                        00E6  1382 ; INIT_UCB macro.
                        00E6  1383
                        00E6  1384 5$:
             55   04 A6   D0 00E6  1385          MOVL    DDB$L_UCB(R6),R5        ; R5 => first UCB if any.
          64 A5   10   C8 00EA  1386          BISL    #UCB$M_ONLINE, -        ; Set the possibly boot UCB online.
                        00EE  1387                  UCB$L_STS(R5)
                30 A5   D5 00EE  1388          TSTL    UCB$L_LINK(R5)          ; Is there another UCB?
                   04   13 00F1  1389          BEQL    10$                     ; EQL implies no more UCB's.
                        00F3  1390          BUG_CHECK       DISKCLASS,FATAL ; For now.
                        00F7  1391 10$:
                        00F7  1392
                        00F7  1393 ; Setup those values which must be correct before IPL is lowered from 31.
                        00F7  1394 ; Then FORK to create an IPL$_SCS fork thread which will complete controller
                        00F7  1395 ; initialization.  Initialization of an MSCP server requires several message
                        00F7  1396 ; exchanges and consumes several seconds.  Therefore, this work is conducted
                        00F7  1397 ; in a fork thread with other system initialization proceeding concurrently.
                        00F7  1398
                10 A8   55 D0 00F7  1399          MOVL    R5, CRB$L_AUXSTRUC(R8)  ; The UCB will act as a CDDB until the
                        00FB  1400                                              ; real one is built.
          00CC C5   64   7D 00FB  1401          MOVQ    (R4), -                 ; Setup remote system ID for call to
                        0100  1402                  UCB$Q_UNIT_ID(R5)       ; DUTU$CREATE_CDDB.
                        0100  1403
                        0100  1404          FORK                            ; Create initialization fork thread.
                        0106  1405
                        0106  1406 ; Create and initialize the CDDB.
                        0106  1407
                   FEF7' 30 0106  1408          BSBW    DUTU$CREATE_CDDB
```

```
                     0109  1409 ;
                     0109  1410 ; Here we call an internal subroutine which:
                     0109  1411 ;
                     0109  1412 ;       1. Makes a connection to the MSCP server in the intelligent
                     0109  1413 ;             controller.
                     0109  1414 ;
                     0109  1415 ;       2. Sends an MSCP command to SET CONTROLLER CHARACTERISTICS.
                     0109  1416 ;
                     0109  1417 ;       3. Allocates an MSCP buffer and RSPID for our future use in
                     0109  1418 ;             connection management.
                     0109  1419 ;
                     0109  1420 ; Upon return R4 => PDT and R5 => CDRP.
                     0109  1421 ;
                     0109  1422
     55   00D0 C5    DE  0109  1423          MOVAL   CDDB$A_PRMCDRP(R5), R5  ; Get permanent CDRP address.
            0092    30  010E  1424          BSBW    MAKE_CONNECTION         ; Call internal subroutine to make
                        0111  1425                                          ;  a connection to the MSCP server in
                        0111  1426                                          ;  the intelligent controller. Input
                        0111  1427                                          ;  and output are R5 => CDRP.
                        0111  1428
                        0111  1429 ;
                        0111  1430 ; Here we determine if the Controller is capable of performing Controller
                        0111  1431 ;       initiated bad block replacement.  If so we branch around.  If not,
                        0111  1432 ;       we call INIT_HIRT to initialize the HIRT (Host Intitiated Replacement
                        0111  1433 ;       Table.
                        0111  1434 ;
                        0111  1435
                        0111  1436          PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                        0111  1437                  cdrp=R5, cddb=R3
                        0118  1438
                  0F    E0  0118  1439          BBS     #MSCP$V_CF_REPLC,-      ; See if Controller initiated replacement.
           03 28 A3           011A  1440                  CDDB$W_CNTRLFLGS(R3),90$;  If so, branch around.
                FEE0'    30  011D  1441          BSBW    DU$INIT_HIRT            ; Initialize HIRT.
                        0120  1442
                50   18 A3   D0  0120  1443 90$:    MOVL    CDDB$L_CRB(R3),R0       ; Get CRB address.
         1C A0   0C7A'CF    9E  0124  1444          MOVAB   W^DU$TMR, -             ; Establish permanent timeout routine.
                        012A  1445                  CRB$L_TOUTROUT(R0)
                51   2A A3   3C  012A  1446          MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Get controller timeout interval.
 18 A0   00000000'GF    51   C1  012E  1447          ADDL3   R1, G^EXE$GL_ABSTIM, -  ; Use that to set next timeout
                        0137  1448                  CRB$L_DUETIME(R0)       ; wakeup time.
                        0137  1449
                        0137  1450          ; The normal MSCP timeout mechanism is now in effect.  Henceforth,
                        0137  1451          ; no fork thread may use the CDDB permanent CDRP as a fork block.
                        0137  1452
                        0137  1453          ASSUME  CDDB$V_DAPBSY GE 8
            13 A3   04   88  0137  1454          BISB    #<CDDB$M_DAPBSY @ -8>, -; Set DAP CDRP in use flag.
                        013B  1455                  CDDB$W_STATUS+1(R3)
            55   54 A3   D0  013B  1456          MOVL    CDDB$L_DAPCDRP(R3), R5  ; Get DAP CDRP address.
                FEBE'    30  013F  1457          BSBW    DUTU$POLL_FOR_UNITS     ; Poll controller for units.
                        0142  1458
         12 A3   0480 8F    AA  0142  1459          BICW    #<CDDB$M_NOCONN -       ; Clear no-connection and
                        0148  1460                   !CDDB$M_DAPBSY>, -     ; DAP-CDRP-busy flags.
                        0148  1461                  CDDB$W_STATUS(R3)
                        0148  1462
         53   0000007C 8F    C3  0148  1463          SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB in R5.
                        014F  1464                   -CDDB$C_UCBCHAIN>, -
                     55         014F  1465                  R3, R5
```

```
                        0150  1466
    55   00C4 C5   D0   0150  1467 100$:   MOVL    UCB$L_CDDB_LINK(R5), R5 ; Get next UCB if any in R5.
                 1A   13   0155  1468        BEQL    130$                    ; EQL implies no more UCB's.
68 A5    0400 8F   AA   0157  1469        BICW    #UCB$M_MSCP_WAITBMP, -  ; Indicate RWAITCNT no longer bumped.
                        015D  1470                UCB$W_DEVSTS(R5)
       56 A5     B7   015D  1471        DECW    UCB$W_RWAITCNT(R5)      ; Decrement wait reason count.
                 03   13   0160  1472        BEQL    120$                    ; If count is zero, startup I/O.
          FE9B'  30   0162  1473        BSBW    DUTU$CHECK_RWAITCNT     ; Else, check for vaild wait count.
                 3F   BB   0165  1474 120$:   PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save registers before call.
00000000'GF      16   0167  1475        JSB     G^SCS$UNSTALLUCB        ; Startup any queued up I/O requests.
                 3F   BA   016D  1476        POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore registers after call.
                 DF   11   016F  1477        BRB     100$                    ; Loop back to test more UCB's (if any).
                        0171  1478
 12 A3     04   AA   0171  1479 130$:   BICW    #CDDB$M_INITING, -      ; Clear the "initing" bit.
                        0175  1480                CDDB$W_STATUS(R3)
                 05   0175  1481        RSB                             ; Terminate this thread of execution.
                        0176  1482
                        0176  1483 INIT_TIMEOUT:                        ; Controller Init Timeout handler.
          0915   31   0176  1484        BRW     DU$RE_SYNCH             ; If we timeout, try to restart.
```

```
                        0179  1486          .SBTTL  MAKE_CONNECTION
                        0179  1487
                        0179  1488 ; MAKE_CONNECTION - Internal subroutine, called from DU_CONTROLLER_INIT and
                        0179  1489 ;       DU$CONNECT_ERR, that establishes a connection to the MSCP server
                        0179  1490 ;       in the intelligent controller.
                        0179  1491 ;
                        0179  1492 ; INPUTS:
                        0179  1493 ;       R5 => permanent CDRP
                        0179  1494 ;
                        0179  1495 ; OUTPUTS:
                        0179  1496 ;       Connection established and initial SET CONTROLLER CHARACTERISTICS
                        0179  1497 ;       command is sent to controller.  Also an MSCP buffer and an RSPID
                        0179  1498 ;       are allocated for the connection.
                        0179  1499 ;
                        0179  1500 ;       Side effects include the fact that all registers, except R5, are
                        0179  1501 ;       modified.
                        0179  1502 ;
                        0179  1503
5F 4C 43 5F 4B 53 49 44 24 53 4D 56   0179  1504 CLASS_DRVR_NAME:        .ASCII  /VMS$DISK_CL_DRVR/
                              52 56 52 44     0185
20 20 20 4B 53 49 44 24 50 43 53 4D   0189  1505 MSCP_SRVR_NAME:         .ASCII  /MSCP$DISK       /
                        20 20 20 20   0195
                        0199  1506
                        0199  1507 HSTIMEOUT_ARRAY:                          ; Host timeouts for various controllers.
                        0199  1508          ASSUME  MSCP$K_CM_HSC50 EQ      1
                        0199  1509          ASSUME  MSCP$K_CM_UDA50 EQ      2
                        0199  1510          ASSUME  MSCP$K_CM_RC25  EQ      3
                        0199  1511          ASSUME  MSCP$K_CM_EMULA EQ      4
                        0199  1512          ASSUME  MSCP$K_CM_TU81  EQ      5
                        0199  1513          ASSUME  MSCP$K_CM_UDA52 EQ      6
                        0199  1514          ASSUME  MSCP$K_CM_RDRX  EQ      7
                  1E    0199  1515          .BYTE   HOST_TIMEOUT            ; Use default constant for HSC50.
                  FF    019A  1516          .BYTE   255                     ; Long timeout for dedicated controller
                        019B  1517                                          ;  (UDA50) with dual portable drives.
                  00    019B  1518          .BYTE   0                       ; Use zero for dedicated controller.(AZTEC)
                  1E    019C  1519          .BYTE   HOST_TIMEOUT            ; Use default constant for Emulator.
                  00    019D  1520          .BYTE   0                       ; Use zero for dedicated controller.(TU81)
                  FF    019E  1521          .BYTE   255                     ; Long timeout for dedicated controller
                        019F  1522                                          ;  (UDA52) with dual portable drives.
                  00    019F  1523          .BYTE   0                       ; Use zero for dedicated controller.(RD/RX)
                  00    01A0  1524          .BYTE   0                       ; Patch space for expansion
                  00    01A1  1525          .BYTE   0
                  00    01A2  1526          .BYTE   0
                        01A3  1527
                        01A3  1528 MAKE_CONNECTION:
                        01A3  1529
                        01A3  1530          PERMCDRP_TO_CDDB -              ; Get CDDB address form CDRP.
                        01A3  1531                  cdrp=R5, cddb=R2
                44 A2 8ED0    01AA  1532          POPL    CDDB$L_SAVED_PC(R2) ; Save caller's return in CDDB field.
                        01AE  1533
                        01AE  1534 10$:
                        01AE  1535          CONNECT DU$IDR,-                ; Entry point of Input Dispatcher Routine.
                        01AE  1536                  DU$DGDR,-               ; Entry point of Datagram Dispatcher.
                        01AE  1537                  DU$CONNECT_ERR,-        ; Error entry point.
                        01AE  1538                  CDDB$B_SYSTEMID(R2),-   ; Destination SYSTEM ID.
                        01AE  1539                  ,-                      ; Remote station address.
                        01AE  1540                  MSCP_SRVR_NAME,-        ; MSCP server name.
```

```
                                01AE  1541                    CLASS_DRVR_NAME,-       ; Ascii of class driver name.
                                01AE  1542                    #INITIAL_CREDIT,-       ;  Needs definition
                                01AE  1543                    #MIN_SEND_CREDIT,-      ; Minimum send credit
                                01AE  1544                    #INITIAL_DG_COUNT,-     ; Initial DataGram count
                                01AE  1545                    ,-                      ; Block transfer priority
                                01AE  1546                    ,-                      ; Connect data
                                01AE  1547                    (R2),-                  ; Also pass CDDB address to CDT$L_AUXSTRUC
                                01AE  1548                    ,                       ; Bad Response packet address
                                01E6  1549
                    29 50   E8  01E6  1550            BLBS    R0,30$                  ; LBS implies success, so branch around.
                                01E9  1551
                                01E9  1552            PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                                01E9  1553                    cdrp=R5, cddb=R3
                  53   18 A3   D0  01F0  1554            MOVL    CDDB$L_CRB(R3), R3      ; Get CRB address.
                1C A3  03'AF   9E  01F4  1555            MOVAB   B^20$,CRB$L_TOUTROUT(R3); Establish LABEL as place to call, for
                                01F9  1556                                            ;  now, for periodic wakeups.
                      0A   C1   01F9  1557            ADDL3   #CONNECT_DELTA,-        ; Establish Due time as a little in
                00000000'GF     01FB  1558                    G^EXE$GL_ABSTIM,-       ;  the future.
                       18 A3    0200  1559                    CRB$L_DUETIME(R3)
                          05    0202  1560            RSB                             ; Return to caller's caller and kill
                                0203  1561                                            ;  this thread.
                                0203  1562  20$:
                  52   10 A3   D0  0203  1563            MOVL    CRB$L_AUXSTRUC(R3),R2   ; R2 => CDDB.
               55   00D0 C2   9E  0207  1564            MOVAB   CDDB$A_PRMCDRP(R2),R5   ; Get permanent CDRP address.
                                020C  1565            SETIPL  #IPL$_SCS               ; Lower IPL after wakeup.
                      FF9C   31  020F  1566            BRW     10$                     ; Loop back and try CONNECT again.
                                0212  1567  30$:       ; A connection has been established
                                0212  1568            PERMCDRP_TO_CDDB -              ; Get CDDB address from CDRP.
                                0212  1569                    cdrp=R5, cddb=R1
              00F4 C1    53   D0  0219  1570            MOVL    R3, CDDB$L_CDT(R1)      ; Save CDT address (in perm CDRP).
                14 A1    54   D0  021E  1571            MOVL    R4, CDDB$L_PDT(R1)      ; Save PDT address.
              01B8 C1    53   D0  0222  1572            MOVL    R3, CDDB$L_DAPCDT(R1)   ; Save CDT address in DAP CDRP too.
                   53    51   D0  0227  1573            MOVL    R1, R3                  ; Now that CDT is saved, move CDDB addr.
                                022A  1574
                  51   18 A3   D0  022A  1575            MOVL    CDDB$L_CRB(R3), R1      ; Get CRB address.
                18 A1     01   CE  022E  1576            MNEGL   #1, CRB$L_DUETIME(R1)   ; Infinite time till next timeout, now.
             1C A1   FF40 CF   9E  0232  1577            MOVAB   INIT_TIMEOUT, -         ; Establish timeout routine that will
                                0238  1578                    CRB$L_TOUTROUT(R1)      ; serve for rest of controller init.
                                0238  1579
                                0238  1580  ;
                                0238  1581  ; Here we prepare to send a SET CONTROLLER CHARACTERISTICS MSCP Packet to
                                0238  1582  ;       the intelligent controller over the connection that we have just
                                0238  1583  ;       established.
                                0238  1584  ;
                                0238  1585
                                0238  1586            ALLOC_RSPID                     ; ALLOCate a ReSPonse ID.
                                023E  1587            ALLOC_MSG_BUF                   ; Allocate an MSCP buffer (and also
                                0241  1588                                            ;  allocate a unit of flow control).
                    07 50   E8  0241  1589            BLBS    R0,50$                  ; If success, branch around.
                53   18 A3   D0  0244  1590            MOVL    CDDB$L_CRB(R3),R3       ; DU$RE-SYNCH expects R3 => CRB.
                      0843   31  0248  1591            BRW     DU$RE_SYNCH             ; If we fail, try to restart.
                                024B  1592  50$:                                       ; Here R2 => MSCP buffer allocated.
                      51     D4  024B  1593            CLRL    R1                      ; First set Controller Characteristics
                                024D  1594                                            ;  with zero (i.e. infinite) host timeout.
                      3C     10  024D  1595            BSBB    PRP_STCON_MSG           ; Call to prepare MSCP command.
                                024F  1596            SEND_MSCP_MSG_DRIVER            ; Returns with end-message addr. in R2.
                    006A     30  0252  1597            BSBW    RECORD_STCON            ; Record Controller Characteristics.
```

```
                     0255  1598
                     0255  1599          RECYCH_MSG_BUF                  ; We recycle the END PACKET and
                     0258  1600                                          ;  thereby allocate a new send credit.
                     0258  1601          RECYCL_RSPID                    ; We also recycle the RSPID.
                     025E  1602
                     025E  1603 ; Determine the correct host timeout interval.  This is the larger of
                     025E  1604 ; HSTIMEOUT_ARRAY[controller_model] and the controller timeout interval
                     025E  1605 ; returned by the just completed Set Controller Characteristics.  There is,
                     025E  1606 ; however, one wrinkle.  Zero represents an infinite timeout and therefore is
                     025E  1607 ; larger than any other number.  Also, the controller already believes the
                     025E  1608 ; host timeout interval to be infinite, as the result of the previous Set
                     025E  1609 ; Controller Characteristics command.  Therefore, no further action need be
                     025E  1610 ; taken when the timeout interval is infinite.
                     025E  1611
      51   26 A3    9A 025E  1612          MOVZBL  CDDB$B_CNTRLMDL(R3),R1  ; Get controller model type.
51    FF31 CF41    9A 0262  1613          MOVZBL  HSTIMEOUT_ARRAY-1[R1],R1; Get corresponding host timeout value.
           1E    13 0268  1614          BEQL    60$                     ; If zero, branch around.
      50   2A A3    3C 026A  1615          MOVZWL  CDDB$W_CNTRLTMO(R3), R0 ; Get controller timeout interval.
           18    13 026E  1616          BEQL    60$                     ; If controller timeout is infinite,
                    0270  1617                                          ; use already set infinite host timeout.
         51   50    D1 0270  1618          CMPL    R0, R1                  ; Compare with HSTIMEOUT_ARRAY value.
           03    1F 0273  1619          BLSSU   55$                     ; Branch if HSTIMEOUT_ARRAY is larger.
         51   50    D0 0275  1620          MOVL    R0, R1                  ; Else, use controller timeout as
                    0278  1621                                          ; host timeout interval.
                    0278  1622 55$:
           11    10 0278  1623          BSBB    PRP_STCON_MSG           ; Else reset controller characteristics.
                    027A  1624          SEND_MSCP_MSG DRIVER            ; Returns with end-message addr. in R2.
           40    10 027D  1625          BSBB    RECORD_STCON            ; Record Controller Characteristics.
                    027F  1626          RECYCH_MSG_BUF                  ; Again we recycle the END PACKET and
                    0282  1627                                          ;  thereby allocate a new send credit.
                    0282  1628          RECYCL_RSPID                    ; We also recycle the RSPID.
                    0288  1629 60$:
                    0288  1630
44 B3    17         0288  1631          JMP     @CDDB$L_SAVED_PC(R3)    ; Return to caller.
```

```
                        028B  1633 ; PRP_STCON_MSG - Prepare a Set Controller Characteristics Command Message.
                        028B  1634 ;
                        028B  1635 ; Inputs:
                        028B  1636 ;       R1 =  Host Timeout Value
                        028B  1637 ;       R2 => MSCP buffer to fill
                        028B  1638 ;       R3 => CDDB
                        028B  1639 ;       R5 => CDRP
                        028B  1640 ;
                        028B  1641
                        028B  1642 PRP_STCON_MSG:
                        028B  1643
            51    DD    028B  1644          PUSHL   R1                      ; Save important register.
                        028D  1645          INIT_MSCP_MSG                   ; Initialize buffer for MSCP message.
            51  8ED0     0290  1646          POPL    R1                      ; Restore important register.
                        0293  1647
            04    90    0293  1648          MOVB    #MSCP$K_OP_STCON,-      ; Insert SET CONTROLLER CHARACTERISTICS
          08 A2           0295  1649                  MSCP$B_OPCODE(R2)       ;  opcode with NO modifiers.
                        0297  1650
          28 A3     B0    0297  1651          MOVW    CDDB$W_CNTRLFLGS(R3),-  ; Set host settable characteristics
          0E A2           029A  1652                  MSCP$W_CNT_FLGS(R2)     ;  bits into MSCP command message.
                        029C  1653
       10 A2    51    B0    029C  1654          MOVW    R1,MSCP$W_HST_TMO(R2)   ; Set host timeout into MSCP packet.
                        02A0  1655
00000000'GF     7D    02A0  1656          MOVQ    G^EXE$GQ_SYSTIME,-      ; Transmit time of century in clunks.
            14 A2       02A6  1657                  MSCP$Q_TIME(R2)
                        02A8  1658
    50    18 A3     D0    02A8  1659          MOVL    CDDB$L_CRB(R3),R0       ; R0 => CRB.
    7E    2A A3     3C    02AC  1660          MOVZWL  CDDB$W_CNTRLTMO(R3),-(SP); Pickup controller delta.
            03    12    02B0  1661          BNEQ    70$                     ; NEQ implies this controller has been
                        02B2  1662                                          ;  init'ed at least once before.
          6E    1E    D0    02B2  1663          MOVL    #INIT_IMMED_DELTA,(SP)  ; Else use compiled in timeout.
                        02B5  1664 70$:
            8E    C1    02B5  1665          ADDL3   (SP)+,-                 ; Establish delta time for time out
00000000'GF             02B7  1666                  G^EXE$GL_ABSTIM,-       ;  to prevent against controller never
          18 A0         02BC  1667                  CRB$L_DUETIME(R0)       ;  responding.
                        02BE  1668
                05    02BE  1669          RSB                             ; Return to caller.
```

```
                        02BF  1671 ; RECORD_STCON - Record data from a Set Controller Characteristics end message
                        02BF  1672 ;                in the CDDB.
                        02BF  1673 ;
                        02BF  1674 ; Inputs:
                        02BF  1675 ;       R2 => MSCP End Message
                        02BF  1676 ;       R3 => CDDB
                        02BF  1677
                        02BF  1678
                        02BF  1679 RECORD_STCON:
                 0E A2  B0  02BF  1680         MOVW    MSCP$W_CNT_FLGS(R2),-   ; Pickup NON-host settable characteristics
                 28 A3      02C2  1681                 CDDB$W_CNTRLFLGS(R3)    ;  from END PACKET and save in CDDB.
                            02C4  1682
                 10 A2  B0  02C4  1683         MOVW    MSCP$W_CNT_TMO(R2),-    ; Likewise with controller timeout.
                 2A A3      02C7  1684                 CDDB$W_CNTRLTMO(R3)
                            02C9  1685
                 14 A2  7D  02C9  1686         MOVQ    MSCP$Q_CNT_ID(R2),-     ; Also save controller unique ID.
                 20 A3      02CC  1687                 CDDB$Q_CNTRLID(R3)
                            02CE  1688
              23 12 A3  06  E2  02CE  1689         BBSS    #CDDB$V_ALCLS_SET, -    ; Branch if allocation class already
                            02D3  1690                 CDDB$W_STATUS(R3), 90$  ; set, and indicate it is now set.
                            02D3  1691
                            02D3  1692         ; The allocation class is about to be set for this device.  The object
                            02D3  1693         ; is to give every reasonable chance for the value to be non-zero.
           50 A3  00000000'GF  D0  02D3  1694         MOVL    G^CLU$GL_ALLOCLS, -     ; Assume a local, single host
                            02DB  1695                 CDDB$L_ALLOCLS(R3)      ; controller.
              05 28 A3  02  E1  02DB  1696         BBC     #MSCP$V_CF_MLTHS, -     ; Branch if a single host controller.
                            02E0  1697                 CDDB$W_CNTRLFLGS(R3), -
                            02E0  1698                 80$
              50 A3  04 A2  9A  02E0  1699         MOVZBL  MSCP$B_CNT_ALCS(R2), -  ; Get set controller characteristics
                            02E5  1700                 CDDB$L_ALLOCLS(R3)      ; allocation class.
                 50  E4 A3  9E  02E5  1701 80$:   MOVAB   <CDDB$L_DDB -           ; Init loop through all DDBs.
                            02E9  1702                 -DDB$L_CONLINK>(R3), R0
                 50  38 A0  D0  02E9  1703 82$:   MOVL    DDB$L_CONLINK(R0), R0   ; Link to next DDB.
                        07  13  02ED  1704         BEQL    90$                     ; Branch if no more DDBs.
              3C A0  50 A3  D0  02EF  1705         MOVL    CDDB$L_ALLOCLS(R3), -   ; Copy allocation class to this
                            02F4  1706                 DDB$L_ALLOCLS(R0)       ; DDB.
                        F3  11  02F4  1707         BRB     82$                     ; Loop till no more DDBs.
                            02F6  1708
                            05  02F6  1709 90$:   RSB
```

```
02F7  1711          .SBTTL  DU_REVALIDATE_DISKS - Revalidate previously online disks
02F7  1712 ;++
02F7  1713 ;
02F7  1714 ; DU_REVALIDATE_DISKS - Revalidate previously online disks
02F7  1715 ;
02F7  1716 ; Functional Description:
02F7  1717 ;
02F7  1718 ;       This routine supervises the revalidation of all disks which once were
02F7  1719 ;       valid (online) but which have gone offline due to a connection
02F7  1720 ;       failure.  This routine is also called whenever the connection manager
02F7  1721 ;       wishes to stall all disk I/O activity by forcing all disks into mount
02F7  1722 ;       verification.  The first case is distinguished from the second by the
02F7  1723 ;       CDDB$V_QUORLOST bit in CDDB$W_STATUS being clear.
02F7  1724 ;
02F7  1725 ;       For each device not already in mount verification whose UCB has the
02F7  1726 ;       UCB$V_VALID bit set in UCB$L_STS, mount verification is invoked.
02F7  1727 ;       Mount verification is started by feeding it one of the permanent
02F7  1728 ;       CDRP's associated with the CDDB and an appropriate error status.
02F7  1729 ;
02F7  1730 ;       If mount verification cannot be performed on the device (for example
02F7  1731 ;       the device is mounted foreign),  then all pending I/O requests for the
02F7  1732 ;       device are located and terminated with SS$_MEDOFL, medium offline.
02F7  1733 ;
02F7  1734 ;       This routine is only interested in getting mount verification started
02F7  1735 ;       wherever possible.  If mount verification can be started it will
02F7  1736 ;       complete in one of the following ways:
02F7  1737 ;
02F7  1738 ;           -   A suitable alternate path will be found.  This can happen even
02F7  1739 ;               before a new connection is established for this CDDB.
02F7  1740 ;
02F7  1741 ;           -   The connection associated with this CDDB will be reestablished
02F7  1742 ;               and access to the device via that connection restored.
02F7  1743 ;
02F7  1744 ;           -   Mount verification will timeout or otherwise abort.
02F7  1745 ;
02F7  1746 ;       All of these cases are handled in DU_END_MOUNTVER.
02F7  1747 ;
02F7  1748 ;           -   Successful completion of mount verification (with RWAITCNT
02F7  1749 ;               equal to zero) causes the UCB to be unstalled and I/O activity
02F7  1750 ;               to be resumed.
02F7  1751 ;
02F7  1752 ;           -   When mount verfication completes successfully but the RWAITCNT
02F7  1753 ;               is not zero, the UCB is not unstalled.  That duty befalls
02F7  1754 ;               whomever has bumped RWAITCNT.
02F7  1755 ;
02F7  1756 ;           -   If mount verification aborted, all pending I/O requests must
02F7  1757 ;               be located and terminated with SS$_VOLINV.
02F7  1758 ;
02F7  1759 ; Synchronizing Device Revalidation After a Connection Failure
02F7  1760 ;
02F7  1761 ;       For each UCB placed into mount verification, this routine increments a
02F7  1762 ;       counter in the CDDB which represents the failed connection.  The UCB
02F7  1763 ;       is also made to point to this CDDB.  After reestablishing a connection
02F7  1764 ;       and polling for units, the reconnection routine determines whether or
02F7  1765 ;       not any UCBs are still waiting for mount verification via the new
02F7  1766 ;       connection.
02F7  1767 ;
```

```
02F7  1768 ;       If no UCBs are waiting, the single step processing CDRPs active at the
02F7  1769 ;       time the connection failed begins immediately.  Otherwise, the
02F7  1770 ;       reconnection thread exits, to wait for the pending mount verification
02F7  1771 ;       operations to complete.
02F7  1772 ;
02F7  1773 ;       As a UCB completes mount verification, the end mount verification
02F7  1774 ;       routine uses the CDDB pointer setup by this routine to locate the CDDB
02F7  1775 ;       which is waiting for mount verification completion.  The counter of
02F7  1776 ;       waiting UCBs in that CDDB is reduced by one (representing the UCB
02F7  1777 ;       which just finished mount verification).  When that count reaches zero,
02F7  1778 ;       the reconnection logic thread is resumed at RESTART_NEXT_CDRP.
02F7  1779 ;
02F7  1780 ;       In this way, all the CDRPs active at the time of the connection
02F7  1781 ;       failure (which may have been due to a insane server) can be retried
02F7  1782 ;       one at a time after the connection has been restored and the disks
02F7  1783 ;       (or UCBs) validated by mount verification.  This is an important
02F7  1784 ;       element of error recovery for MSCP devices.
02F7  1785 ;
02F7  1786 ;       N.B. CDRPs for disks failed over to an alternate controller do not
02F7  1787 ;       participate in this single step retry mechanism, but by treating all
02F7  1788 ;       types of mount verification completetion equally, waiting
02F7  1789 ;       unnecessarily for a failed over disk is avoided.
02F7  1790 ;
02F7  1791 ;       Since this routine is called from both reconnection processing on
02F7  1792 ;       controller initialization and since having RESTART_NEXT_CDRP entered
02F7  1793 ;       due to controller initialization is very undesirable, this routine
02F7  1794 ;       does not set UCB$L_WAIT_CDDB when the CDDB$V_INITING flag is set in
02F7  1795 ;       the input CDDB.  This effectively eliminates the chain of events which
02F7  1796 ;       would cause RESTART_NEXT_CDRP to be called.
02F7  1797 ;
02F7  1798 ;       Consider a few cases:
02F7  1799 ;
02F7  1800 ;       The connection fails for a device which has been chugging along and
02F7  1801 ;       has some requests outstanding.  The outstanding requests are placed on
02F7  1802 ;       the CDDB restart queue.  Mount verification begins throwing requests
02F7  1803 ;       at the driver.  These requests complete quickly because no connection
02F7  1804 ;       is active and no alternate path is available.  The UCB points to the
02F7  1805 ;       CDDB which it turn is waiting for mount verification to complete for
02F7  1806 ;       the UCB.  When the connection is restored, the reconnect thread notices
02F7  1807 ;       the waiting UCB as exits before restarting the pending CDRPs.  This
02F7  1808 ;       leaves the wait count bumpped by 2, one for the mount verification and
02F7  1809 ;       one for the incomplete reconnection thread.  When mount verification
02F7  1810 ;       completes, the waint count is reduced by one to one, but this is not
02F7  1811 ;       sufficient to unstall any pending requests (those received while the
02F7  1812 ;       connection was broken).  Eventually, the end mount verification
02F7  1813 ;       routine finds no more waiting UCBs on the CDDB and the single step
02F7  1814 ;       delivery of requests begins.  Once all outstanding requests (from the
02F7  1815 ;       time of the connection failure) have been singly passed to the server,
02F7  1816 ;       the wait count is reduced by one to zero, this allows the pending
02F7  1817 ;       requests to be starting (in parrallel mode).  If the device can
02F7  1818 ;       failover to another server or if mount verification aborts,
02F7  1819 ;       outstanding requests are removed from the CDDB restart queue and the
02F7  1820 ;       end of mount verification processing is performed as described above.
02F7  1821 ;
02F7  1822 ;       If mount verification were already in progress when the connection
02F7  1823 ;       failed, there is no need to begin it again for recovery after the
02F7  1824 ;       connection reforms.  Also, the CDDB need not wait for mount
```

```
                     02F7  1825 ;       verification to complete on the new connection before starting single
                     02F7  1826 ;       step delivery of outstanding requests.  The only possible outstanding
                     02F7  1827 ;       request is from mount verification.  All other requests are on the
                     02F7  1828 ;       pending I/O queue.  When the connection fails, the mount verification
                     02F7  1829 ;       I/O request is either active or inactive.  If it is active, it will be
                     02F7  1830 ;       queued for post processing with a status of SS$_MEDOFL.  If it is
                     02F7  1831 ;       inactive, the next attempt to make a mount verification request will
                     02F7  1832 ;       be caught by the start I/O routine.
                     02F7  1833 ;
                     02F7  1834 ; Inputs:
                     02F7  1835 ;
                     02F7  1836 ;       R3      CDDB address
                     02F7  1837 ;
                     02F7  1838 ; Implicit inputs:
                     02F7  1839 ;
                     02F7  1840 ;       CDDB$L_UCBCHAIN(R3)     chain of UCBs on this connection
                     02F7  1841 ;       CDDB$W_STATUS(R3)       CDDB$V_INITING set if called from controller
                     02F7  1842 ;                               initialization.
                     02F7  1843 ;                               CDDB$V_QUORLOST set if called to force mount
                     02F7  1844 ;                               verification after a quorum loss.
                     02F7  1845 ;
                     02F7  1846 ; Outputs:
                     02F7  1847 ;
                     02F7  1848 ;       R3      CDDB address (unchanged)
                     02F7  1849 ;
                     02F7  1850 ;       R0 - R2, and R4 - R5 are destroyed.
                     02F7  1851 ;       All other registers are preserved.
                     02F7  1852 ;
                     02F7  1853 ; Implicit outputs:
                     02F7  1854 ;
                     02F7  1855 ;       CDDB$L_WTUCBCTR(R3)     incremented once for each UCB waiting for
                     02F7  1856 ;                               mount verification to complete.
                     02F7  1857 ;       UCB$L_WAIT_CDDB         of each waiting UCB is made to point to the
                     02F7  1858 ;                               input CDDB.
                     02F7  1859 ;--
                     02F7  1860
                     02F7  1861 DU_REVALIDATE_DISKS:
                     02F7  1862
55   53   FFFFFF84 8F   C1  02F7  1863          ADDL3   #<CDDB$L_UCBCHAIN -             ; Get "previous" UCB address.
                     02FF  1864                          -UCB$L_CDDB_LINK>, R3, R5
                     02FF  1865
                     02FF  1866 NEXT_REVAL_UCB:
                     02FF  1867
     55   00C4 C5   D0  02FF  1868          MOVL    UCB$L_CDDB_LINK(R5), R5         ; Link to next UCB on conn.
                       6B  13  0304  1869          BEQL    REVAL_STARTED           ; Branch if no more UCBs.
     F4 64 A5     0B  E1  0306  1870 10$:     BBC     #UCB$V_VALID, UCB$L_STS(R5), - ; Does UCB need revalidation?
                     030B  1871                          NEXT_REVAL_UCB          ; Branch if no revalidation.
     EF 68 A5     08  E0  030B  1872          BBS     #UCB$V_MSCP_MNTVERIP, -         ; Is mount verification already
                     0310  1873                          UCB$W_DEVSTS(R5), -     ; in progress?
                     0310  1874                          NEXT_REVAL_UCB          ; Branch if mv in progress.
     0E 12 A3     09  E1  0310  1875          BBC     #CDDB$V_QUORLOST, -             ; Branch if not quorum lost
                     0315  1876                          CDDB$W_STATUS(R3), 30$  ; processing.
                     0315  1877
                     0315  1878          ; For quorum lost processing, this routine is called for all disk
                     0315  1879          ; CDDBs known to the system.  It must then decide whether or not the
                     0315  1880          ; UCBs hanging off those CDDBs must have their I/O requests stalled.
                     0315  1881          ; The rule is that any disk accessible from more than one host must
```

```
                     0315  1882          ; have its I/O stalled.  The following tests implement that rule.
                     0315  1883
05 28 A3   02   E0   0315  1884          BBS     #MSCP$V_CF_MLTHS, -                 ; Branch if this is a multi-
                     031A  1885                  CDDB$W_CNTRLFLGS(R3), 25$           ; host controller.
E0 3C A5   04   E1   031A  1886          BBC     #DEV$V_2P, -                        ; Branch if disk is not
                     031F  1887                  UCB$L_DEVCHAR2(R5), -               ; dual-pathed.
                     031F  1888                  NEXT_REVAL_UCB
                     031F  1889
                     031F  1890  25$:     ASSUME  UCB$V_CLUTRAN GE 16                 ; Since this disk needs MV after
   66 A5   20   88   031F  1891          BISB    #<UCB$M_CLUTRAN @ -16>, -           ; a VAXcluster state transition,
                     0323  1892                  UCB$L_STS+2(R5)                     ; set UCB$V_CLUTRAN.
                     0323  1893
                     0323  1894  30$:     ; Mount verification needs to be called for this device (or UCB).
                     0323  1895          ; Understanding what follows requires several pieces of information:
                     0323  1896          ;
                     0323  1897          ;   1.  Mount verification returns to its caller only if it cannot
                     0323  1898          ;       perform verification for the UCB submitted to it.  For
                     0323  1899          ;       example, control is returned to the caller if the device is
                     0323  1900          ;       not mounted.
                     0323  1901          ;   2.  In all other cases, mount verification returns to its caller's
                     0323  1902          ;       caller.
                     0323  1903          ;   3.  If mount verification is possible, EXE$MOUNTVER will call the
                     0323  1904          ;       driver's mount verification routine, DU$MOUNTVER, which is
                     0323  1905          ;       supposed to process the IRP.  Since this IRP is a dummy, one
                     0323  1906          ;       of the permanent IRP/CDRP pairs associated with the CDDB,
                     0323  1907          ;       DU$MOUNTVER will do mount verification book keeping and then
                     0323  1908          ;       RSB; since no special action is required.
                     0323  1909          ;   4.  Mount verification could result in a failover, which would
                     0323  1910          ;       alter UCB$L_CDDB_LINK.  Since the UCBs on the current
                     0323  1911          ;       connection are the only ones of interest, the next UCB address
                     0323  1912          ;       is preserved on the stack across the call to mount verification.
                     0323  1913          ;   5.  Setting the UCB$M_SUPMVMSG bit causes those mount verficiation
                     0323  1914          ;       messages associated with a successful mount verification
                     0323  1915          ;       operation to be suppressed.  If something goes wrong or the
                     0323  1916          ;       mount verfication operation nears the timeout stage, these
                     0323  1917          ;       messages will be output (as usual).
                     0323  1918
         00C4 C5     DD   0323  1919          PUSHL   UCB$L_CDDB_LINK(R5)             ; Push next UCB address.
   64 A5 00040000 8F  C8   0327  1920          BISL    #UCB$M_SUPMVMSG, UCB$L_STS(R5)  ; Suppress mount ver. messages.
               28  BB   032F  1921          PUSHR   #^M<R3,R5>                      ; Save CDDB and UCB addresses.
53 00BC C5 00000070 8F  C1   0331  1922          ADDL3   #CDDB$A_PRMIRP, -               ; Get a permanent IRP address.
                     033B  1923                  UCB$L_CDDB(R5), R3
         50   01A4 8F  3C   033B  1924          MOVZWL  #SS$_MEDOFL, R0                 ; Setup a suitable status.
               51  D4   0340  1925          CLRL    R1
            54'AF  9F   0342  1926          PUSHAB  B^50$                           ; Setup caller's caller.
        00000000'GF  16   0345  1927          JSB     G^EXE$MOUNTVER                  ; Start mount verification.
                     034B  1928
                     034B  1929          ; Cannot do mount verification
                     034B  1930
               8E    D5   034B  1931          TSTL    (SP)+                           ; Pop caller's caller addr.
               28    BA   034D  1932          POPR    #^M<R3,R5>                      ; Restore CDDB and UCB addrs.
            FCAE'    30   034F  1933          BSBW    DUTU$TERMINATE_PENDING          ; Terminate all pending I/O.
               18    11   0352  1934          BRB     90$                             ; Loop through all UCBs.
                     0354  1935
                     0354  1936  50$:     ; Mount verification is in progress
                     0354  1937
               28    BA   0354  1938          POPR    #^M<R3,R5>                      ; Restore CDDB and UCB addrs.
```

```
                12 A3   0204 8F    B3   0356  1939          BITW    #<CDDB$M_INITING -                      ; If called from controller
                                           035C  1940                   !CDDB$M_QUORLOST>, -                ; initialization or quorum
                                           035C  1941                   CDDB$W_STATUS(R3)                   ; lost processing, skip this
                              0E    12     035C  1942          BNEQ    90$                                     ; setup.
                         00DC C5    D5     035E  1943          TSTL    UCB$L_WAIT_CDDB(R5)                     ; Check of double waiting UCB.
                              0E    12     0362  1944          BNEQ    DBL_WAIT_UCB                            ; This is a serious error.
                            5E A3    B6     0364  1945          INCW    CDDB$W_WTUCBCTR(R3)                     ; Count waiting UCB.
                      00DC C5 53     D0     0367  1946          MOVL    R3, UCB$L_WAIT_CDDB(R5)                 ; Store waiting CDDB address.
                              55 8ED0  036C  1947 90$:    POPL    R5                                      ; Get next UCB.
                              95    12     036F  1948          BNEQ    10$                                     ; If there is a UCB there, go
                                           0371  1949                                                      ; processs it.
                                           0371  1950
                                           0371  1951          ; All UCBs on the connection have been processed, one way or the
                                           0371  1952          ; other.  Control is now return this routine's caller.  As devices
                                           0371  1953          ; (or UCBs) complete mount verification DU$MOUNTVER will receive
                                           0371  1954          ; control and perform end of mount verification processing.
                                           0371  1955
                                           0371  1956 REVAL_STARTED:
                                           0371  1957
                                    05     0371  1958          RSB
                                           0372  1959
                                           0372  1960 DBL_WAIT_UCB:
                                           0372  1961          BUG_CHECK DISKCLASS, FATAL                  ; Doublely waiting UCB; fatal
                                           0376  1962                                                      ; error.
```

```
0376  1964          .SBTTL  Mount Verification Entry Point
0376  1965 ;++
0376  1966 ;
0376  1967 ; DU$MOUNTVER
0376  1968 ;
0376  1969 ; Functional Description:
0376  1970 ;
0376  1971 ;       The disk class driver and mount verification are in bed together.
0376  1972 ;       They are so intimately related that one of them is most certainly
0376  1973 ;       "with child" and nuptual annoucements will soon be made.
0376  1974 ;
0376  1975 ;       In addition to the usual services performed by mount verification, the
0376  1976 ;       disk class driver calls for its assistance in restoring access to
0376  1977 ;       disks lost due to the various failure conditions which can occur with
0376  1978 ;       a MSCP server.
0376  1979 ;
0376  1980 ;       The following is a list of reasons why this routine might be called:
0376  1981 ;
0376  1982 ;           -   starting mount verification due to appropriate status is a
0376  1983 ;               normal I/O request.  This probably is the result of the a
0376  1984 ;               change of state in the drive.  Hopefully, mount verification
0376  1985 ;               will restore the drive to the online state and work can
0376  1986 ;               continue.
0376  1987 ;           -   completing mount verification for the above.
0376  1988 ;
0376  1989 ;           -   starting mount verification to "remount" a disk which was
0376  1990 ;               automatically dismounted due to a connection breakage.  This
0376  1991 ;               is done by force feeding mount verification one of the
0376  1992 ;               connection permanent CDRP's.
0376  1993 ;           -   completing mount verification for the above.
0376  1994 ;
0376  1995 ;       The first two cases represent pretty much normal mount verification.
0376  1996 ;       At least, they are the kind of mount verification one would expect to
0376  1997 ;       encounter for other disk devices.  The second two cases are a special
0376  1998 ;       usage of mount verification by the disk class driver.
0376  1999 ;
0376  2000 ;       Since the connection lost processing done when mount verification
0376  2001 ;       calls the driver is nearly identical to normal mount verification
0376  2002 ;       processing, all that work is performed within this routine or its
0376  2003 ;       servers.  To fully understand some aspects of this routine, it may be
0376  2004 ;       necessary to review the DU_REVALIDATE_DISKS routine.
0376  2005 ;
0376  2006 ; RWAITCNT Handling:
0376  2007 ;
0376  2008 ;       This routine increments RWAITCNT(ucb) when beginning mount
0376  2009 ;       verification for the first time and decrements it when mount
0376  2010 ;       verification is ended.  While RWAITCNT is non-zero, all I/O requests
0376  2011 ;       are added to the I/O queue hanging off the UCB.  Thus all I/O is
0376  2012 ;       stalled until the mount verification completes.  The mount
0376  2013 ;       verification IRP is not stalled because the start I/O routine handles
0376  2014 ;       it specially.
0376  2015 ;
0376  2016 ;       The code on this page acts as a simple dispatcher for the beginning
0376  2017 ;       and ending mount verification cases.
0376  2018 ;
0376  2019 ; Inputs:
0376  2020 ;
```

```
                     0376  2021 ;       For beginning mount verification:
                     0376  2022 ;
                     0376  2023 ;       R3      IRP address
                     0376  2024 ;       R5      UCB address
                     0376  2025 ;
                     0376  2026 ;       For ending mount verification:
                     0376  2027 ;
                     0376  2028 ;       R3      zero
                     0376  2029 ;       R5      UCB address
                     0376  2030 ;
                     0376  2031 ; Outputs:
                     0376  2032 ;
                     0376  2033 ;       All registers are preserved.
                     0376  2034 ;--
                     0376  2035
                     0376  2036 DU$MOUNTVER:
                     0376  2037
53   D5              0376  2038          TSTL    R3                      ; Is this ending mount ver.?
37   13              0378  2039          BEQL    DU_END_MNTVER           ; Branch if ending.
                     037A  2040 ;        BRB     DU_BEGIN_MNTVER         ; Fall through if beginning.
```

```
                        .SBTTL   - DU_BEGIN_MNTVER - Begin mount verification
                ;++
                ;
                ; DU_BEGIN_MNTVER - Begin mount verification
                ;
                ; Functional Description:
                ;
                ;       This routine is called whenever mount verification needs to present a
                ;       new IRP to the driver to be "remembered" and restarted after mount
                ;       verification completes.  If this is a "normal" mount verification
                ;       operation, the is inserted into the pending I/O request queue, ordered
                ;       by sequence number.  However, if the IRP is one of those permanantly
                ;       attached to a CDDB, it is simply discarded.  In any event, RWAITCNT is
                ;       bumped, unless its already bumped because of mount verification,
                ;       UCB$V_MSCP_MNTVERIP is set.
                ;
                ; Inputs:
                ;
                ;       R3      IRP address
                ;       R5      UCB address
                ;
                ; Implicit inputs:
                ;
                ;       IRP$L_DUTUFLAGS(R3)     CDRP$V_PERM bit set to indicate presence of
                ;                               one of the permanent IRP/CDRP pairs belonging
                ;                               to the connection.
                ;       IRP$L_SEQNUM(R3)        monotonically increasing I/O request sequence
                ;                               number
                ;       UCB$L_IOQ[F/B]L(R5)     header for queue of pending I/O requests for
                ;                               this unit
                ;
                ; Outputs:
                ;
                ;       R0 and R1 are destroyed.
                ;       All other registers are preserved.
                ;
                ; Implicit outputs:
                ;
                ;       UCB$W_RWAITCNT(R5)      incremented if mount verification not already
                ;                               in progress.
                ;       UCB$W_DEVSTS(R5)        UCB$V_MSCP_MNTVERIP bit set
                ;
                ; Side effects:
                ;
                ;       The IRP pointed to by R3 is linked into the pending I/O request queue.
                ;--
                ;
DU_BEGIN_MNTVER:
                ;
50   68 A5  01  08  EF  EXTZV   #UCB$V_MSCP_MNTVERIP, #1, -     ; Get previous mount ver. bit
                        UCB$W_DEVSTS(R5), R0                    ; for an upcomming sanity check.
  03 68 A5  08  E2  BBSS    #UCB$V_MSCP_MNTVERIP, -             ; Set MSCP mount ver. in
                        UCB$W_DEVSTS(R5), 15$                   ; progress and branch if already
        56 A5  B6  INCW     UCB$W_RWAITCNT(R5)                  ; set.  Else, bump wait count.
1B 00A0 C3  03  E0  15$:    BBS     #CDRP$V_PERM, -             ; Branch if permanent CDRP
                        IRP$L_DUTUFLAGS(R3), 50$                ; used to start special
                                                                ; purpose mount verification?
```

```
                     038E  2099          ; "Normal" mount verification
                     038E  2100
     51   4C A5   9E 038E  2101          MOVAB   <UCB$L_IOQFL-IRP$L_IOQFL>(R5), -; Get pending I/O queue
                     0392  2102                  R1                              ; listhead address.
        50   51   D0 0392  2103          MOVL    R1, R0                          ; Copy listhead address.
                     0395  2104
        50   60   D0 0395  2105 20$:     MOVL    IRP$L_IOQFL(R0), R0             ; Link to next pending IRP.
        51   50   D1 0398  2106          CMPL    R0, R1                          ; Reached end of list yet?
             07   13 039B  2107          BEQL    40$                             ; Branch if end of list.
  50 A3   50 A0   D1 039D  2108          CMPL    IRP$L_SEQNUM(R0), -             ; Does new IRP belong here?
                     03A2  2109                  IRP$L_SEQNUM(R3)
             F1   1F 03A2  2110          BLSSU   20$                             ; Branch if IRP doesn't belong.
                     03A4  2111
     04 B0   63   0E 03A4  2112 40$:     INSQUE  (R3), @IRP$L_IOQBL(R0)          ; Insert new IRP between
                     03A8  2113                                                  ; previous and current IRPs in
                     03A8  2114                                                  ; pending I/O queue.
             05      03A8  2115          RSB                                     ; Return.
                     03A9  2116
                     03A9  2117          ; Special purpose mount verification
                     03A9  2118
        01 50   E8 03A9  2119 50$:     BLBS    R0, 999$                        ; Branch if nested mount ver.
             05      03AC  2120          RSB                                     ; Otherwise, exit.
                     03AD  2121
                     03AD  2122 999$:    BUG_CHECK DISKCLASS, FATAL              ; Initating special mount ver.
                     03B1  2123                                                  ; when mount ver. is already
                     03B1  2124                                                  ; in progress is a no no.
```

```
03B1  2126          .SBTTL  - DU_END_MNTVER - End mount verification
03B1  2127 ;++
03B1  2128 ;
03B1  2129 ; DU_END_MNTVER - End mount verification
03B1  2130 ;
03B1  2131 ; Functional Description:
03B1  2132 ;
03B1  2133 ;       This routine completes mount verification processing.
03B1  2134 ;
03B1  2135 ;       The mount verification in progress flag is cleared and the wait count
03B1  2136 ;       is reduce one to offset its being raised one when mount verification
03B1  2137 ;       was initiated.  If mount verification failed to bring the disk back
03B1  2138 ;       online, all requests are aborted with SS$_VOLINV to indicate that the
03B1  2139 ;       volume is no longer valid.
03B1  2140 ;
03B1  2141 ;       Next, the status of the wait count and the wait count bumped bit must
03B1  2142 ;       be made appropriate for the status of the current connection to the
03B1  2143 ;       MSCP server.  Mount verification, failover, and reconnection processing
03B1  2144 ;       all occur independently of each other.  However, at this point they
03B1  2145 ;       must be coordinated.  If reconnection processing is in progress the
03B1  2146 ;       wait count must be bumped by at least one and the wait count bumped
03B1  2147 ;       bit must be set.  If reconnection processing is not in progress, the
03B1  2148 ;       wait count must not be bumped and the wait count bumped bit must be
03B1  2149 ;       clear.  During a reconnection attempt, the wait count must be bumped
03B1  2150 ;       from the beginning of the attempt to the end of the attempt, and this
03B1  2151 ;       routine, since it represents the last driver routine in a mount
03B1  2152 ;       verification attempt and since mount verification is the primary
03B1  2153 ;       failover tool, must coordinate handling of the wait count.
03B1  2154 ;       Reconnection modification of the wait count is handled in
03B1  2155 ;       DU$CONNECT_ERR and END_SINGLE_STREAM.
03B1  2156 ;
03B1  2157 ;       If mount verification succeded in bringing the disk back online,
03B1  2158 ;       this routine performs the end of mount verification processing
03B1  2159 ;       necessary to stall attempting to single step CDRPs active when a
03B1  2160 ;       connection failed until all disks (or UCBs) on a given CDDB are made
03B1  2161 ;       valid by mount verification.  This scheme is detailed in the
03B1  2162 ;       description of DU_REVALIDATE_DISKS.
03B1  2163 ;
03B1  2164 ; Inputs:
03B1  2165 ;
03B1  2166 ;       R5      UCB address
03B1  2167 ;
03B1  2168 ; Implicit inputs:
03B1  2169 ;
03B1  2170 ;       UCB$W_RWAITCNT(R5)      count of the reasons for stalling I/O requests
03B1  2171 ;                               (zero implies requests can be restarted)
03B1  2172 ;       UCB$L_IOQ[F/B]L(R5)     header for queue of pending I/O requests for
03B1  2173 ;                               this unit
03B1  2174 ;       UCB$L_STS(R5)           bit UCB$V_VALID clear if mount verification
03B1  2175 ;                               failed
03B1  2176 ;       UCB$L_WAIT_CDDB(R5)     address of a CDDB waiting for mount
03B1  2177 ;                               verification to complete, or zero if none
03B1  2178 ;
03B1  2179 ; Outputs:
03B1  2180 ;
03B1  2181 ;       R0 through R4 are destroyed.
03B1  2182 ;       All other registers are preserved.
```

```
                        03B1  2183 ;
                        03B1  2184 ; Implicit outputs:
                        03B1  2185 ;
                        03B1  2186 ;       UCB$W_RWAITCNT(R5)      decremented
                        03B1  2187 ;       UCB$W_DEVSTS(R5)        UCB$V_MSCP_MNTVERIP bit cleared.
                        03B1  2188 ;       UCB$L_STS(R5)           UCB$V_SUPMVMSG bit cleared.
                        03B1  2189 ;       UCB$L_WAIT_CDDB(R5)     zeroed
                        03B1  2190 ;--
                        03B1  2191
                        03B1  2192 DU_END_MNTVER:
                        03B1  2193
64 A5   00040000 BF  CA 03B1  2194          BICL    #UCB$M_SUPMVMSG, UCB$L_STS(R5)  ; Make next mount ver. noisy.
   68 A5   0100 BF  AA 03B9  2195          BICW    #UCB$M_MSCP_MNTVERIP, -         ; Clear mount verification
                        03BF  2196                  UCB$W_DEVSTS(R5)                ; in progress bit.
           56 A5     B7 03BF  2197          DECW    UCB$W_RWAITCNT(R5)              ; Reduce wait count.
                        03C2  2198
      24 64 A5     0B  E1 03C2  2199          BBC     #UCB$V_VALID, UCB$L_STS(R5), 90$; Branch if mount ver. failed.
                        03C7  2200
                        03C7  2201          ; Mount verification was successful.
                        03C7  2202
    00000000'GF     16  03C7  2203          JSB     G^SCS$UNSTALLUCB                ; If possible, start pending
                        03CD  2204
 53   00DC C5     D0  03CD  2205 70$:       MOVL    UCB$L_WAIT_CDDB(R5), R3         ; Get wait CDDB address.
               16    13  03D2  2206          BEQL    80$                             ; Branch if none.
          00DC C5    D4  03D4  2207          CLRL    UCB$L_WAIT_CDDB(R5)             ; Else, clear waiting CDDB.
            5E A3    B7  03D8  2208          DECW    CDDB$W_WTUCBCTR(R3)             ; Decrement waiting UCB count.
               0D    12  03DB  2209          BNEQ    80$                             ; Branch if UCBs still waiting.
08 12 A3     08    E5  03DD  2210          BBCC    #CDDB$V_RSTRTWAIT, -            ; Branch if connection is not
                        03E2  2211                  CDDB$W_STATUS(R3), 80$          ; waiting to restart CDRPs.
            55    DD  03E2  2212          PUSHL   R5                              ; Else, save UCB.
          0829    30  03E4  2213          BSBW    RESTART_NEXT_CDRP               ; Begin single steping CDRPs.
            55  8ED0  03E7  2214          POPL    R5                              ; Restore UCB.
                        03EA  2215
               05    03EA  2216 80$:       RSB                                     ; Exit.
                        03EB  2217
                        03EB  2218          ; Mount verification was not successful.
                        03EB  2219
          FC12'    30  03EB  2220 90$:       BSBW    DUTU$TERMINATE_PENDING          ; Terminate all pending I/O.
            DD    11  03EE  2221          BRB     70$                             ; Rejoin end mount ver. code.
```

```
                        .SBTTL  DU$DSE_FDT - Data Security Erase FDT Routine
                ;++
                ;
                ; DU$DSE_FDT - Data Security Erase FDT Routine
                ;
                ; Functional Description:
                ;
                ;       This is the FDT routine for the Data Security Erase operation
                ;       supported by MSCP speaking devices.  The byte count (P2) is stored in
                ;       IRP$L_BCNT.  The starting logical block (P3) is stored in IRP$L_MEDIA.
                ;       Control is transfered to EXE$QIODRVPKT, thus queueing the I/O request
                ;       to the driver's start I/O routine.
                ;
                ; Inputs:
                ;
                ;       R3      IRP address
                ;       P2(AP)  byte count
                ;       P3(AP)  starting logical block
                ;
                ; Implicit inputs: None.
                ;
                ; Outputs:
                ;
                ;       IRP$L_BCNT(R3)  <== P2(AP) <byte count>
                ;       IRP$L_MEDIA(R3) <== P3(AP) <starting logical block>
                ;
                ; Implicit outputs: None.
                ;
                ; Condition codes: None.
                ;
                ; Side effects:
                ;
                ;       Control is transfered to EXE$QIODRVPKT, thus queueing the I/O request
                ;       to the driver's start I/O routine.
                ;--
                ;
                DU$DSE_FDT:
                ;
32 A3   04 AC   D0      MOVL    P2(AP), IRP$L_BCNT(R3)          ; Setup erase byte count.
38 A3   08 AC   D0      MOVL    P3(AP), IRP$L_MEDIA(R3)         ; Setup erase starting LBN
  00000000'GF   17      JMP     G^EXE$QIODRVPRT                 ; Send request to STARTIO.
```

```
0400  2265          .SBTTL  DU$SHADOW_PACKACK_FDT - Packack FDT for Volume Shadowing
0400  2266 ;++
0400  2267 ;
0400  2268 ; DU$SHADOW_PACKACK_FDT - Packack FDT for Volume Shadowing
0400  2269 ;
0400  2270 ; Functional Description:
0400  2271 ;
0400  2272 ;       This is one of the FDT routines used by the disk class driver for the
0400  2273 ;       pack acknowledge (IO$_PACKACK) function.  It processes all requests
0400  2274 ;       relating in some way to shadowing.  There are two possible types of
0400  2275 ;       requests; requests which constitute a shadow set, and requests which
0400  2276 ;       bring a shadow set master unit online.
0400  2277 ;
0400  2278 ;       Requests to constitute a shadow set MUST ALWAYS be identified by the
0400  2279 ;       presence of the IO$M_SHADOW modifier.  For such requests, the legality
0400  2280 ;       of the request is tested.  For example, a shadow set master unit may
0400  2281 ;       not be a member of another shadow set, and only controllers which
0400  2282 ;       support shadowing may form shadow sets.  If the shadow set formation
0400  2283 ;       request is illegal, SS$_ILLIOFUNC status is returned.  Otherwise,
0400  2284 ;       shadow volume set specific information is copied to the IRP where it
0400  2285 ;       will be available to be copied into the MSCP command packet by the
0400  2286 ;       start I/O routine.
0400  2287 ;
0400  2288 ;       The following information is copied to the IRP:
0400  2289 ;           -   the unit number to be assigned (used) for the pseudo-unit
0400  2290 ;               represnting the shadow volume set, P4.  This unit number may
0400  2291 ;               or may not have the MSCP$M_SHADOW bit set, to indicate that
0400  2292 ;               the unit is a shadow unit.  This routine will set it
0400  2293 ;               automatically.
0400  2294 ;           -   the starting LBN and block count of the area to be excluded
0400  2295 ;               from any catchup copy, P3 and P2 respectively.
0400  2296 ;           -   the catchup copy speed, P5.  This is one of the values
0400  2297 ;               specified for the MSCP ONLINE command COPY SPEED parameter.
0400  2298 ;
0400  2299 ;       Requests to bring a shadow set master online have two variants which
0400  2300 ;       are identified by whether or not the IO$M_SHADOW function modifier is
0400  2301 ;       set.  If IO$M_SHADOW is set and the shadow unit number, P4, matches
0400  2302 ;       the device's MSCP unit number, then this is a legal request to bring a
0400  2303 ;       shadow set master online.  Otherwise, it is an attempt to include a
0400  2304 ;       shadow set master as a member of another shadow set, which is illegal.
0400  2305 ;       If IO$M_SHADOW is clear by the MSCP unit number indicates that this is
0400  2306 ;       a shadow set master, the IO$M_SHADOW bit must be set for proper
0400  2307 ;       processing to occur within the MSCP server.  Therefore, this routine
0400  2308 ;       sets IO$M_SHADOW and prepares the IRP for a proper MSCP shadow online
0400  2309 ;       request.
0400  2310 ;
0400  2311 ;       This routine does not terminate FDT processing.  The FDT must contain
0400  2312 ;       a second entry for PACKACK which follows the entry for this routine
0400  2313 ;       and which posts the IRP for delivery to the start I/O routine.
0400  2314 ;
0400  2315 ; Inputs:
0400  2316 ;
0400  2317 ;       R3      IRP address
0400  2318 ;       R5      UCB address
0400  2319 ;       P2(AP)  exclusion area block count
0400  2320 ;       P3(AP)  starting logical block number for the exclusion area
0400  2321 ;       P4(AP)  shadow unit number
```

```
                     0400  2322 ;       P5(AP)  copy speed
                     0400  2323 ;
                     0400  2324 ;       IRP$W_FUNC(R3) I/O function code and modifiers
                     0400  2325 ;
                     0400  2326 ; Implicit inputs:
                     0400  2327 ;
                     0400  2328 ;       UCB$L_CDDB(R5)              CDDB address
                     0400  2329 ;       CDDB$W_CNTRLFLGS(CDDB)      MSCP$V_CF_SHADW set to indicate controller
                     0400  2330 ;                                   supports shadowing
                     0400  2331 ;       UCB$W_MSCPUNIT(R5)          MSCP unit number; MSCP$V_SHADOW being set
                     0400  2332 ;                                   implies that this is a shadow set master and
                     0400  2333 ;                                   not a candidate for shadowing
                     0400  2334 ;
                     0400  2335 ; Outputs:
                     0400  2336 ;
                     0400  2337 ;       R0 and R1 are destroyed.
                     0400  2338 ;       All other registers are preserved.
                     0400  2339 ;
                     0400  2340 ;       IRP$L_BCNT(R3)    <== P2(AP) <exclusion area block count>
                     0400  2341 ;       IRP$L_MEDIA(R3)   <== P3(AP) <exclusion area starting LBN>
                     0400  2342 ;       IRP$L_MEDIA+4(R3) <== P4(AP) .or. MSCP$M_SHADOW <shadow unit number>
                     0400  2343 ;       IRP$W_BOFF(R3)    <== P5(AP) <copy speed>
                     0400  2344 ;
                     0400  2345 ;       N.B. the IRP field usage conventions used above are effective solely
                     0400  2346 ;       within this driver.
                     0400  2347 ;
                     0400  2348 ; Implicit outputs: None.
                     0400  2349 ;
                     0400  2350 ; Condition codes:
                     0400  2351 ;
                     0400  2352 ;       SS$_ILLIOFUNC   returned if specified unit cannot be a shadow set
                     0400  2353 ;                       member
                     0400  2354 ;
                     0400  2355 ; Side effects:
                     0400  2356 ;
                     0400  2357 ;       This routine exits with an RSB.  This means that another FDT entry for
                     0400  2358 ;       PACKACK must exist following the entry for this routine in the FDT.
                     0400  2359 ;--
                     0400  2360
                     0400  2361 DU$SHADOW_PACKACK_FDT:
                     0400  2362
        2A 20 A3  06 E1   0400  2363         BBC     #IO$V_SHADOW, -                 ; Is this a shadow packack?
                     0405  2364                 IRP$W_FUNC(R3), 50$                 ; Branch if not shadow packack.
51   0C AC  8000 BF A9   0405  2365         BISW3   #MSCP$M_SHADOW, P4(AP), R1      ; Get QIO shadow unit number.
                     040C  2366         ASSUME  MSCP$V_SHADOW EQ 15
        50  00D4 C5 B0   040C  2367         MOVW    UCB$W_MSCPUNIT(R5), R0          ; Get MSCP unit number.
                  15 19   0411  2368         BLSS    40$                             ; Branch if shadow set master.
                     0413  2369
     32 A3   04 AC 3C   0413  2370         MOVZWL  P2(AP), IRP$L_BCNT(R3)          ; Plant exc. area block count.
     38 A3   08 AC D0   0418  2371         MOVL    P3(AP), IRP$L_MEDIA(R3)         ; Plant exc. area starting LBN.
     30 A3   10 AC B0   041D  2372         MOVW    P5(AP), IRP$W_BOFF(R3)          ; Plant catchup copy speed.
        3C A3   51 B0   0422  2373         MOVW    R1, IRP$L_MEDIA+4(R3)           ; Plant shadow master unit no.
                  18 11   0426  2374         BRB     70$                             ; Exit routine.
                     0428  2375
              51  50 B1   0428  2376 40$:    CMPW    R0, R1                          ; Is QIO shadow master right?
                  09 13   042B  2377         BEQL    55$                             ; If right, process request.
                  1C 11   042D  2378         BRB     99$                             ; Else, error.
```

```
                        042F  2379
                        042F  2380 50$:    ASSUME  MSCP$V_SHADOW EQ 15
 50   00D4 C5   B0      042F  2381          MOVW    UCB$W_MSCPUNIT(R5), R0          ; Get MSCP unit number.
            14   18      0434  2382          BGEQ    80$                             ; Exit routine if not a
                        0436  2383 55$:    ASSUME  IRP$L_BCNT EQ IRP$W_BOFF+2      ; shadow set master.
       32 A3   7C      0436  2384          CLRQ    IRP$W_BCNT(R3)                  ; Else, initialize IRP for a
       38 A3   D4      0439  2385          CLRL    IRP$L_MEDIA(R3)                 ; shadow set master online.
    3C A3   50   B0     043C  2386          MOVW    R0, IRP$L_MEDIA+4(R3)           ; Plant shadow master unit no.
                        0440  2387
 50   00BC C5   D0      0440  2388 70$:    MOVL    UCB$L_CDDB(R5), R0              ; Get CDDB address.
01 28 A0   01   E1      0445  2389          BBC     #MSCP$V_CF_SHADW, -             ; Does controller shadow?
                        044A  2390                  CDDB$W_CNTRLFLGS(R0), 99$       ; Branch if no shadowing.
                        044A  2391
            05   05      044A  2392 80$:    RSB                                     ; Continue FDT processing.
                        044B  2393
                        044B  2394 ; ERROR - this unit cannot be a member of a shadow set.
                        044B  2395
 50   00F4 8F   3C      044B  2396 99$:    MOVZWL  #SS$_ILLIOFUNC, R0              ; Set error status.
  00000000'GF   17      0450  2397          JMP     G^EXE$FINISHIOC                 ; Complete I/O request.
```

```
                        0456  2399          .SBTTL  START I/O
                        0456  2400          .SBTTL  - Out of main line code
                        0456  2401
                        0456  2402 ; Handle device which is in mount verification.
                        0456  2403 ;
                        0456  2404 ;   R3 UCB address
                        0456  2405 ;   R5 CDRP address
                        0456  2406
                        0456  2407 START_MOUNT_VER:
                        0456  2408
                        0456  2409          ; This is a request from mount verification.  Since mount verification
                        0456  2410          ; guarantees that the volume is valid before presenting requests to
                        0456  2411          ; the drivers, UCB$V_VALID being clear is cause for immediately
                        0456  2412          ; terminating the request with SS$_VOLINV.  Presumably, mount
                        0456  2413          ; verification will pickup the pieces and retry the mount verification
                        0456  2414          ; operation from the top.  To avoid unnecessary hangs, the conditions
                        0456  2415          ; which would prevent the IRP from being processed are tested.  If any
                        0456  2416          ; such condition exists, an attempt to failover to the alternate path
                        0456  2417          ; will be made.  If that succeeds, the request will be processed on
                        0456  2418          ; the alternate path.  If the failover fails, the IRP will be
                        0456  2419          ; immediately completed with a SS$_MEDOFL status.
                        0456  2420
      32 64 A3   0B  E1 0456  2421          BBC     #UCB$V_VALID, -         ; In mount verification, valid bit
                        045B  2422                  UCB$L_STS(R3), 199$     ; should always be set.
            28 A5  D4   045B  2423          CLRL    CDRP$L_RWCPTR(R5)       ; Clear wait count pointer.
   50   68 A3   01  08  EF   045E  2424          EXTZV   #UCB$V_MSCP_MNTVERIP, - ; Get mount verification in progress
                        0464  2425                  #1, UCB$W_DEVSTS(R3), R0; bit.
               2A  13   0464  2426          BEQL    999$                    ; It had better be one.
      02 68 A3   0A  E1 0466  2427          BBC     #UCB$V_MSCP_WAITBMP, -  ; Is wait count bumped?
                        046B  2428                  UCB$W_DEVSTS(R3), 10$   ; Branch if wait count not bumped.
               50  B6   046B  2429          INCW    R0                      ; Increment expected wait count.
         56 A3   50  B1   046D  2430 10$:    CMPW    R0, UCB$W_RWAITCNT(R3)  ; Is wait count as expected?
               0A  12   0471  2431          BNEQ    90$                     ; Branch if not as expected.
   50   00BC C3   D0   0473  2432          MOVL    UCB$L_CDDB(R3), R0      ; Get CDDB address.
0E 12 A0   07  E1   0478  2433          BBC     #CDDB$V_NOCONN, -       ; Is there a connection?
                        047D  2434                  CDDB$W_STATUS(R0), 105$ ; Branch if connection present.
         FB80'  30   047D  2435 90$:     BSBW    DUTU$FAILOVER_UCB       ; Try failover to alternate path.
      08 50     E8   0480  2436          BLBS    R0, 105$                ; Branch if failover succeeded.
         51     D4   0483  2437          CLRL    R1                      ; Init second IOSB longword.
                        0485  2438          ALT_REQCOM                      ; Complete mount ver. request now.
                        048B  2439
         77     11   048B  2440 105$:    BRB     DU_RESTARTIO            ; Branch assists.
       0110     31   048D  2441 199$:    BRW     VOLNOTVAL
                        0490  2442
                        0490  2443
                        0490  2444 999$:    BUG_CHECK DISKCLASS, FATAL      ; Received mount verification IRP
                        0494  2445                                          ; when mount verification was not
                        0494  2446                                          ; in progress.  Somebody blew it.
                        0494  2447
                        0494  2448 ; Handle device with I/O requests stalled.
                        0494  2449 ;
                        0494  2450 ;   R3 UCB address
                        0494  2451 ;   R5 CDRP address
                        0494  2452
                        0494  2453          ; For any one of several reasons requests on this UCB are stalled.
                        0494  2454          ; If this is a mount verification, presumably mount verification is in
                        0494  2455          ; progress which would stall this UCB.  So go process the mount
```

```
                                0494  2456          ; verfication IRP.
                                0494  2457          ;
                                0494  2458          ; Otherwise, determine whether or not there is any hope that this IRP
                                0494  2459          ; will work when the UCB becomes unstalled.  If mount verification is
                                0494  2460          ; in progress, the volume is still valid, or the request if a physical
                                0494  2461          ; function, then assume there is hope.  Otherwise, terminate the
                                0494  2462          ; request now with SS$_VOLINV.
                                0494  2463          ;
                                0494  2464          ;   R3 UCB address
                                0494  2465          ;   R5 CDRP address
                                0494  2466
                                0494  2467 IO_STALLED:
                                0494  2468
BD CA A5    0D    E0            0494  2469          BBS     #IRP$V_MVIRP, -         ; Is this a mount verification IRP?
                                0499  2470                  CDRP$W_STS(R5), -       ; Branch if mount verification IRP.
                                0499  2471                  START_MOUNT_VER
1C 68 A3    08    E0            0499  2472          BBS     #UCB$V_MSCP_MNTVERIP, - ; Is mount verification in progress?
                                049E  2473                  UCB$W_DEVSTS(R3), -     ; Branch if mount verification is in
                                049E  2474                  QUEUE_IRP               ; progress.
17 64 A3    0B    E0            049E  2475          BBS     #UCB$V_VALID, -         ; Is the volume still valid?
                                04A3  2476                  UCB$L_STS(R3), -        ; Branch if volume is valid.
                                04A3  2477                  QUEUE_IRP
12 68 A3    0C    E0            04A3  2478          BBS     #UCB$V_MSCP_PKACK, -    ; Branch if a PACKACK is in progress.
                                04A8  2479                  UCB$W_DEVSTS(R3), -
                                04A8  2480                  QUEUE_IRP
0A CA A5    08    E1            04A8  2481          BBC     #IRP$V_PHYSIO, -        ; Volume not valid:  branch if this
                                04AD  2482                  CDRP$W_STS(R5), 90$     ; request is not a physical function.
 50    00BC C3    D0            04AD  2483          MOVL    UCB$L_CDDB(R3), R0      ; Get CDDB address.
03 12 A0    07    E1            04B2  2484          BBC     #CDDB$V_NOCONN, -       ; Is there a connection?
                                04B7  2485                  CDDB$W_STATUS(R0), -    ; Branch if connection present.
                                04B7  2486                  QUEUE_IRP
            00E6    31          04B7  2487 90$:     BRW     VOLNOTVAL               ; Else, kill the request now.
                                04BA  2488
                                04BA  2489 QUEUE_IRP:
50 B3    A0 A5    0E            04BA  2490          INSQUE  CDRP$L_IOQFL(R5), -     ; Queue IRP (based upon CDRP address).
                                04BF  2491                  @UCB$L_IOQBL(R3)
                    05          04BF  2492          RSB                             ; Then discontinue processing on this
                                04C0  2493                                          ; request.
                                04C0  2494
                                04C0  2495 ; Handle class driver reference to a locally connected device
                                04C0  2496 ;
                                04C0  2497 ;   R3  IRP address
                                04C0  2498 ;   R5  UCB address
                                04C0  2499
                                04C0  2500 START_LOCAL_DEVICE:
55    00A8 C5    D0            04C0  2501          MOVL    UCB$L_2P_ALTUCB(R5), R5 ; Get local UCB address.
  50     38 A3    D0            04C5  2502          MOVL    IRP$L_MEDIA(R3), R0     ; Get logical block address.
 00000000'GF    16            04C9  2503          JSB     G^IOC$CVTLOGPHY         ; Convert it to a physical address.
 00000000'GF    17            04CF  2504          JMP     G^EXE$INSIOQ            ; Go hand this IRP to local driver.
```

```
                                04D5  2506          .SBTTL  - Start I/O entry point
                                04D5  2507 DU_STARTIO:
                                04D5  2508          ASSUME  UCB$V_BSY GE 8
          65 A5   01    8A      04D5  2509          BICB    #<UCB$M_BSY @ -8>, -    ; Undo bit setting so that multiple
                                04D9  2510                  UCB$W_STS+1(R5)         ; IRP's can be started.
                                04D9  2511
                                04D9  2512          ; If this UCB indicates that the device is a local (non-MSCP) device
                                04D9  2513          ; that has also been made available to us via 1) dual porting and
                                04D9  2514          ; 2) an MSCP server on the node to which it is dual ported, then
                                04D9  2515          ; shunt this IRP to the local driver.
                                04D9  2516
          3C A5   08    D3      04D9  2517          BITL    #DEV$M_CDP, -           ; Is this a class driver path to a
                                04DD  2518                  UCB$L_DEVCHAR2(R5)      ; local device?
                  E1    12      04DD  2519          BNEQ    START_LOCAL_DEVICE      ; Branch if such a path.
                                04DF  2520
                                04DF  2521          ; Initialize CDRP fields.
                                04DF  2522
          50    60 A3   9E      04DF  2523          MOVAB   IRP$L_FQFL(R3), R0      ; Get address of CDRP portion of IRP.
             53    55   D0      04E3  2524          MOVL    R5, R3                  ; Move UCB address.
             55    50   D0      04E6  2525          MOVL    R0, R5                  ; Move CDRP address.
                                04E9  2526          ASSUME  CDRP$B_CD_TYPE EQ CDRP$W_CDRPSIZE+2
                                04E9  2527          ASSUME  CDRP$B_FIPL    EQ CDRP$W_CDRPSIZE+3
    08 A5   0839FFA0 8F   D0      04E9  2528          MOVL    #< <IPL$_SCSA24> -      ; Initialize CDRP size, type and fork
                                04F1  2529                   ! <DYN$C_CDRP@16> -    ; IPL fields.
                                04F1  2530                   ! <CDRP$L_IOQFL&^xFFFF> >, -
                                04F1  2531                  CDRP$W_CDRPSIZE(R5)
                                04F1  2532          ASSUME  CDRP$L_RSPID    EQ      CDRP$L_MSG_BUF+4
          1C A5   7C      04F1  2533          CLRQ    CDRP$L_MSG_BUF(R5)      ; Prevent spurious DEALLOC_MSG_BUF and
                                04F4  2534                                      ; spurious DEALLOC_RSPID calls.
          2C A5   D4      04F4  2535          CLRL    CDRP$L_LBUFH_AD(R5)     ; Prevent spurious UNMAP calls.
                                04F7  2536          ASSUME  CDRP$W_DUTUCNTR  EQ <CDRP$L_DUTUFLAGS+4>
                                04F7  2537          ASSUME  CDRP$W_ENDMSGSIZ EQ <CDRP$L_DUTUFLAGS+6>
          40 A5   7C      04F7  2538          CLRQ    CDRP$L_DUTUFLAGS(R5)    ; Clear flags, counter, & msg. size.
    28 A5   56 A3   9E      04FA  2539          MOVAB   UCB$W_RWAITCNT(R3), -   ; Point CDRP field to wait counter
                                04FF  2540                  CDRP$L_RWCPTR(R5)       ; in the UCB.
                                04FF  2541
          56 A3   B5      04FF  2542          TSTW    UCB$W_RWAITCNT(R3)      ; Are "normal" I/O requests stalled?
             90   12      0502  2543          BNEQ    IO_STALLED              ; Branch if requests are stalled.
                                0504  2544
                                0504  2545 DU_RESTARTIO:                       ; Label where we RESTART CDRP's after
                                0504  2546                                      ;  virtual circuit re-CONNECTION.
                                0504  2547
          00C8 C3   D0      0504  2548          MOVL    UCB$L_CDT(R3),-         ; Place CDT pointer into CDRP for handy
            24 A5           0508  2549                  CDRP$L_CDT(R5)          ;  reference by SCS routines. Note we
                                050A  2550                                      ;  do this after label DU_RESTARTIO so
                                050A  2551                                      ;  that it is refreshed upon restart.
    54    0084 C3   D0      050A  2552          MOVL    UCB$L_PDT(R3),R4        ; R4 => port's PDT.
                                050F  2553
       40 A5    04   D3      050F  2554          BITL    #CDRP$M_ERLIP, -        ; When set, this bit means an operation
                                0513  2555                  CDRP$L_DUTUFLAGS(R5)    ; which replaced a block is being
                  09   12      0513  2556          BNEQ    10$                     ; restarted and that the previously
                                0515  2557                                      ; allocated RSPID should continue to
                                0515  2558                                      ; be used.
          20 A5   D4      0515  2559          CLRL    CDRP$L_RSPID(R5)        ; Prevent spurious DEALLOC_RSPID.
                                0518  2560          ALLOC_RSPID             ; ALLOCate a ReSPonse ID.
    64 A3   0800 8F   B3      051E  2561 10$:     BITW    #UCB$M_VALID, -         ; Is volume software valid?
                                0524  2562                  UCB$W_STS(R3)
```

```
                     67   13  0524  2563          BEQL    VOL_INVALID             ; Branch if not volume software valid.
                                0526  2564
                                0526  2565 PHYIO_VOLINV:
                                0526  2566          ALLOC_MSG_BUF                 ; Allocate an MSCP buffer (and also
                                0529  2567                                        ;  allocate a unit of flow control).
                   5E 50        E9  0529  2568          BLBC    R0,MSG_BUF_FAILURE ; If failure, branch out of line.
                                052C  2569
                                052C  2570 ; Here a little common MSCP packet initialization.
                                052C  2571
                50   52        D0  052C  2572          MOVL    R2, R0              ; Copy message buffer address.
                                052F  2573          .REPEAT MSCP$K_MXCMDLEN / 8
                                052F  2574          CLRQ    (R0)+               ; Zero entire message buffer.
                     80        7C  052F  2575          .ENDR
                     80        D4  0537  2576          .IIF    NE MSCP$K_MXCMDLEN & 4, CLRL (R0)+
                                0539  2577          .IIF    NE MSCP$K_MXCMDLEN & 2, CLRW (R0)+
                                0539  2578          .IIF    NE MSCP$K_MXCMDLEN & 1, CLRB (R0)+
                                0539  2579
                   20 A5        D0  0539  2580          MOVL    CDRP$L_RSPID(R5),-  ; Use RSPID as command reference
                      62            053C  2581                  MSCP$L_CMD_REF(R2)  ;  number for all commands.
                                053D  2582
                00D4 C3         B0  053D  2583          MOVW    UCB$W_MSCPUNIT(R3),- ; Indicate UNIT number in MSCP
                     04 A2            0541  2584                  MSCP$W_UNIT(R2)     ;  packet.
                                0543  2585
                                0543  2586 DU_BEGIN_IVCMD:
             51      C0 A5      B0  0543  2587          MOVW    CDRP$W_FUNC(R5), R1 ; Get function code & modifiers.
                                0547  2588          ASSUME  IO$V_FCODE EQ 0
          50   51   FFC0 8F     AB  0547  2589          BICW3   #^C<IO$M_FCODE>, R1, R0 ; Extract just the I/O function code.
                                054D  2590
                                054D  2591          DISPATCH R0, type=B, prefix=IO$_, < -       ; Dispatch to correct
                                054D  2592                  <NOP,           START_NOP>, -       ; function processing.
                                054D  2593                  <UNLOAD,        START_UNLOAD>, -
                                054D  2594                  <PACKACK,       START_PACKACK>, -
                                054D  2595                  <WRITECHECK,    START_WRITECHECK>, -
                                054D  2596                  <WRITEPBLK,     START_WRITEPBLK>, -
                                054D  2597                  <READPBLK,      START_READPBLK>, -
                                054D  2598                  <AVAILABLE,     START_AVAILABLE>, -
                                054D  2599                  <DSE,           START_DSE> -
                                054D  2600                  >
                                057D  2601 ; ----- BRB     START_NOSUCH                ; Illegal function code
```

```
                     057D  2603          .SBTTL  - More out of main line code
                     057D  2604 ;++
                     057D  2605 ;
                     057D  2606 ; Handle an illegal I/O function.
                     057D  2607 ;
                     057D  2608 ; Since a message buffer has been allocated, DUTU$RESTORE_CREDIT must be
                     057D  2609 ; called to restore the allocated send credit.  Then the request can be
                     057D  2610 ; finished off with SS$_ILLIOFUNC.
                     057D  2611 ;--
                     057D  2612
                     057D  2613 START_NOSUCH:
          FA80'  30  057D  2614          BSBW    DUTU$RESTORE_CREDIT     ; Restore allocated send credit.
    50 00F4  8F  3C  0580  2615          MOVZWL  #SS$_ILLIOFUNC,R0
          51  D4  0585  2616          CLRL    R1
        03CD  31  0587  2617          BRW     FUNCTION_EXIT           ; Branch to exit I/O function.
                     058A  2618
                     058A  2619
                     058A  2620 ;++
                     058A  2621 ;
                     058A  2622 ; Handle a message buffer allocation failure.
                     058A  2623 ;
                     058A  2624 ; Control get here whenever there is a message buffer allocation failure.
                     058A  2625 ; This implies that the connection to the MSCP server is broken.  The action
                     058A  2626 ; to be taken is to kill this thread of execution by placing the CDRP at the
                     058A  2627 ; tail of the outstanding CDRP queue.  Because of the connection failure, a
                     058A  2628 ; thread exists that is currently executing that is gathering all CDRP's
                     058A  2629 ; associated with this connection for a future restart attempt.  Therefore,
                     058A  2630 ; control is pased to DUTU$KILL_THIS_THREAD which will handle this error.
                     058A  2631 ;--
                     058A  2632
                     058A  2633 MSG_BUF_FAILURE:
                     058A  2634
          FA73'  31  058A  2635          BRW     DUTU$KILL_THIS_THREAD   ; Branch to where we collect all active
                     058D  2636                                          ;  CDRP's prior to re-CONNECTION.
                     058D  2637
                     058D  2638 ;++
                     058D  2639 ;
                     058D  2640 ; Handle volume which is not software valid.
                     058D  2641 ;
                     058D  2642 ; When a volume is not software valid, only physical opeations are legal.  All
                     058D  2643 ; other operations must produce a SS$_VOLINV status.  The only exception is a
                     058D  2644 ; small race window where the volume can become invalid while a thread is
                     058D  2645 ; waiting for a RSPID.  In this case, a check for a packack which might make
                     058D  2646 ; things better is required.
                     058D  2647 ;--
                     058D  2648
                     058D  2649 VOL_INVALID:
          08  E0  058D  2650          BBS     #IRP$V_PHYSIO,-         ; See if PHYSICAL I/O requested.
       CA A5          058F  2651                  CDRP$W_STS(R5),-        ;  If physical, then branch back to
          94          0591  2652                  PHYIO_VOLINV            ;  continue even tho VOLINV.
       56 A3  B5  0592  2653          TSTW    UCB$W_RWAITCNT(R3)      ; Is the wait count raised?
          09  13  0595  2654          BEQL    VOLNOTVAL               ; Branch if wait count not raised.
                     0597  2655          DEALLOC_RSPID                   ; Else, release RSPID.
        FEF4  31  059D  2656          BRW     IO_STALLED              ; Join the I/O is stalled code path.
                     05A0  2657 VOLNOTVAL:
    50 0254  8F  3C  05A0  2658          MOVZWL  #SS$_VOLINV,R0          ; Indicate error status.
          51  D4  05A5  2659          CLRL    R1                      ; Clear second word of I/O status.
```

```
03AD   31  05A7  2660          BRW     FUNCTION_EXIT          ; GOTO common exit.
           05AA  2661
           05AA  2662 ;  End of out of line code
           05AA  2663 ;-
```

```
                        .SBTTL  START_NOP
                ; START_NOP - Prepare an MSCP packet to do a GET UNIT STATUS command.
                ;
                ; INPUTS:
                ;       R1 => I/O function code & modifiers
                ;       R2 => MSCP buffer
                ;       R3 => UCB
                ;       R4 => PDT
                ;       R5 => CDRP
                ;
                ;       The MSCP packet contains zeros except for:
                ;
                ;       MSCP$L_CMD_REF(R2)      <== CDRP$L_RSPID(R5)
                ;       MSCP$W_UNIT(R2)         <== UCB$W_MSCPUNIT(R3)
                ;
                START_NOP:
03      90      MOVB    #MSCP$K_OP_GTUNT,-      ; Transfer GET UNIT STATUS opcode
08 A2           MSCP$B_OPCODE(R2)               ;  to packet.

                SEND_MSCP_MSG                   ; Send message to the MSCP server.

                DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                ACTION_ENTRY    SUCC,  SS$_NORMAL,      NOP_SUCC
                ACTION_ENTRY    OFFLN, SS$_DEVOFFLINE,  NOP_OFFLINE
                ACTION_ENTRY    AVLBL, SS$_MEDOFL,      NOP_AVAIL
                ACTION_ENTRY    DRIVE, SS$_DRVERR,      NOP_DRVERR
                ACTION_ENTRY    CNTLR, SS$_CTRLERR,     NOP_CTRLERR
                ACTION_ENTRY    ICMD,  SS$_CTRLERR,     NOP_IVCMD
                ACTION_ENTRY    SHST,  SS$_SHACHASTA,   NOP_SSSC
                ACTION_ENTRY    END_TABLE

0878    31      BRW     INVALID_STS             ; Unexpected MSCP end status.

                NOP_IVCMD:
                        IVCMD_BEGIN             ; Begin invalid command processing.
08 A2   03  90  MOVB    #MSCP$K_OP_GTUNT, -     ; Setup Get Unit Status opcode
                MSCP$B_OPCODE(R2)               ; in duplicate MSCP command.
                IVCMD_END                       ; Complete invalid command processing.
                ; ----- BRB     NOP_SUCC        ; Fall through to complete command.

                NOP_SUCC:
                NOP_OFFLINE:
                NOP_AVAIL:
                NOP_CTRLERR:
                NOP_DRVERR:
                NOP_SSSC:
  51    D4      CLRL    R1                      ; Clear for I/O status block.
036D    31      BRW     FUNCTION_EXIT           ; Branch to common exit.
```

```
                                05EA  2717          .SBTTL  START_PACKACK
                                05EA  2718
                                05EA  2719 ; START_PACKACK - Prepare an MSCP packet to do an ONLINE command.
                                05EA  2720 ;
                                05EA  2721 ; INPUTS:
                                05EA  2722 ;       R1 => I/O function code & modifiers
                                05EA  2723 ;       R2 => MSCP buffer
                                05EA  2724 ;       R3 => UCB
                                05EA  2725 ;       R4 => PDT
                                05EA  2726 ;       R5 => CDRP
                                05EA  2727 ;
                                05EA  2728 ;       The MSCP packet contains zeros except for:
                                05EA  2729 ;
                                05EA  2730 ;       MSCP$L_CMD_REF(R2)      <== CDRP$L_RSPID(R5)
                                05EA  2731 ;       MSCP$W_UNIT(R2)         <== UCB$W_MSCPUNIT(R3)
                                05FA  2732 ;
                                05EA  2733
                                05EA  2734 START_PACKACK:
                                05EA  2735
 03 68 A3    0C    E2           05EA  2736          BBSS    #UCB$V_MSCP_PKACK, -    ; Branch if PACKACK in progress is
                                05EF  2737                  UCB$W_DEVSTS(R3), 5$    ; set and set it.
        56 A3    B6           05EF  2738          INCW    UCB$W_RWAITCNT(R3)      ; Bump wait count to stall I/O.
                                05F2  2739
    08 A2    09    90           05F2  2740 5$:      MOVB    #MSCP$K_OP_ONLIN, -     ; Transfer ONLINE opcode
                                05F6  2741                  MSCP$B_OPCODE(R2)       ;  to packet.
                                05F6  2742
        00E0 C3    B0           05F6  2743          MOVW    UCB$W_UNIT_FLAGS(R3),-  ; Copy unit flags to MSCP packet.
          0E A2                 05FA  2744                  MSCP$W_UNT_FLGS(R2)
                                05FC  2745
 1C A2    00D8 C3    D0           05FC  2746          MOVL    UCB$L_MSCPDEVPARAM(R3), - ; Copy Device dependent parameters to
                                0602  2747                  MSCP$L_DEV_PARM(R2)     ;  MSCP packet.
                                0602  2748
    18 51    06    E1           0602  2749          BBC     #IO$V_SHADOW, R1, 10$   ; Branch if not shadow packack.
 20 A2    DC A5    B0           0606  2750          MOVW    CDRP$L_MEDIA+4(R5), -   ; Copy shadow device pseudo-unit unit
                                060B  2751                  MSCP$W_SHDW_UNT(R2)     ; number.
 18 A2    D2 A5    B0           060B  2752          MOVW    CDRP$L_BCNT(R5), -      ; Plant exc. area block count.
                                0610  2753                  MSCP$W_EXCL_LBC(R2)
 14 A2    D8 A5    D0           0610  2754          MOVL    CDRP$L_MEDIA(R5), -     ; Plant exc. area starting LBN.
                                0615  2755                  MSCP$L_EXCL_LBA(R2)
 22 A2    D0 A5    B0           0615  2756          MOVW    CDRP$W_BOFF(R5), -      ; Plant catchup copy speed.
                                061A  2757                  MSCP$W_COPY_SPD(R2)
    0A A2    10    A8           061A  2758          BISW    #MSCP$M_MD_SHDSP, -     ; Set the shadow unit specified
                                061E  2759                  MSCP$W_MODIFIER(R2)     ; modifier.
                                061E  2760
                                061E  2761 10$:     IF_IVCMD then=PKAK_IVCMD_END    ; Branch if invalid command processing.
                                0622  2762
                                0622  2763          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                                0625  2764
                                0625  2765          ASSUME  UCB$V_VALID GE 8
    65 A3    08    8A           0625  2766          BICB    #<UCB$M_VALID @ -8>, -  ; Initialize software volume valid.
                                0629  2767                  UCB$W_STS+1(R3)
                                0629  2768
                                0629  2769          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                                062C  2770          ACTION_ENTRY    SUCC,  SS$_NORMAL,      PACKACK_SUCC
                                0631  2771          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      PACKACK_OFFLINE
                                0636  2772          ACTION_ENTRY    ABRTD, SS$_ABORT,       END_PACKACK
                                063B  2773          ACTION_ENTRY    DRIVE, SS$_DRVERR,      END_PACKACK
```

```
                     0640  2774          ACTION_ENTRY    MFMTE, SS$_FORMAT,      END_PACKACK
                     0645  2775          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     PACKACK_IVCMD
                     064A  2776          ACTION_ENTRY    DATA,  SS$_FORMAT,      END_PACKACK
                     064F  2777          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     END_PACKACK
                     0654  2778          ACTION_ENTRY    SHST,  SS$_SHACHASTA,   END_PACKACK
                     0659  2779          ACTION_ENTRY    END_TABLE
                     065B  2780
           07F6   31 065B  2781          BRW     INVALID_STS             ; Unexpected MSCP end status.
                     065E  2782
                     065E  2783 PKAK_IVCMD_END:
           009F   31 065E  2784          BRW     PACKACK_IVCMD_END       ; Branch assist
                     0661  2785
                     0661  2786 PACKACK_SUCC:
                     0661  2787
                     0661  2788          ; At this point, R0 is assumed to contain SS$_NORMAL and
                     0661  2789          ; the online is assumed to have completed successfully.
                     0661  2790
                     0661  2791          ; Test for otherwise undetected errors during ONLINE command.
                     0661  2792
6E 00E0 C3   02   E0 0661  2793          BBS     #MSCP$V_UF_576, -       ; Is disk 576 bytes/sector?
                     0667  2794                  UCB$W_UNIT_FLAGS(R3), - ; Branch if disk is 576 bytes/sector.
                     0667  2795                  PACKACK_576
                     0667  2796          ASSUME  CDRP$V_CAND EQ 0
      71 40 A5   E8  0667  2797          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                     066B  2798                  PACKACK_CANCEL          ; Branch if request was canceled.
                     066B  2799
                     066B  2800          ; Having done a successful ONLINE command; the MSCP end packet status
                     066B  2801          ; for the online command is saved in CDRP$W_DUTUCNTR.  Since the
                     066B  2802          ; online was successful, this field is not longer needed to failover
                     066B  2803          ; retries.  The saved end status will be used later to determine
                     066B  2804          ; whether or not to call DU$ONLINE_COMPLETE for hirt processing.  This
                     066B  2805          ; needs to be done only after the device is first brought online.  All
                     066B  2806          ; succeeding onlines can ignore it.  Status information in the ONLINE
                     066B  2807          ; command end message is saved in the UCB, and other valuable
                     066B  2808          ; information is collected using a GET UNIT STATUS command.
                     066B  2809
   44 A5   0A A2   B0 066B  2810          MOVW    MSCP$W_STATUS(R2), -    ; Copy MSCP end status for later
                     0670  2811                  CDRP$W_DUTUCNTR(R5)     ; testing.
           00CF   30 0670  2812          BSBW    RECORD_ONLINE           ; Record ONLINE date in UCB.
                     0673  2813
                     0673  2814          RESET_MSCP_MSG                  ; Setup message buf. etc. for reuse.
        03 08 A2   90 0676  2815          MOVB    #MSCP$K_OP_GTUNT,-      ; Opcode is for GET UNIT STATUS.
                     0678  2816                  MSCP$B_OPCODE(R2)
                     067A  2817          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                     067D  2818
                     067D  2819          IF_MSCP SUCCESS then=PACKACK_GTUNT_SUCC  ; Branch if GTUNT successful.
                     0683  2820          ASSUME  CDRP$V_CAND EQ 0        ; Else, ...
      55 40 A5   E8  0683  2821          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Was I/O request canceled?
                     0687  2822                  PACKACK_CANCEL          ; Branch if request was canceled.
                     0687  2823          RESET_MSCP_MSG                  ; If here, the drive went offline.
           FF5D   31 068A  2824          BRW     START_PACKACK           ; Go try again.
                     068D  2825
                     068D  2826 PACKACK_GTUNT_SUCC:
                     068D  2827
           00D4   30 068D  2828          BSBW    RECORD_UNIT_STATUS      ; Record UNIT STATUS data in UCB.
                     0690  2829
      50   01   D0 0690  2830          MOVL    #SS$_NORMAL, R0         ; The PACKACK has been successful.
```

```
                     0F   E0  0693  2831          BBS     #MSCP$V_UF_REPLC,-      ; Branch around, if controller initiated
                00E0 C3       0695  2832                  UCB$W_UNIT_FLAGS(R3),-  ; bad block replacement since we already
                     1F       0698  2833                  VALID_PACKACK           ; have our status in R0.
            1A 68 A3   0D   E0  0699  2834          BBS     #UCB$V_MSCP_WRTP, -     ; Also branch around if drive is
                               069E  2835                  UCB$W_DEVSTS(R3), -     ; write protected.
                               069E  2836                  VALID_PACKACK
            15 44 A5   08   E0  069E  2837          BBS     #MSCP$V_SC_ALONL, -     ; Also branch around if drive was
                               06A3  2838                  CDRP$W_DUTOCNTR(R5), -  ; already online when MSCP online
                               06A3  2839                  VALID_PACKACK           ; command was issued.
                    F95A'  30  06A3  2840          BSBW    DU$LOCK_HIRT            ; Else grab hold of the HIRT.
                    F957'  30  06A6  2841          BSBW    DU$ONLINE_COMPLETE      ; Make sure no replacement in progress.
                    F954'  30  06A9  2842          BSBW    DU$UNLOCK_HIRT          ; Release HIRT.
                      55   D5  06AC  2843          TSTL    R5                      ; Was CDRP canceled?
                      01   12  06AE  2844          BNEQ    2$                      ; Branch if CDRP not canceled.
                           05  06B0  2845          RSB                             ; Else, kill this fork thread.
                               06B1  2846
                40 A5  51   C8  06B1  2847 2$:     BISL    R1, CDRP$L_DUTUFLAGS(R5); Set ERLIP bit if its on in R1.
                               06B5  2848
                    29 50   E9  06B5  2849          BLBC    R0, PACKACK_BADRCT      ; Branch if something wrong with RCT.
                               06B8  2850
                               06B8  2851 VALID_PACKACK:
                               06B8  2852          ASSUME  UCB$V_VALID GE 8
                65 A3  08   88  06B8  2853          BISB    #<UCB$M_VALID @ -8>, -  ; Set software volume valid.
                               06BC  2854                  UCB$W_STS+1(R3)
                               06BC  2855
                               06BC  2856 END_PACKACK:
                               06BC  2857          ASSUME  UCB$V_MSCP_PKACK GE 8
                      10   8A  06BC  2858          BICB    #<UCB$M_MSCP_PKACK@-8>,-; Clear PACKACK in progress flag.
                    69 A3      06BE  2859                  UCB$W_DEVSTS+1(R3)
                    56 A3  B7  06C0  2860          DECW    UCB$W_RWAITCNT(R3)      ; Unbump wait count.
                      39   BB  06C3  2861          PUSHR   #^M<R0,R3,R4,R5>        ; Save important registers.
                55    53   D0  06C5  2862          MOVL    R3, R5                  ; Setup UCB address.
            00000000'GF    16  06C8  2863          JSB     G^SCS$UNSTALLUCB        ; Unstall I/O requests.
                      39   BA  06CE  2864          POPR    #^M<R0,R3,R4,R5>        ; Restore important registers.
                      51   D4  06D0  2865          CLRL    R1                      ; Clear R1, for I/O status block.
                    0282   31  06D2  2866          BRW     FUNCTION_EXIT           ; Packack function is done.
```

```
                                06D5  2868          .SBTTL  PACKACK Errors
                                06D5  2869 PACKACK_576:
                50   00BC 8F   3C  06D5  2870          MOVZWL  #SS$_FORMAT, R0         ; Encountered 576 bytes/sector disk.
                             0A   11  06DA  2871          BRB     UNDO_ONLINE             ; Get bad disk format error.
                                                                                        ; Negate the successful ONLINE command.
                                06DC  2872
                                06DC  2873 PACKACK_CANCEL:                              ; PACKACK operation canceled.
                     50    2C   3C  06DC  2874          MOVZWL  #SS$_ABORT, R0          ; Set canceled status.
                             05   11  06DF  2875          BRB     UNDO_ONLINE             ; Negate possibly successful ONLINE.
                                06E1  2876
                                06E1  2877 PACKACK_BADRCT:                              ; Encountered a bad RCT.
                50   216C 8F   3C  06E1  2878          MOVZWL  #SS$_BADRCT, R0         ; Set BADRCT error status.
                                06E6  2879
                                06E6  2880 UNDO_ONLINE:                                 ; Must undo the effect of a successful
                                06E6  2881                                              ; MSCP online command.
                  D8 A5   50    D0  06E6  2882          MOVL    R0, CDRP$L_MEDIA(R5)    ; Save function error status.
                                06EA  2883          RESET_MSCP_MSG                      ; Ready message for a new MSCP command.
                  08 A2   08    90  06ED  2884          MOVB    #MSCP$K_OP_AVAIL, -     ; Undo online with available command.
                                06F1  2885                  MSCP$B_OPCODE(R2)
                                06F1  2886          SEND_MSCP_MSG                       ; Sent AVAILABLE to the server.
                     50   D8 A5   D0  06F4  2887          MOVL    CDRP$L_MEDIA(R5), R0    ; Restore function error status.
                             C2   11  06F8  2888          BRB     END_PACKACK             ; Exit PACKACK processing.
                                06FA  2889
                                06FA  2890 PACKACK_IVCMD:
                                06FA  2891          IVCMD_BEGIN                         ; Begin invalid command processing.
                           FEEA   31  06FD  2892          BRW     START_PACKACK           ; Duplicate PACKACK setup.
                                0700  2893 PACKACK_IVCMD_END:
                                0700  2894          IVCMD_END                           ; Complete invalid command processing.
                             B8   11  0702  2895          BRB     END_PACKACK             ; Complete PACKACK operation.
                                0704  2896
                                0704  2897 PACKACK_OFFLINE:
51   0A A2   0B   05   EF       0704  2898          EXTZV   #MSCP$V_ST_SBCOD, -     ; Extract the end message substatus.
                                070A  2899                  #MSCP$S_ST_SBCOD, -
                                070A  2900                  MSCP$W_STATUS(R2), R1   ; Then, dispatch on that substatus:
                                070A  2901          DISPATCH R1, type=W, prefix=MSCP$K_SC_, < -
                                070A  2902                  <UNKNO, 100$>, -        ; Unit unknown to this server
                                070A  2903                  <INOPR, 100$>, -        ; Unit inoperative on this server
                                070A  2904                  <DUPUN,  20$>, -        ; Duplicate unit number
                                070A  2905                  <NOVOL, END_PACKACK> -  ; Run/Stop switch not depressed
                                070A  2906                  >
                50   008C 8F   3C  0718  2907          MOVZWL  #SS$_DRVERR, R0         ; Everything else is a drive error.
                             9D   11  071D  2908          BRB     END_PACKACK             ; Branch to exit PACKACK.
                                071F  2909
                                071F  2910 20$:     ; Duplicate unit
                       F8DE'    30  071F  2911          BSBW    DUTU$SEND_DUPLICATE_UNIT; Send a message to the operator.
                50   21C4 8F   3C  0722  2912          MOVZWL  #SS$_DUPUNIT,R0         ; Return final status.
                             93   11  0727  2913 9$:      BRB     END_PACKACK             ; Exit PACKACK processing.
                                0729  2914
                                0729  2915 100$:    ; Unknown or inoperative unit
                     44 A5   B6  0729  2916          INCW    CDRP$W_DUTUCNTR(R5)     ; Count this retry attempt.
                44 A5   04   B1  072C  2917          CMPW    #4, CDRP$W_DUTUCNTR(R5) ; Are the retries exhausted?
                             8A   1B  0730  2918          BLEQU   END_PACKACK             ; Branch if no more retries.
                40 A5   04   CA  0732  2919          BICL    #CDRP$M_ERLIP, -        ; Since RSPID is going away, clear the
                                0736  2920                  CDRP$L_DUTUFLAGS(R5)    ; errorlog in progress (to get another).
                       F8C7'    30  0736  2921          BSBW    DUTU$DEALLOC_ALL        ; Release all SCS resources.
                       F8C4'    30  0739  2922          BSBW    DUTU$FAILOVER_UCB       ; Failover to secondary path, if any.
                     E8 50      E9  073C  2923          BLBC    R0, 9$                  ; Branch if switch attempt failed.
                           FDC2   31  073F  2924          BRW     DU_RESTARTIO            ; Otherwise, retry the PACKACK.
```

```
                     0742  2926          .SBTTL  PACKACK Support Routines
                     0742  2927
                     0742  2928 ;+
                     0742  2929 ; RECORD_ONLINE - copy data form ONLINE END MESSAGE to UCB.
                     0742  2930 ;
                     0742  2931 ; Inputs:
                     0742  2932 ;       R2 => End Message
                     0742  2933 ;       R3 => UCB
                     0742  2934 ;
                     0742  2935 ; Outputs:
                     0742  2936 ;       R0 is destroyed
                     0742  2937 ;       UCB fields set
                     0742  2938 ;-
                     0742  2939
                     0742  2940 RECORD_ONLINE:
                     0742  2941
  14 A2      7D      0742  2942          MOVQ    MSCP$Q_UNIT_ID(R2),-    ; In the event of success, copy unit
00CC C3              0745  2943                  UCB$Q_UNIT_ID(R3)       ;  characteristics data to UCB.
  1C A2      D0      0748  2944          MOVL    MSCP$L_MEDIA_ID(R2),-   ; Starting with the UNIT ID, followed
008C C3              074B  2945                  UCB$L_MEDIA_ID(R3)      ;  by the media identifier and
   F8AF'     30      074E  2946          BSBW    DUTU$GET_DEVTYPE        ;  device type and
  24 A2      D0      0751  2947          MOVL    MSCP$L_UNT_SIZE(R2),-   ; Then followed by the unit size and
00B0 C3              0754  2948                  UCB$L_MAXBLOCK(R3)
  28 A2      D0      0757  2949          MOVL    MSCP$L_VOL_SER(R2),-    ; Then by the volume serial number.
00EC C3              075A  2950                  UCB$L_DU_VOLSER(R3)
  0E A2      B0      075D  2951          MOVW    MSCP$W_UNT_FLGS(R2),-   ; Copy new unit flags from end packet.
00E0 C3              0760  2952                  UCB$W_UNIT_FLAGS(R3)
                     0763  2953
                     0763  2954          RSB                             ; Return to caller.
```

```
                        0764  2956 ;++
                        0764  2957 ; RECORD_UNIT_STATUS - copy data from GET UNIT STATUS end message to UCB
                        0764  2958 ;
                        0764  2959 ; Functional Description:
                        0764  2960 ;
                        0764  2961 ;       The supplied MSCP end message is analyzed and appropriate fields in
                        0764  2962 ;       the UCB are filled in with information contained in the end message.
                        0764  2963 ;
                        0764  2964 ;       If the end message is shorter than MSCP$K_LEN, it is zero filled
                        0764  2965 ;       to that length.  This compensates for controllers possibly passing
                        0764  2966 ;       some fields back as zeros by returning short messages.  Then various
                        0764  2967 ;       geometry parameters are copied or calculated from the geometry
                        0764  2968 ;       information in the end message.  If the basic cylinders/tracks/sectors
                        0764  2969 ;       information produced by these calulations contains any zeros, a class
                        0764  2970 ;       driver bugcheck is declared.  Finally, the two write-locked bits are
                        0764  2971 ;       tested.  If either is set, the DEV$M_SWL bit is set.  Otherwise, the
                        0764  2972 ;       bit is not set.
                        0764  2973 ;
                        0764  2974 ; Inputs:
                        0764  2975 ;
                        0764  2976 ;       R2      address of MSCP get unit status end message
                        0764  2977 ;       R3      UCB address
                        0764  2978 ;       R5      CDRP address
                        0764  2979 ;
                        0764  2980 ;       CDRP$W_ENDMSGSIZ(R5) size of MSCP end message
                        0764  2981 ;
                        0764  2982 ; Outputs:
                        0764  2983 ;
                        0764  2984 ;       R0 & R1 destroyed
                        0764  2985 ;
                        0764  2986 ;       The following UCB fields are altered:
                        0764  2987 ;
                        0764  2988 ;       UCB$W_DU_LBNPTRK        LBNs per track
                        0764  2989 ;       UCB$W_DU_TRKPGRP        tracks per group
                        0764  2990 ;       UCB$W_DU_GRPPCYL        groups per cylinder
                        0764  2991 ;       UCB$W_DU_RCTSIZE        LBNs in the RCT
                        0764  2992 ;       UCB$B_DU_RBNPTRK        RBNs per track
                        0764  2993 ;       UCB$B_DU_RCTCPYS        number of RCT copies
                        0764  2994 ;       UCB$L_DU_TOTSIZ         LBNs plus RCT blocks
                        0764  2995 ;       UCB$B_SECTORS           sectors per track (must not be zero)
                        0764  2996 ;       UCB$B_TRACKS            tracks per cylinder (must not be zero)
                        0764  2997 ;       UCB$W_CYLINDERS         cylinders per volume (must not be zero)
                        0764  2998 ;--
                        0764  2999
                        0764  3000 RECORD_UNIT_STATUS:
                        0764  3001
          51    46 A5   3C  0764  3002          MOVZWL  CDRP$W_ENDMSGSIZ(R5), R1; Get size of end message received.
       50   30    51 A3   A3  0768  3003          SUBW3   R1, #MSCP$K_LEN, R0     ; Compute size unsent message.
                 0B 15   15  076C  3004          BLEQ    10$                     ; Branch if whole message sent.
                 3C BB   BB  076E  3005          PUSHR   #^M<R2,R3,R4,R5>        ; Else, save favorite registers.
6241    50    00   6E 00   2C  0770  3006          MOVC5   #0, (SP), #0, R0, (R2)[R1] ; Zero-fill unsent message.
                 3C BA   BA  0777  3007          POPR    #^M<R2,R3,R4,R5>        ; Restore saved registers.
                            0779  3008
                   24 A2 B0  0779  3009 10$:    MOVW    MSCP$W_TRACK(R2),-      ; Copy date from end message to UCB.
                 00FC C3     077C  3010                  UCB$W_DU_LBNPTRK(R3)    ; LBN's per track.
                   26 A2 B0  077F  3011          MOVW    MSCP$W_GROUP(R2),-      ; Then tracks per group.
                 00FE C3     0782  3012                  UCB$W_DU_TRKPGRP(R3)
```

```
                 28 A2    B0   0785  3013          MOVW    MSCP$W_CYLINDER(R2),-   ; Groups per cylinder.
               0100 C3          0788  3014                  UCB$W_DU_GRPPCYL(R3)
               00B0 C3    D0    078B  3015          MOVL    UCB$L_MAXBLOCK(R3),-    ; Unit size in LBNs.
               00F0 C3          078F  3016                  UCB$L_DU_USIZE(R3)
                 2C A2    B0    0792  3017          MOVW    MSCP$W_RCT_SIZE(R2),-   ; Size of the RCT in blocks.
               00F8 C3          0795  3018                  UCB$W_DU_RCTSIZE(R3)
                 2E A2    90    0798  3019          MOVB    MSCP$B_RBNS(R2),-       ; RBN's per track.
               00FB C3          079B  3020                  UCB$B_DU_RBNPTRK(R3)
                              079E  3021
                              079E  3022          ASSUME  DEV$V_RCT GE 8
              39 A3   01    88  079E  3023          BISB    #<DEV$M_RCT @ -8>, -    ; Assume this unit has an RCT.
                              07A2  3024                  UCB$L_DEVCHAR+1(R3)
              50   2F A2    9A  07A2  3025          MOVZBL  MSCP$B_RCT_CPYS(R2), R0 ; R0 = # of RCT copies on the unit.
                   04    12  07A6  3026          BNEQ    15$                     ; Branch if unit has an RCT.
              39 A3   01    CA  07A8  3027          BICL    #<DEV$M_RCT @ -8>, -    ; Else, signal that unit has no RCT.
                              07AC  3028                  UCB$L_DEVCHAR+1(R3)
            00FA C3   50    90  07AC  3029 15$:     MOVB    R0, UCB$B_DU_RCTCPYS(R3); Save number of RCT copies on unit.
                              07B1  3030
                50   2C A2    A4  07B1  3031          MULW2   MSCP$W_RCT_SIZE(R2),R0  ; R0 = size of RCT area.
         00F4 C3    50   00B0 C3  C1  07B5  3032          ADDL3   UCB$L_MAXBLOCK(R3),R0,- ; Size of user visible area plus size
                              07BD  3033                  UCB$L_DU_TOTSZ(R3)      ;  of RCT area is total size.
                              07BD  3034
                 24 A2    90  07BD  3035          MOVB    MSCP$W_TRACK(R2),-      ; Init UCB devdepend fields.
                 44 A3        07C0  3036                  UCB$B_SECTORS(R3)       ; Tracksize is number of sectors.
                      42    13  07C2  3037          BEQL    999$                    ; Branch to bugcheck if zero.
                      50    D4  07C4  3038          CLRL    R0                      ; Clear Product.
                 26 A2    A5  07C6  3039          MULW3   MSCP$W_GROUP(R2),-      ; Group size * cylinder size =
              50   28 A2        07C9  3040                  MSCP$W_CYLINDER(R2),R0  ;  # of tracks in a cylinder.
              45 A3    50    90  07CC  3041          MOVB    R0,UCB$B_TRACKS(R3)
                       34    13  07D0  3042          BEQL    999$                    ; Branch to bugcheck if zero.
              50   24 A2    A4  07D2  3043          MULW2   MSCP$W_TRACK(R2),R0     ; * #sectors/track = #sectors/cyl
                       7E    D4  07D6  3044          CLRL    -(SP)                   ; Put UCB$L_MAXBLOCK into low order
                   00B0 C3   DD  07D8  3045          PUSHL   UCB$L_MAXBLOCK(R3)      ;  longword of workarea on stack.
         51    50    8E    50  7B  07DC  3046          EDIV    R0,(SP)+,R0,R1          ; UCB$L_MAXBLOCK/R0 ... quotient -> R0,
                              07E1  3047                                          ;  remainder -> R1, workarea popped off
                              07E1  3048                                          ;  stack.
                       51    D5  07E1  3049          TSTL    R1                      ; Test for any remainder.
                       02    13  07E3  3050          BEQL    25$                     ; EQL means no remainder.
                       50    D6  07E5  3051          INCL    R0                      ; If remainder, round quotient up.
              46 A3    50    B0  07E7  3052 25$:     MOVW    R0,UCB$W_CYLINDERS(R3)  ; Store number of cylinders on disk.
                       19    13  07EB  3053          BEQL    999$                    ; Branch to bugcheck if zero.
                              07ED  3054
                              07ED  3055          ASSUME  MSCP$V_UF_WRTPH GE 8
                              07ED  3056          ASSUME  MSCP$V_UF_WRTPS GE 8
                              07ED  3057          ASSUME  MSCP$V_UF_WRTPD GE 8
                              07ED  3058          ASSUME  UCB$V_MSCP_WRTP GE 8
              69 A3    20    8A  07ED  3059          BICB    #<UCB$M_MSCP_WRTP@-8>, -; Clear class driver write protected
                              07F1  3060                  UCB$W_DEVSTS+1(R3)      ; flag.
                       93    07F1  3061          BITB    #<<MSCP$M_UF_WRTPD -    ; Is the unit data loss,
                              07F2  3062                    !MSCP$M_UF_WRTPH -    ; hardware, or
                              07F2  3063                    !MSCP$M_UF_WRTPS>@-8>,-; software write protected?
                 0F A2    31  07F2  3064                  MSCP$W_UNT_FLGS+1(R2)
                       04    13  07F5  3065          BEQL    50$                     ; Branch if not write protected.
              69 A3    20    88  07F7  3066          BISB    #<UCB$M_MSCP_WRTP@-8>, -; Else, set the class driver write
                              07FB  3067                  UCB$W_DEVSTS+1(R3)      ; protected flag.
                              07FB  3068
            05 0E A2    0E    E1  07FB  3069 50$:     BBC     #MSCP$V_UF_SSMEM, -     ; Branch if unit is not a shadow set
```

```
                0800  3070                    MSCP$W_UNT_FLGS(R2), 60$; member.
00 3C A3   06   E2  0800  3071          BBSS    #DEV$V_SSM, -           ; Else, set shadow set member device
                0805  3072                    UCB$L_DEVCHAR2(R3), 60$ ; characteristic.
                0805  3073
                0805  3074 60$:      RSB                             ; Return to caller w/ status in R0.
                0806  3075
                0806  3076 999$:     BUG_CHECK DISKCLASS, FATAL      ; This bugcheck is here to catch MSCP
                080A  3077                                            ; servers which return invalid
                080A  3078                                            ; geometry information from a
                080A  3079                                            ; Get Unit Status to an ONLINE drive
                080A  3080                                            ; red-handed.
```

```
                     080A  3082          .SBTTL  START_UNLOAD and START_AVAILABLE
                     080A  3083
                     080A  3084 ; START_AVAILABLE - Prepare an MSCP packet to do an AVAILABLE command without
                     080A  3085 ;       the spindown modifier.
                     080A  3086 ;
                     080A  3087 ; START_UNLOAD - Prepare an MSCP packet to do an AVAILABLE command with
                     080A  3088 ;       spindown specified.
                     080A  3089 ;
                     080A  3090 ; INPUTS:
                     080A  3091 ;       R1 => I/O function code & modifiers
                     080A  3092 ;       R2 => MSCP buffer
                     080A  3093 ;       R3 => UCB
                     080A  3094 ;       R4 => PDT
                     080A  3095 ;       R5 => CDRP
                     080A  3096 ;
                     080A  3097 ;       The MSCP packet contains zeros except for:
                     080A  3098 ;
                     080A  3099 ;       MSCP$L_CMD_REF(R2)      <== CDRP$L_RSPID(R5)
                     080A  3100 ;       MSCP$W_UNIT(R2)         <== UCB$W_MSCPUNIT(R3)
                     080A  3101 ;
                     080A  3102
                     080A  3103 START_UNLOAD:
                     080A  3104
   02    B0          080A  3105          MOVW    #MSCP$M_MD_SPNDW,-      ; Specify the SPINDOWN bit in the
 0A A2                080C  3106                  MSCP$W_MODIFIER(R2)     ;  modifier word.
                      080E  3107
                      080E  3108 START_AVAILABLE:
                      080E  3109
   08    90           080E  3110          MOVB    #MSCP$K_OP_AVAIL,-      ; Transfer AVAILABLE opcode
 08 A2                0810  3111                  MSCP$B_OPCODE(R2)       ;  to packet.
                      0812  3112
                      0812  3113          IF_IVCMD then=AVAIL_IVCMD_END   ; Branch if invalid command processing.
                      0816  3114
                      0816  3115          SEND_MSCP_MSG                   ; Send message to the MSCP server.
                      0819  3116
                      0819  3117          ASSUME  UCB$V_VALID GE 8
 65 A3    08    8A     0819  3118          BICB    #<UCB$M_VALID @ -8>, -  ; Clear software volume valid.
                      081D  3119                  UCB$W_STS+1(R3)
                      081D  3120
                      081D  3121          DO_ACTION       NONTRANSFER     ; Decode MSCP end status.
                      0820  3122          ACTION_ENTRY    SUCC,  SS$_NORMAL,      AVAILABLE_SUCC
                      0825  3123          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      AVAILABLE_MEDOFL
                      082A  3124          ACTION_ENTRY    ABRTD, SS$_ABORT,       AVAILABLE_ABORT
                      082F  3125          ACTION_ENTRY    DRIVE, SS$_DRVERR,      AVAILABLE_DRVERR
                      0834  3126          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     AVAILABLE_CTRLERR
                      0839  3127          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     AVAIL_IVCMD
                      083E  3128          ACTION_ENTRY    SHST,  SS$_SHACHASTA,   AVAILABLE_SSSC
                      0843  3129          ACTION_ENTRY    END_TABLE
                      0845  3130
             060C     31   0845  3131          BRW     INVALID_STS             ; Unexpected MSCP end status.
                      0848  3132
                      0848  3133 AVAIL_IVCMD:
                      0848  3134          IVCMD_BEGIN                     ; Begin invalid command processing.
             FCF5     31   084B  3135          BRW     DU_BEGIN_IVCMD          ; Rebuild failing MSCP packet.
                      084E  3136 AVAIL_IVCMD_END:
                      084E  3137          IVCMD_END                       ; Complete invalid command processing.
                      0850  3138 ; ----- BRB     AVAILABLE_SUCC          ; Fall through to complete available
```

```
                     0850  3139                                          ; operation.
                     0850  3140
                     0850  3141
                     0850  3142 AVAILABLE_SUCC:                         ; Action routine for MSCP$K_ST_SUCC.
                     0850  3143 AVAILABLE_MEDOFL:                       ; Action routine for MSCP$K_ST_MEDOFL.
                     0850  3144 AVAILABLE_ABORT:                       ; Action routine for MSCP$K_ST_ABORT.
                     0850  3145 AVAILABLE_DRVERR:                      ; Action routine for MSCP$K_ST_DRVERR.
                     0850  3146 AVAILABLE_CTRLERR:                     ; Action routine for MSCP$K_ST_CNTRL.
                     0850  3147 AVAILABLE_SSSC:
00 3C A3   06   E5   0850  3148          BBCC    #DEV$V_SSM, -          ; Since its not longer one, clear the
                     0855  3149                  UCB$L_DEVCHAR2(R3), 10$ ; shadow set member device char.
                     0855  3150 10$:     ASSUME  UCB$V_MSCP_WRTP_GE_8
   69 A3   20   8A   0855  3151          BICB    #<UCB$M_MSCP_WRTPa-8>, -; Also clear class driver write
                     0859  3152                  UCB$W_DEVSTS+1(R3)     ; protected flag.
         51   D4   0859  3153          CLRL    R1                     ; Clear R1, for I/O status block.
      00F9   31   085B  3154          BRW     FUNCTION_EXIT
```

```
                     085E  3156          .SBTTL  START_DSE - Start data security erase
                     085E  3157          .SBTTL  START_WRITEPBLK & START_WRITECHECK - Start write opertions
                     085E  3158          .SBTTL  START_READPBLK - Start read operation
                     085E  3159
                     085E  3160 ; START_READPBLK - Prepare an MSCP packet to do a READ command.
                     085E  3161 ;
                     085E  3162 ; START_WRITEPBLK - Prepare an MSCP packet to do a WRITE command.
                     085E  3163 ;
                     085E  3164 ; START_WRITECHECK - Prepare an MSCP packet to do a COMPARE HOST DATA command.
                     085E  3165 ;
                     085E  3166 ; START_DSE - Prepare an MSCP packet to do a ERASE command.
                     085E  3167 ;
                     085E  3168 ; INPUTS:
                     085E  3169 ;       R1 => I/O function code & modifiers
                     085E  3170 ;       R2 => MSCP buffer
                     085E  3171 ;       R3 => UCB
                     085E  3172 ;       R4 => PDT
                     085E  3173 ;       R5 => CDRP
                     085E  3174 ;
                     085E  3175 ;       The MSCP packet contains zeros except for:
                     085E  3176 ;
                     085E  3177 ;       MSCP$L_CMD_REF(R2)      <== CDRP$L_RSPID(R5)
                     085E  3178 ;       MSCP$W_UNIT(R2)         <== UCB$W_MSCPUNIT(R3)
                     085E  3179 ;
                     085E  3180
                     085E  3181          .ENABLE LSB
                     085E  3182
                     085E  3183 START_DSE:
                     085E  3184
          12   90    085E  3185          MOVB    #MSCP$K_OP_ERASE,-      ; Transfer ERASE opcode to packet.
        08 A2         0860  3186                  MSCP$B_OPCODE(R2)
                      0862  3187          IF_IVCMD then=90$               ; Branch if invalid command processing.
         00A8   31    0866  3188          BRW     AFTER_MAP               ; Else, branch around to common code.
         018C   31    0869  3189 90$:     BRW     DSE_IVCMD_END           ; Branch assist.
                      086C  3190
                      086C  3191 PHYS_IO:                                 ; Out of line code to handle "physical"
                      086C  3192                                          ;  READ operations.  I.e. reads of RCT.
        D8 A5   D1    086C  3193          CMPL    CDRP$L_MEDIA(R5),-      ; Make sure block to be read within
      00F4 C3         086F  3194                  UCB$L_DU_TOTSZ(R3)      ;  limits of disk.
          19   18     0872  3195          BGEQ    3$                      ; GEQ implies NOT on the disk.
        D8 A5   D1    0874  3196          CMPL    CDRP$L_MEDIA(R5),-      ; Make sure block to be read within
      00B0 C3         0877  3197                  UCB$L_MAXBLOCK(R3)      ;  RCT area.
          11   19     087A  3198          BLSS    3$                      ; LSS implies NOT in RCT.
D2 A5  00000200 8F    D1    087C  3199 1$:     CMPL    #512,CDRP$L_BCNT(R5)    ; Reads of RCT should be 1 block only.
          71   18     0884  3200          BGEQ    DO_MAP                  ; GEQ implies only reading one block.
                      0886  3201                                          ;  so we return to inline code.
      50  034C 8F    3C    0886  3202          MOVZWL  #SS$_IVBUFLEN,R0        ; Else indicate error and branch around.
          05   11     088B  3203          BRB     6$
                      088D  3204 3$:
      50  0134 8F    3C    088D  3205          MOVZWL  #SS$_IVADDR,R0          ; Bad address on disk.
                      0892  3206 6$:
          51   D4     0892  3207          CLRL    R1                      ; Clear for I/O status block.
        F769'  30    0894  3208          BSBW    DUTU$RESTORE_CREDIT     ; Restore allocated send credit.
         00BD   31    0897  3209          BRW     FUNCTION_EXIT           ; And goto common function exit.
                      089A  3210
                      089A  3211 START_WRITECHECK:
                      089A  3212
```

```
                 20   90   089A  3213          MOVB    #MSCP$K_OP_COMP,-       ; Compare host data opcode
               08 A2        089C  3214                  MSCP$B_OPCODE(R2)       ;  to packet.
     00F4 C3   D2 A5   D1   089E  3215          CMPL    CDRP$L_BCNT(R5), -      ; Make sure block to WRITECHECK within
                            08A4  3216                  UCB$L_DU_TOTSZ(R3)      ;  limits of disk.
                    E7   18   08A4  3217          BGEQ    3$                      ; GEQ implies NOT on the disk.
     00B0 C3   D2 A5   D1   08A6  3218          CMPL    CDRP$L_BCNT(R5), -      ; See if block to WRITECHECK within
                            08AC  3219                  UCB$L_MAXBLOCK(R3)      ;  RCT area.
                    49   19   08AC  3220          BLSS    DO_MAP                  ; LSS implies NOT in RCT so continue.
                    CC   11   08AE  3221          BRB     1$                      ; If in RCT, go check BCNT.
                              08B0  3222
                              08B0  3223          .DISABLE LSB
                              08B0  3224 ;+
                              08B0  3225 ; Out of line processing for all read and write modifiers.
                              08B0  3226 ;
                              08B0  3227 ;       This code section receives control whenever any of the possible
                              08B0  3228 ;       modifiers for read or write functions are detected to be set in a
                              08B0  3229 ;       single test in the common read/write code below.  Here the individual
                              08B0  3230 ;       modifiers are checked and processed.  This scheme reduces the number
                              08B0  3231 ;       of instructions in the main-line no-modifiers-set code path.
                              08B0  3232 ;-
                              08B0  3233
                              08B0  3234 START_WITH_MODIFIERS:
                              08B0  3235
        05 51      0E   E1   08B0  3236          BBC     #IO$V_DATACHECK, R1, 10$; Branch if data check not specified.
                              08B4  3237          ASSUME  MSCP$V_MD_COMP GE 8     ; Else, set the read/write with
     0B A2    40 8F   88   08B4  3238          BISB    #<MSCP$M_MD_COMP@-8>, - ; data compare modifier.
                              08B9  3239                  MSCP$W_MODIFIER+1(R2)
                              08B9  3240
        04 51      0F   E1   08B9  3241 10$:    BBC     #IO$V_INHRETRY, R1, 20$ ; Branch if retries not inhibited.
                              08BD  3242          ASSUME  MSCP$V_MD_SEREC GE 8    ; Else, set the suppress error
     0B A2      01   88   08BD  3243          BISB    #<MSCP$M_MD_SEREC@-8>, -; modifier.
                              08C1  3244                  MSCP$W_MODIFIER+1(R2)
                              08C1  3245
        0B 51      08   E1   08C1  3246 20$:    BBC     #IO$V_FORCERR, R1, 30$  ; Branch if forced error not specified.
     51    06      00   ED   08C5  3247          CMPZV   #IO$V_FCODE, #IO$S_FCODE, -     ; Is this a write function?
                        0B  08C9  3248                  R1, #IO$_WRITEPBLK
                        04   12   08CA  3249          BNEQ    30$                     ; If not, ignore forced error requests.
                              08CC  3250          ASSUME  MSCP$V_MD_ERROR GE 8    ; Else, set the WRITE with forced
     0B A2      10   88   08CC  3251          BISB    #<MSCP$M_MD_ERROR@-8>, -; modifier.
                              08D0  3252                  MSCP$W_MODIFIER+1(R2)
                              08D0  3253
        1D 51      07   E1   08D0  3254 30$:    BBC     #IO$V_EXPRESS, R1, -    ; Branch if express request not
                              08D4  3255                  TEST_PHYSIO             ; specified.
                              08D4  3256          ASSUME  MSCP$V_MD_EXPRS GE 8    ; Else, set the express request
  0B A2    80 8F   88   08D4  3257          BISB    #<MSCP$M_MD_EXPRS@-8>, -; modifier.
                              08D9  3258                  MSCP$W_MODIFIER+1(R2)
                        16   11   08D9  3259          BRB     TEST_PHYSIO             ; Rejoin common code.
                              08DB  3260
                              08DB  3261 PHYSICAL:                                ; Branch assist.
                        8F   11   08DB  3262                  BRB     PHYS_IO
                              08DD  3263 XFER_IVCMD_END:                          ; Branch assist.
                      010E   31   08DD  3264                  BRW     TRANSFER_IVCMD_END
                              08E0  3265
                              08E0  3266 START_WRITEPBLK:
                              08E0  3267
     08 A2      22   90   08E0  3268          MOVB    #MSCP$K_OP_WRITE, -     ; Setup write MSCP opcode.
                              08E4  3269                  MSCP$B_OPCODE(R2)
```

```
                     04   11  08E4  3270          BRB     START_READWRITE         ; Join common code.
                                08E6  3271
                                08E6  3272 START_READPBLK:
                                08E6  3273
                  08 A2  21  90 08E6  3274          MOVB    #MSCP$K_OP_READ, -      ; Setup read MSCP opcode.
                                08EA  3275                  MSCP$B_OPCODE(R2)
                                08EA  3276
                                08EA  3277 START_READWRITE:
                                08EA  3278
                                08EA  3279                                          ; Test for any special modifiers set:
                51  C180 8F  B3  08EA  3280          BITW    #<IO$M_DATACHECK -      ;       data check
                                08EF  3281                   !IO$M_INHRETRY -       ;       inhibit retries
                                08EF  3282                   !IO$M_FORCERR -        ;       write forced error
                                08EF  3283                   !IO$M_EXPRESS>, R1     ;       express request
                     BF   12    08EF  3284          BNEQ    START_WITH_MODIFIERS    ; Branch if any modifier set.
                                08F1  3285
                                08F1  3286 TEST_PHYSIO:
                                08F1  3287
                                08F1  3288          ASSUME  IRP$V_PHYSIO GE 8
                  CB A5  01  B3 08F1  3289          BITW    #<IRP$M_PHYSIO @ -8>, - ; Is this a physical request?
                                08F5  3290                  CDRP$W_STS+1(R5)
                     E4   12    08F5  3291          BNEQ    PHYSICAL                ; If physical request, get physical.
                                08F7  3292
                                08F7  3293 DO_MAP:
                                08F7  3294          IF_IVCMD then=XFER_IVCMD_END    ; Branch if invalid command processing.
                                08FB  3295
                     30 A5  9E  08FB  3296          MOVAB   CDRP$T_LBUFHNDL(R5),-   ; Put address of Local BUFfer HaNDLe
                     2C A5      08FE  3297                  CDRP$L_LBUFH_AD(R5)     ;  field into field that points to it.
                                0900  3298          MAP_IRP                         ; Allocate mapping resources and load
                                0903  3299                                          ;  them with data from SVAPTE, BOFF,
                                0903  3300                                          ;  and BCNT derived from IRP within
                                0903  3301                                          ;  CDRP.
                                0903  3302
                52   1C A5  D0  0903  3303          MOVL    CDRP$L_MSG_BUF(R5),R2   ; Refresh R2 => MSCP packet.
                     30 A5  7D  0907  3304          MOVQ    CDRP$T_LBUFHNDL(R5),-   ; Copy contents of buffer handle to
                     10 A2      090A  3305                  MSCP$B_BUFFER(R2)       ;  MSCP buffer descriptor field.
                     38 A5  D0  090C  3306          MOVL    CDRP$T_LBUFHNDL+8(R5),- ; Buffer handle is 96 bits (12 bytes)
                     18 A2      090F  3307                  MSCP$B_BUFFER+8(R2)     ;  in length.
                                0911  3308 AFTER_MAP:
                     D2 A5  D0  0911  3309          MOVL    CDRP$L_BCNT(R5),-
                     0C A2      0914  3310                  MSCP$L_BYTE_CNT(R2)     ; Copy byte count of transfer.
                     D8 A5  D0  0916  3311          MOVL    CDRP$L_MEDIA(R5),-
                     1C A2      0919  3312                  MSCP$L_LBN(R2)          ; And also the Logical Block Number.
                                091B  3313
                                091B  3314          SEND_MSCP_MSG   INLINE          ; Send message to the MSCP server.
                                093A  3315
                                093A  3316          ; To make the normal (successful) case go faster, an explicit test for
                                093A  3317          ; a successful MSCP status is made here.  If that test fails,
                                093A  3318          ; an action table will be used to determine the appropriate action.
                                093A  3319          ; If it succeeds, however, we're goin' a git out o' here like a shot.
                                093A  3320
                                093A  3321          IF_MSCP FAILURE then=TRANSFER_MSCP_ERROR  ; Branch if not successful.
                                0940  3322
                                0940  3323          ; Transfer was successful.
                50 00010000 8F  D0  0940  3324          MOVL    #SS$_NORMAL@16, R0      ; Set success status.
                                0947  3325
                                0947  3326 TRANSFER_RTN_BCNT:                       ; Common TRANSFER action routine.
```

```
                        0947  3327
                        0947  3328          MOVL    MSCP$L_BYTE_CNT(R2), R1 ; Get # bytes actually transferred.
                        094B  3329
                        094B  3330 TRANSFER_SHIFT:
                        094B  3331
50   50   F0 8F   79      094B  3332          ASHQ    #-16, R0, R0            ; Shift into proper position for IOSB.
                        0950  3333
   09 A2   80 8F   93   0950  3334          BITB    #MSCP$M_EF_BBLKR, -     ; Has a bad block been reported?
                        0955  3335                  MSCP$B_FLAGS(R2)        ; If so,
         45   12        0955  3336          BNEQ    XFER_REPLACE            ; it must be replaced.
                        0957  3337
                        0957  3338 NORMAL_TRANSFEREND:
                        0957  3339 ; ----- BRW     FUNCTION_EXIT           ; Fall though to function exit.
```

```
                     0957  3341          .SBTTL  FUNCTION_EXIT - Common request completion processing
                     0957  3342 ;++
                     0957  3343 ;
                     0957  3344 ; FUNCTION_EXIT - Common request completion processing
                     0957  3345 ;
                     0957  3346 ; Functional Description:
                     0957  3347 ;
                     0957  3348 ;       If need to log final I/O status call ERL$LOGSTATUS.  Deallocate all
                     0957  3349 ;       SCS resources held by the request.  If necessary perform special
                     0957  3350 ;       single stream processing.  Else, complete the request normally.
                     0957  3351 ;
                     0957  3352 ; Inputs:
                     0957  3353 ;
                     0957  3354 ;       R3      UCB address
                     0957  3355 ;       R4      PDT address
                     0957  3356 ;       R5      CDRP address
                     0957  3357 ;
                     0957  3358 ; Outputs: None.
                     0957  3359 ;--
                     0957  3360
                     0957  3361          .ENABLE LSB
                     0957  3362
                     0957  3363 FUNCTION_EXIT:
                     0957  3364
52   1C A5   D0      0957  3365          MOVL    CDRP$L_MSG_BUF(R5), R2  ; Get end message address.
         0C  13      095B  3366          BEQL    20$                     ; Branch if no end message present.
                     095D  3367
09 A2    20  93      095D  3368          BITB    #MSCP$M_EF_ERLOG, -     ; Was an error log message generated?
                     0961  3369                  MSCP$B_FLAGS(R2)
         24  12      0961  3370          BNEQ    LOG_FINAL_STATUS        ; Branch if error log generated.
40 A5    04  D3      0963  3371          BITL    #CDRP$M_ERLIP, -        ; How about a very old error log msg.,
                     0967  3372                  CDRP$L_DUTUFLAGS(R5)    ; which the server has forgotten?
         1E  12      0967  3373          BNEQ    LOG_FINAL_STATUS        ; Branch if error log generated.
                     0969  3374
40 A5    04  CA      0969  3375 20$:     BICL    #CDRP$M_ERLIP, -        ; Just in case clear bit.
                     096D  3376                  CDRP$L_DUTUFLAGS(R5)
                     096D  3377
   7E    50  7D      096D  3378          MOVQ    R0,-(SP)                ; Save final I/O status on stack.
      F68D' 30      0970  3379          BSBW    DUTU$DEALLOC_ALL        ; Free resources owned by this CDRP.
   50    8E  7D      0973  3380          MOVQ    (SP)+,R0                ; Restore final I/O status.
                     0976  3381
52   00BC C3  D0      0976  3382          MOVL    UCB$L_CDDB(R3), R2      ; Get CDDB address in R2.
   12 A2    01  B3      097B  3383          BITW    #CDDB$M_SNGLSTRM, -     ; Is CDDB in single stream mode?
                     097F  3384                  CDDB$W_STATUS(R2)
         0E  12      097F  3385          BNEQ    SINGLE_STREAM           ; Branch if single stream.
                     0981  3386          ALT_REQCOM                      ; Else, complete the request.
                     0987  3387
                     0987  3388 LOG_FINAL_STATUS:
00000000'GF  16      0987  3389          JSB     G^ERL$LOGSTATUS         ; Go log software status for errorlog.
         DA  11      098D  3390          BRB     20$                     ; Rejoin mainline code.
                     098F  3391
                     098F  3392 SINGLE_STREAM:
         14  BB      098F  3393          PUSHR   #^M<R2,R4>              ; Save CDDB and PDT addresses.
00000000'GF  16      0991  3394          JSB     G^IOC$ALTREQCOM         ; Complete request, but keep control.
         18  BA      0997  3395          POPR    #^M<R3,R4>              ; Restore CDDB and PDT addresses.
       0274  31      0999  3396          BRW     RESTART_NEXT_CDRP       ; Branch to code to restart next CDRP.
                     099C  3397          .DISABLE LSB
```

```
                     099C  3399          .SBTTL  READ/WRITE Error processing
                     099C  3400
                     099C  3401          .ENABLE LSB
                     099C  3402
                     099C  3403  XFER_REPLACE:                          ; A branch assist.
009F   31            099C  3404          BRW     TRANSFER_REPLACE
                     099F  3405
                     099F  3406  ; All data transfer error cases begin their processing here.
                     099F  3407  ;
                     099F  3408  ; CDRP$L_MSG_BUF and R2 point to the END PACKET which was returned by the
                     099F  3409  ; server.
                     099F  3410
                     099F  3411  TRANSFER_MSCP_ERROR:
                     099F  3412
                     099F  3413          DO_ACTION       TRANSFER        ; Decode MSCP end status.
                     09A2  3414          ACTION_ENTRY    SUCC,  SS$_NORMAL,      TRANSFER_RTN_BCNT
                     09A7  3415          ACTION_ENTRY    AVLBL, SS$_MEDOFL,      TRANSFER_MEDOFL
                     09AC  3416          ACTION_ENTRY    OFFLN, SS$_MEDOFL,      TRANSFER_RTN_BCNT
                     09B1  3417          ACTION_ENTRY    DATA,  SS$_FORCEDERROR, TRANSFER_DATA_ERROR
                     09B6  3418          ACTION_ENTRY    DRIVE, SS$_DRVERR,      TRANSFER_RTN_BCNT
                     09BB  3419          ACTION_ENTRY    CNTLR, SS$_CTRLERR,     TRANSFER_RTN_BCNT
                     09C0  3420          ACTION_ENTRY    WRTPR, SS$_WRITLCK,     TRANSFER_RTN_BCNT
                     09C5  3421          ACTION_ENTRY    COMP,  SS$_DATACHECK,   TRANSFER_RTN_BCNT
                     09CA  3422          ACTION_ENTRY    MFMTE, SS$_FORMAT,      TRANSFER_RTN_BCNT
                     09CF  3423          ACTION_ENTRY    ABRTD, SS$_ABORT,       TRANSFER_RTN_BCNT
                     09D4  3424          ACTION_ENTRY    ICMD,  SS$_CTRLERR,     TRANSFER_INVALID_COMMAND
                     09D9  3425          ACTION_ENTRY    HSTBF, SS$_IVBUFLEN,    TRANSFER_HOST_BUFFER_ERROR
                     09DE  3426          ACTION_ENTRY    SHST,  SS$_SHACHASTA,   TRANSFER_RTN_BCNT
                     09E3  3427          ACTION_ENTRY    END_TABLE
                     09E5  3428
046C   31            09E5  3429          BRW     INVALID_STS             ; Unexpected MSCP end status.
                     09E8  3430
                     09E8  3431  TRANSFER_INVALID_COMMAND:
                     09E8  3432
                     09E8  3433          IVCMD_BEGIN                     ; Begin invalid command processing.
FB55   31            09EB  3434          BRW     DU_BEGIN_IVCMD          ; Repeat MSCP command packet setup
                     09EE  3435                                          ; upto but not including the MAP_IRP.
                     09EE  3436                                          ; Then do everything after the
                     09EE  3437  TRANSFER_IVCMD_END:                     ; MAP_IRP here, by hand.
                     09EE  3438          ASSUME  CDRP$S_LBUFHNDL EQ 12
10 A2  30 A5   7D    09EE  3439          MOVQ    CDRP$T_LBUFHNDL(R5), -  ; Copy contents of buffer handle to
                     09F3  3440                  MSCP$B_BUFFER(R2)       ; MSCP buffer descriptor field.
18 A2  38 A5   D0    09F3  3441          MOVL    CDRP$T_LBUFHNDL+8(R5), -
                     09F8  3442                  MSCP$B_BUFFER+8(R2)
                     09F8  3443  DSE_IVCMD_END:
0C A2  D2 A5   D0    09F8  3444          MOVL    CDRP$L_BCNT(R5), -      ; Copy transfer byte count.
                     09FD  3445                  MSCP$L_BYTE_CNT(R2)
1C A2  D8 A5   D0    09FD  3446          MOVL    CDRP$L_MEDIA(R5), -     ; Copy starting logical block number.
                     0A02  3447                  MSCP$L_LBN(R2)
                     0A02  3448
                     0A02  3449          IVCMD_END                       ; Complete invalid command processing.
68 51          E9    0A04  3450          BLBC    R1, 35$                 ; If this is the first time through
                     0A07  3451                                          ; here, try the MSCP command one more
                     0A07  3452                                          ; time.
51             D4    0A07  3453          CLRL    R1                      ; Else, clear "byte count."
FF3F           31    0A09  3454          BRW     TRANSFER_SHIFT          ; And blow this operation away.
                     0A0C  3455
```

```
                        0A0C  3456 TRANSFER_MEDOFL:
                        0A0C  3457
                 06 E1  0A0C  3458          BBC     #MSCP$V_SC_INOPR,-      ; Branch around if NOT unit inoperative
              0A A2     0A0E  3459                  MSCP$W_STATUS(R2),-     ;  substatus.
                 2A     0A10  3460                  XFER_RTN_BCNT
      50 008C0000 8F    0A11  3461          MOVL    #SS$_DRVERR@16,R0       ; Else set up R0 with proper SS$_ code
                        0A18  3462                                          ;  in high order word and
           FF2C 31      0A18  3463          BRW     TRANSFER_RTN_BCNT       ; Branch around.
                        0A1B  3464
                        0A1B  3465 TRANSFER_HOST_BUFFER_ERROR:
                        0A1B  3466
                 05 EF  0A1B  3467          EXTZV   #MSCP$S_ST_MASK,-       ; Extract the sub-code only.
                 0B     0A1D  3468                  #16-MSCP$S_ST_MASK,-
           51 0A A2     0A1E  3469                  MSCP$W_STATUS(R2),R1
              51 02 B1  0A21  3470          CMPW    #MSCP$K_SC_ODDBC,R1     ; Compare to Odd Byte Count error.
                 15 13  0A24  3471          BEQL    XFER_RTN_BCNT           ; Branch around if Odd BCNT.
              042B 31   0A26  3472          BRW     INVALID_STS             ; Here we got an invalid MSCP status.
                        0A29  3473
                        0A29  3474 TRANSFER_DATA_ERROR:                    ; TRANSFER action routine for MSCP$K_ST_DATA
                        0A29  3475                                          ; Here R0 contains SS$_FORCEDERROR in
                        0A29  3476                                          ;  high order word.
           51 0A A2 3C  0A29  3477          MOVZWL  MSCP$W_STATUS(R2),R1    ; Zero extend MSCP status.
                        0A2D  3478
                        0A2D  3479 ;        ASSUME THAT FORCED ERROR SUBCODE IS ZERO.
                        0A2D  3480
      51 51 FB 8F 78    0A2D  3481          ASHL    #-MSCP$S_ST_MASK,R1,R1  ; Shift out major status code.
                 07 13  0A32  3482          BEQL    XFER_RTN_BCNT           ; EQL implies that it was forced error
                        0A34  3483                                          ;  so we simply branch since we already
                        0A34  3484                                          ;  have R0 set up.
      50 01F40000 8F    0A34  3485          MOVL    #SS$_PARITY@16,R0       ; Else set up R0 with proper SS$_ code
                        0A3B  3486                                          ;  in high order word.
                        0A3B  3487 XFER_RTN_BCNT:
           FF09 31      0A3B  3488          BRW     TRANSFER_RTN_BCNT       ; Now, go complete the request.
                        0A3E  3489
                        0A3E  3490 TRANSFER_REPLACE:                       ; Here perform Bad Block Replacement.
                        0A3E  3491
                 05 E1  0A3E  3492          BBC     #MSCP$V_EF_ERLOG,-      ; Branch around if End Message does NOT
              04 09 A2  0A40  3493                  MSCP$B_FLAGS(R2),30$    ;  have ERLOG flag set.
           40 A5 04 C8  0A43  3494          BISL    #CDRP$M_ERLIP, -        ; Else set corresponding bit in CDRP to
                        0A47  3495                  CDRP$L_DUTUFLAGS(R5)    ;  remember this fact.
                        0A47  3496
      3F 68 A3 0D E0    0A47  3497 30$:     BBS     #UCB$V_MSCP_WRTP, -     ; If device is write protected, don't
                        0A4C  3498                  UCB$W_DEVSTS(R3), -     ; even attempt to perform the replace;
                        0A4C  3499                  XFER_NORMALEND          ; it won't work.
           A0 A5 50 7D  0A4C  3500          MOVQ    R0,CDRP$L_IOQFL(R5)     ; Store valid I/O status in unused
                        0A50  3501                                          ;  FLINK and BLINK of IRP portion.
                 F5AD' 30 0A50  3502          BSBW    DU$LOCK_HIRT            ; Grab hold of HIRT.
                 F5AA' 30 0A53  3503          BSBW    DU$REPLACE_LBN          ; Perform the replacement.
                 F5A7' 30 0A56  3504          BSBW    DU$UNLOCK_HIRT          ; Release HIRT.
              52 50 D0  0A59  3505          MOVL    R0, R2                  ; Copy replacement status to R2.
                 55 D5  0A5C  3506          TSTL    R5                      ; Was CDRP canceled?
                 01 12  0A5E  3507          BNEQ    32$                     ; Branch if CDRP not canceled.
                    05  0A60  3508          RSB                             ; Else, kill this thread.
                        0A61  3509
           40 A5 51 A8  0A61  3510 32$:     BISW    R1, CDRP$L_DUTUFLAGS(R5); Set ERLOGIP bit if on in R1.
           50 A0 A5 7D  0A65  3511          MOVQ    CDRP$L_IOQFL(R5),R0     ; Restore I/O status of original QIO.
              1C 52 E9  0A69  3512          BLBC    R2,60$                  ; LBC means unsuccessful replacement.
```

```
                     0A6C  3513
                     0A6C  3514
                     0A6C  3515          BLBS    R0, XFER_NORMALEND      ; If here replacement successful
                     0A6F  3516                                          ; LBS means original I/O was success.
                     0A6F  3517 35$:
                     0A6F  3518          PUSHL   CDRP$L_RSPID(R5)        ; Save current RSPID so as to possibly
                     0A72  3519                                          ;  reuse.
                     0A72  3520          BBS     #CDRP$V_ERLIP, -        ; If set, we reuse RSPID so that error
                     0A77  3521                  CDRP$L_DUTUFLAGS(R5), - ;  log messages co-relate. So we branch.
                     0A77  3522                  40$
                     0A77  3523          CLRL    (SP)                    ; If no errorlog, erase RSPID to prevent
                     0A79  3524          BRB     50$                     ;  spurious deallocates and branch.
                     0A7B  3525 40$:
                     0A7B  3526          CLRL    CDRP$L_RSPID(R5)        ; Prevent current RSPID from being deallocat
                     0A7E  3527 50$:
                     0A7E  3528          BSBW    DUTU$DEALLOC_ALL        ; Free resources owned by this CDRP before
                     0A81  3529                                          ;  retrying operation.
                     0A81  3530          POPL    CDRP$L_RSPID(R5)        ; Restore RSPID or zero to CDRP.
                     0A85  3531          BRW     DU_RESTARTIO            ; Else go to retry the original request.
                     0A88  3532 60$:
                     0A88  3533          MOVW    R2,R0                   ; Replace failed replacement status.
                     0A8B  3534 XFER_NORMALEND:
                     0A8B  3535          BRW     FUNCTION_EXIT           ; And branch to FUNCTION_EXIT.
                     0A8E  3536          .DISABLE        LSB
                     0A8E  3537
                     0A8E  3538 ;; This code previously resided prior to label 24$ above.  It is left here
                     0A8E  3539 ;;       as reference material.
                     0A8E  3540 ;;
                     0A8E  3541 ;        BBC     #IRP$V_PHYSIO,-         ; Branch around if NOT physical I/O.
                     0A8E  3542 ;                CDRP$W_STS(R5),24$
                     0A8E  3543 ;        BBC     #IRP$V_FUNC,-           ; Branch around if NOT read function.
                     0A8E  3544 ;                CDRP$W_STS(R5),24$
                     0A8E  3545 ;
                     0A8E  3546 ;; Here if this is a read physical function.  We must back translate the
                     0A8E  3547 ;;       physical media address to an LBN.
                     0A8E  3548 ;;
                     0A8E  3549 ;; User specified physical media address has following format: (in CDRP$L_MEDIA)
                     0A8E  3550 ;;
                     0A8E  3551 ;;       ,-------------------------------,---------------,---------------,
                     0A8E  3552 ;;       |           Cylinder            |     Track     |     Sector    |
                     0A8E  3553 ;;       |-------------------------------|---------------|---------------|
                     0A8E  3554 ;;
                     0A8E  3555 ;
                     0A8E  3556 ;
                     0A8E  3557 ;        MOVZBL  UCB$B_TRACKS(R3),R0     ; R0 = tracks/cylinder.
                     0A8E  3558 ;        MOVZBL  UCB$B_SECTORS(R3),R1    ; R1 = sectors/track.
                     0A8E  3559 ;        MULL    R1,R0                   ; R0 = sectors/cylinder.
                     0A8E  3560 ;        MOVZWL  CDRP$L_MEDIA+2(R5),-(SP); Push user specified cylinder #.
                     0A8E  3561 ;        MULL    R0,(SP)                 ; (SP) = LBN of 1st sector in cylinder.
                     0A8E  3562 ;        MOVZBL  CDRP$L_MEDIA+1(R5),R0   ; R0 = user specified track #.
                     0A8E  3563 ;        MULL    R0,R1                   ; R1 = relative LBN of 1st sector on track.
                     0A8E  3564 ;        MOVZBL  CDRP$L_MEDIA(R5),R0     ; R0 = user specified sector #.
                     0A8E  3565 ;        ADDL    R0,R1                   ; R1 = relative LBN of user sector.
                     0A8E  3566 ;        ADDL3   R1,(SP)+,MSCP$L_LBN(R2) ; Deposit calculated LBN into packet.
```

```
0A8E  3568          .SBTTL  re-CONNECTION after VC error or failure
0A8E  3569
0A8E  3570 ; DU$CONNECT_ERR - Block of code invoked during the time that we
0A8E  3571 ;       re-CONNECT to the intelligent controller following some disturbance
0A8E  3572 ;       that caused dismanteling of the logical CONNECTION between the
0A8E  3573 ;       class driver and the controller.  The ultimate purpose of the code
0A8E  3574 ;       here is to locate all CDRP's relevant to this controller and place
0A8E  3575 ;       them in the proper order into CDDB$L_RSTRTQFL.  Once
0A8E  3576 ;       all the CDRP's are on this list we "execute" each of these CDRP's, one
0A8E  3577 ;       by one, until they are all done.  When the last such CDRP is completed
0A8E  3578 ;       we resume normal QIO processing.  This code works in cooperation with
0A8E  3579 ;       code in FUNCTION_EXIT.
0A8E  3580 ;
0A8E  3581 ;       We are invoked here either by the Port Driver calling us at our error
0A8E  3582 ;       entry point or by the Disk Class Driver branching here as a result of
0A8E  3583 ;       deciding that the intelligent controller has gone "insane".
0A8E  3584 ;
0A8E  3585 ;       The actions herein taken are the following:
0A8E  3586 ;
0A8E  3587 ;       1.  We disable the Timeout Mechanism Routine wakeups by placing a
0A8E  3588 ;               longword of all 1's in CRB$L_DUETIME.
0A8E  3589 ;
0A8E  3590 ;       2.  In order to prevent new CDRP's from starting up, we increment
0A8E  3591 ;               UCB$W_RWAITCNT for each UCB associated with this controller.
0A8E  3592 ;               This count is used to count the number of CDRP's associated
0A8E  3593 ;               with a UCB that have run into resource wait situations.
0A8E  3594 ;               Whenever this count is non-zero, new CDRP's are automatically
0A8E  3595 ;               backed up onto the UCB$L_IRPQFL queue.  Incrementing this
0A8E  3596 ;               count here, insures that it will not be run to zero and will
0A8E  3597 ;               cause all new CDRP's to backup.
0A8E  3598 ;
0A8E  3599 ;       3.  We deallocate resources owned by the permanent CDRP used by the
0A8E  3600 ;               Timeout Mechanism Routine.
0A8E  3601 ;
0A8E  3602 ;       4.  At the time that we are called here, our active CDRP's can be
0A8E  3603 ;               found in one of the following places:
0A8E  3604 ;
0A8E  3605 ;               a) On the HIRT wait Q.  If here note that the associated UCB
0A8E  3606 ;                  RWAITCNT has been bumped due to being on this list in
0A8E  3607 ;                  addition to the bump given in step 2 above.
0A8E  3608 ;
0A8E  3609 ;               b) On the RDT resource wait Q.  Here also RWAITCNT has been
0A8E  3610 ;                  bumped once to many times.
0A8E  3611 ;
0A8E  3612 ;               c) On the CDDB$L_CDRPQFL.  Here RWAITCNT is normal except for
0A8E  3613 ;                  the bump given in step 2.
0A8E  3614 ;
0A8E  3615 ;               d) On some other resource wait Q (Flow control, message buffer,
0A8E  3616 ;                  mapping resources, etc.). Here again RWAITCNT has been bumped
0A8E  3617 ;                  once to much.
0A8E  3618 ;
0A8E  3619 ;               e) On the CDDB$L_RSTRTQ.  If here, the CONNECTION has failed
0A8E  3620 ;                  while we were in the middle of cleaning up a previous
0A8E  3621 ;                  CONNECTION failure.  The CDRP's here need no further
0A8E  3622 ;                  gathering.
0A8E  3623 ;
0A8E  3624 ;               Our aim here is to gather all the active CDRP's onto the
```

```
0A8E  3625 ;                CDDB$L_RSTRTQ.  To do this we search for them in the above
0A8E  3626 ;                mentioned places in the order in which they were mentioned.
0A8E  3627 ;                This order is important as will be explained below.
0A8E  3628 ;
0A8E  3629 ;        5.  Note here that at the time of the call to DU$CONNECT_ERR, we
0A8E  3630 ;                may have been on the middle of MOUNT VERIFICATION.  In such
0A8E  3631 ;                a case the particular volume would have been marked as
0A8E  3632 ;                invalid and during re-CONNECTION we would not try to bring
0A8E  3633 ;                the unit online.  Also we would have a set of inactive
0A8E  3634 ;                (i.e. no resources allocated for them) CDRP'a (IRP's) on
0A8E  3635 ;                the MOUNT VERIFICATION QUEUE of the UCB and possibly one
0A8E  3636 ;                MOUNT VERIFICATION specific CDRP active.  This all meshes
0A8E  3637 ;                perfectly with our re-CONNECTION design.  The contents of
0A8E  3638 ;                the MOUNT VERIFICATION QUEUE can be ignored.  The active
0A8E  3639 ;                MOUNT VERIFICATION CDRP will be treated normally.  Its
0A8E  3640 ;                I/O will be retried and will probably fail and MOUNT
0A8E  3641 ;                VERIFICATION will re-submit it and it will wind up on the
0A8E  3642 ;                normal UCB I/O QUEUE awaiting the RWAITCNT's going to zero.
0A8E  3643 ;                After re-CONNECTION, it will start up normally and everything
0A8E  3644 ;                should resume transparently.
0A8E  3645 ;
0A8E  3646 ;        6.  First we scan the HIRT wait Q and remove any CDRP's associated
0A8E  3647 ;                with the current CDDB.  We do this first so that if perchance,
0A8E  3648 ;                some of our CDRP's are here, they will not be selected
0A8E  3649 ;                inadvertantly when the current HIRT owner is possibly killed.
0A8E  3650 ;
0A8E  3651 ;                This scan is done by going down the entire HIRT wait Q and
0A8E  3652 ;                removing the 1st entry of ours that we find.  If in a pass
0A8E  3653 ;                we DO remove an entry , then we go back and scan from the
0A8E  3654 ;                start of the Q.  When we make an entire pass without any hits,
0A8E  3655 ;                we finish.  Note that when we remove an entry, we decrement
0A8E  3656 ;                the RWAITCNT prior to calling INSERT_RSTRTQ to undo the bump
0A8E  3657 ;                we gave in calling LOCK_HIRT.
0A8E  3658 ;
0A8E  3659 ;        7.  We scan the RDT resource wait Q.  Again we scan until we find our
0A8E  3660 ;                first entry and after a removal we begin to scan from the
0A8E  3661 ;                beginning.  Only a clean scan ends the process.  Also we
0A8E  3662 ;                must decrement RWAITCNT for each removal.
0A8E  3663 ;
0A8E  3664 ;        8.  We REMQUE each entry on CDDB$L_CDRPQFL and call INSERT_RSTRTQ
0A8E  3665 ;                for each one.
0A8E  3666 ;
0A8E  3667 ;        9.  Here we should note that INSERT_RSTRTQ deallocates all resources
0A8E  3668 ;                owned by a CDRP prior to inserting it in CDDB$L_RSTRTQ.
0A8E  3669 ;                Because of this, the only CDRP's belonging to us that still
0A8E  3670 ;                own RSPID's are the CDRP's which are on other resource wait
0A8E  3671 ;                queues.  So here we scan the RDT looking for entries that
0A8E  3672 ;                belong to us.  When we find one we REMQUE it, decrement its
0A8E  3673 ;                RWAITCNT and call INSERT_RSTRTQ for it.  Note that this
0A8E  3674 ;                deallocates its resources and as a result of this could cause
0A8E  3675 ;                another of our CDRP's to receive these resources and proceed
0A8E  3676 ;                up to the CDDB$L_CDRPQFL.  Therefore after a removal here,
0A8E  3677 ;                we branch back to step 7 to safeguard against this possibility.
0A8E  3678 ;                A complete scan of the RDT with no hits implies that we now
0A8E  3679 ;                have gathered all our CDRP's and that we can continue.
0A8E  3680 ;
0A8E  3681 ;*******************************************************************************
```

```
0A8E  3682 ;       9.  If the two counts above are equal, then we have all CDRP's on
0A8E  3683 ;           CDDB$L_RSTRTQFL.  No more CDRP's will trickle in so we clear
0A8E  3684 ;           CDDB$M_CDRPTRCKL in CDDB$W_STATUS.
0A8E  3685 ;
0A8E  3686 ;       10. We DISCONNECT the now dead connection and then re-CONNECT to
0A8E  3687 ;           establish a new channel to the MSCP server in the controller.
0A8E  3688 ;
0A8E  3689 ;       11. We are now ready to begin single stream execution of CDRPs, until
0A8E  3690 ;           exhaust the contents of the CDRP$L_RSTRTQFL.  However we
0A8E  3691 ;           want to guard against the possibility that a particular
0A8E  3692 ;           request (i.e. CDRP) may repeatedly hang a controller (i.e.
0A8E  3693 ;           cause a re-CONNECTION) and thereby prevent anything from
0A8E  3694 ;           getting through.  To deal with this we only retry a given
0A8E  3695 ;           request a fixed maximum number of times (MAX_RETRY).  The
0A8E  3696 ;           algorithm which resolves this retry logic dilemma relies
0A8E  3697 ;           on several data items in the CDDB:
0A8E  3698 ;
0A8E  3699 ;                   a) CDDB$L_RSTRTCDRP - the address of the CDRP that is
0A8E  3700 ;                       currently being processed in single stream mode if
0A8E  3701 ;                       we are in single stream mode.
0A8E  3702 ;
0A8E  3703 ;                   b) CDDB$B_RETRYCNT - the number of remaining retries
0A8E  3704 ;                       for the current CDRP being processes in single
0A8E  3705 ;                       stream mode if we are in single stream mode.
0A8E  3706 ;
0A8E  3707 ;                   c) CDDB$V_SNGLSTRM - bit in CDDB$W_STATUS which tells
0A8E  3708 ;                       us if we are in single stream mode.
0A8E  3709 ;
0A8E  3710 ;           The algorithm is as follows: If upon selecting the first CDRP
0A8E  3711 ;           on CDDB$L_RSTRTQFL, we find CDDB$V_SNGLSTRM clear, we merely
0A8E  3712 ;           set it and we can be assurred that this is the first time
0A8E  3713 ;           that we are attempting to retry this request in single stream
0A8E  3714 ;           mode.  This is so because the bit being clear implies either
0A8E  3715 ;           that this is the first re-CONNECTION since the system came up
0A8E  3716 ;           or that the last re-CONNECTION ran to completion thereby leaving
0A8E  3717 ;           the bit clear.  In this case we select this first CDRP, set
0A8E  3718 ;           CDDB$B_RETRYCNT to the maximum and establish this CDRP as the
0A8E  3719 ;           current one by storing its address in CDDB$L_RSTRTCDRP.
0A8E  3720 ;
0A8E  3721 ;           If however CDDB$V_SNGLSTRM is set upon selecting a CDRP, we
0A8E  3722 ;           must compare the CDRP address to the current value of
0A8E  3723 ;           CDDB$L_RSTRTCDRP.  If they are NOT equal, then again this is
0A8E  3724 ;           the first retry attempt for this CDRP and we merely set the
0A8E  3725 ;           CDDB$B_RETRYCNT to the maximum and store the CDRP in
0A8E  3726 ;           CDDB$L_RSTRTCDRP.  If the CDRP has the same address however,
0A8E  3727 ;           we must decrement one from the retry count and if it is not
0A8E  3728 ;           exhausted attempt to process the CDRP again.
0A8E  3729 ;
0A8E  3730 ;           Note this all works even though the address of a CDRP is not
0A8E  3731 ;           necessarily unique. That is, many I/O requests in the life of
0A8E  3732 ;           the system may occupy the same CDRP in virtual space.  However,
0A8E  3733 ;           once re-CONNECTION logic begins, it deals only with the CDRPs
0A8E  3734 ;           on the CDDB$L_RSTRTQFL.  This list never grows until re-
0A8E  3735 ;           CONNECTION is run to completion since all new IRPs are
0A8E  3736 ;           being backed up.  Therefore even though we may run repeated
0A8E  3737 ;           re-CONNECTIONs that do not run to completion but rather each
0A8E  3738 ;           causes the connection to go down, through all this the
```

```
                     0A8E  3739 ;                   CDDB$L_RSTRTQFL is always monotonically decreasing and no
                     0A8E  3740 ;                   new CDRPs are entered onto it that were not there at the time
                     0A8E  3741 ;                   that we began to process the first re-CONNECTION.  In a fixed
                     0A8E  3742 ;                   list of CDRPs which all exist at the same time, the address
                     0A8E  3743 ;                   is a unique descriptor.
                     0A8E  3744 ;
                     0A8E  3745 ;           12. Note that CDDB$M_SNGLSTRM in CDDB$W_STATUS acts as a flag to
                     0A8E  3746 ;                   FUNCTION_EXIT so that it can aid in the one at a time re-
                     0A8E  3747 ;                   execution of the CDRP's.
                     0A8E  3748 ;
                     0A8E  3749 ;           13. For debugging sake, we loop thru all UCB's and check that their
                     0A8E  3750 ;                   UCB$W_RWAITCNT values are all equal to 1.
                     0A8E  3751 ;                   Also for debugging sake we check that CDDB$L_CDRPQFL is
                     0A8E  3752 ;                   empty.
                     0A8E  3753 ;
                     0A8E  3754 ;           14. We REMQUE the 1st CDRP on CDDB$L_RSTRTQFL and branch to
                     0A8E  3755 ;                   DU_RESTARTIO to begin its execution.
                     0A8E  3756 ;
                     0A8E  3757 ; Inputs: (for DU$RE_SYNCH)
                     0A8E  3758 ;          R3 => CRB
                     0A8E  3759
                     0A8E  3760
                     0A8E  3761 DU$RE_SYNCH:
                     0A8E  3762
53   10 A3   D0      0A8E  3763          MOVL    CRB$L_AUXSTRUC(R3),R3   ; R3 => CDDB.
54   14 A3   D0      0A92  3764          MOVL    CDDB$L_PDT(R3),R4       ; R4 => PDT.
26 A3   04   91      0A96  3765          CMPB    #MSCP$K_CM_EMULA,-      ; If this is the MSCP server, the right
                     0A9A  3766                  CDDB$B_CNTRLMDL(R3)     ; resynch technique is DISCONNECT.
        0A   13      0A9A  3767          BEQL    RECONN_COMMON           ; So, skip the MRESET setup.
        10   A8      0A9C  3768          BISW    #CDDB$M_RESYNCH,-       ; Signal that we should reset
     12 A3           0A9E  3769                  CDDB$W_STATUS(R3)       ;  intelligent controller.
        04   11      0AA0  3770          BRB     RECONN_COMMON           ; Branch around to common code.
                     0AA2  3771
                     0AA2  3772 ; Inputs: (for DU$CONNECT_ERR)
                     0AA2  3773 ;          R3 => CDT
                     0AA2  3774 ;          R4 => PDT
                     0AA2  3775 ;
                     0AA2  3776
                     0AA2  3777 DU$CONNECT_ERR:
                     0AA2  3778
   53   5C A3   D0      0AA2  3779          MOVL    CDT$L_AUXSTRUC(R3),R3   ; R3 => CDDB.
                     0AA6  3780 RECONN_COMMON:
        3A A3   B6      0AA6  3781          INCW    CDDB$W_RSTRTCNT(R3)     ; Count number times thru here.
                AA      0AA9  3782          BICW    #<CDDB$M_IMPEND -       ; Signal: no immediate command pending
                        0AAA  3783                   !CDDB$M_INITING -      ;         out of initialization
                        0AAA  3784                   !CDDB$M_SNGLSTRM -     ;         no single stream in progress
                        0AAA  3785                   !CDDB$M_RSTRTWAIT>,-   ;         not waiting to restart CDRPs
12 A3   0107 8F         0AAA  3786                  CDDB$W_STATUS(R3)
   50   18 A3   D0      0AAF  3787          MOVL    CDDB$L_CRB(R3),R0       ; R0 => CRB.
   18 A0   01   CE      0AB3  3788          MNEGL   #1,CRB$L_DUETIME(R0)    ; Prevent Timeout Mechanism wakeups.
                        0AB7  3789
        08   A8      0AB7  3790          BISW    #CDDB$M_RECONNECT,-     ; Indicate that we are re-connecting
     12 A3           0AB9  3791                  CDDB$W_STATUS(R3)       ;  state.
                     0ABB  3792
53   0000007C 8F   C3   0ABB  3793          SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB address in R1.
                        0AC2  3794                   -CDDB$L_UCBCHAIN>, -
                     51 0AC2  3795                  R3, R1
```

```
                     0AC3  3796
51   00C4 C1   D0    0AC3  3797 10$:     MOVL    UCB$L_CDDB_LINK(R1), R1 ; Chain to next UCB.
           0A  13    0AC8  3798          BEQL    20$                     ; EQL implies no more UCB's here.
F4 68 A1   0A  E2    0ACA  3799          BBSS    #UCB$V_MSCP_WAITBMP, -  ; If already bumped, branch around;
                     0ACF  3800                  UCB$W_DEVSTS(R1), 10$   ; else indicate RWAITCNT bumped.
        56 A1  B6    0ACF  3801          INCW    UCB$W_RWAITCNT(R1)      ; Prevent new CDRP's from starting up.
           EF  11    0AD2  3802          BRB     10$                     ; Go look for more UCB's.
                     0AD4  3803 20$:
                     0AD4  3804
                     0AD4  3805 ;
                     0AD4  3806 ; Now we are sure that no new CDRP's will start.
                     0AD4  3807 ;
                     0AD4  3808
        F529'  30    0AD4  3809          BSBW    DUTU$DISCONNECT_CANCEL  ; Perform disconnect cancel cleanup.
                     0AD7  3810
                     0AD7  3811 ; Deallocate RSPID & message buffer on each of the CDDB perm. IRP/CDRP pairs.
                     0AD7  3812
55   00D0 C3   9E    0AD7  3813          MOVAB   CDDB$A_PRMCDRP(R3), R5  ; Get permanent CDRP address.
        F521'  30    0ADC  3814          BSBW    DUTU$DEALLOC_RSPID_MSG  ; Deallocate its RSPID & msg. buf.
55   0194 C3   9E    0ADF  3815          MOVAB   CDDB$A_DAPCDRP(R3), R5  ; Get DAP permanent CDRP address.
        F519'  30    0AE4  3816          BSBW    DUTU$DEALLOC_RSPID_MSG  ; Deallocate its RSPID & msg. buf.
                     0AE7  3817
                     0AE7  3818 ;
                     0AE7  3819 ;       Registers here are:
                     0AE7  3820 ;               R3 => CDDB
                     0AE7  3821 ;               R4 => PDT.
                     0AE7  3822 ;               R5 => Permanent CDRP.
                     0AE7  3823 ;
                     0AE7  3824
        F516'  30    0AE7  3825          BSBW    DU$DISCONNECT_HIRT      ; Cleanup HIRT wait queue.
                     0AEA  3826
                     0AEA  3827 ; All CDRPs belonging to this CDDB have been removed from the HIRT wait queue.
                     0AEA  3828
                     0AEA  3829 ; Locate and prepare for restarting all CDRPs currently waiting for a RSPID.
                     0AEA  3830 ; Since the class driver allocates a RSPID as the first step in any function,
                     0AEA  3831 ; CDRPs found now will not be holding any resources and will not be active.
                     0AEA  3832 ; Since these CDRPs hold no resources, their cleanup will not cause any other
                     0AEA  3833 ; waiting requests to become active.  (This fact is not currently used, but it
                     0AEA  3834 ; might be useful.)
                     0AEA  3835
53   00F4 C3   D0    0AEA  3836          MOVL    CDDB$L_CDT(R3), R3              ; Get CDT address.
                     0AEF  3837
        51   D4      0AEF  3838          CLRL    R1                              ; Set SCAN_RSPID_WAIT flag.
                     0AF1  3839          SCAN_RSPID_WAIT -                       ; Use SCS service to scan RSPID
                     0AF1  3840                  action = DUTU$RECONN_LOOKUP     ; wait queue.
                     0AFE  3841                                                  ; DUTU$RECONN_LOOKUP is in
                     0AFE  3842                                                  ; DUTUSUBS.
                     0AFE  3843
                     0AFE  3844 ; Remove all CDRPs on the active requests queue.  These CDRPs:
                     0AFE  3845 ;   a. have outstanding requests in the intelligent controller,
                     0AFE  3846 ;   b. suffered allocation failures due to a broken connection,
                     0AFE  3847 ;   c. represent the request during which an "insane" controller was detected.
                     0AFE  3848 ; In any case, these CDRPs are not on any resource wait queue and do not have
                     0AFE  3849 ; their associated resource wait count bumped due to need for a resource.
                     0AFE  3850
        F4FF'  30    0AFE  3851          BSBW    DUTU$DRAIN_CDDB_CDRPQ           ; Cleanup active requests.
                     0B01  3852
```

```
                        0B01  3853 ; Now scan the entire Response-id Descriptor Table for any remaining CDRPs
                        0B01  3854 ; belonging to this connection.  Presumably these CDRPs are on a resource wait
                        0B01  3855 ; queue somewhere.  In addtion, releasing whatever resources such CDRPs hold
                        0B01  3856 ; may cause other waiting CDRPs to become active.  Therefore, after every CDRP
                        0B01  3857 ; is located and processed, the active CDRP queue must be scanned again.
                        0B01  3858
                51   D6 0B01  3859          INCL    R1                              ; Set SCAN_RDT flag.
                        0B03  3860          SCAN_RDT -                              ; Use SCS service to scan RDT.
                        0B03  3861                  action = DUTU$RECONN_LOOKUP     ; DUTU$RECONN_LOOKUP is in
                        0B10  3862                                                  ; DUTUSUBS.
                        0B10  3863
            53  5C A3   D0 0B10  3864          MOVL    CDT$L_AUXSTRUC(R3), R3          ; Restore the CDDB address.
                        0B14  3865
                        0B14  3866 RESTART_FIRST_CDRP:
                        0B14  3867
                        0B14  3868 ;
                        0B14  3869 ; We come here either by falling thru from above code or by branching here
                        0B14  3870 ;       from CALL_SEND_MSG_BUF when the last CDRP has trickled in.
                        0B14  3871 ;
                        0B14  3872 ;
                        0B14  3873 ; If here all CDRP's are in CDDB$L_RSTRTQFL.  So no more will trickle.
                        0B14  3874 ;       Clear bit that prevents CALL_SEND_MSG_BUF from doing its job.
                        0B14  3875 ;
                        0B14  3876 ; INPUTS:
                        0B14  3877 ;       R3 => CDDB
                        0B14  3878 ;       R4 => PDT
                        0B14  3879 ;
                        0B14  3880
                        0B14  3881
                        0B14  3882 ;
                        0B14  3883 ; Here we DISCONNECT the old connection.
                        0B14  3884 ;
                        0B14  3885
         55   00D0 C3   9E 0B14  3886          MOVAB   CDDB$A_PRMCDRP(R3), R5  ; CDRP into R5 for coming BSBWs.
              50   53   D0 0B19  3887          MOVL    R3,R0                   ; R0 => CDDB.
           53     24 A5   D0 0B1C  3888          MOVL    CDRP$L_CDT(R5),R3       ; Set R3 => CDT.
      12 A0    0080 8F   A8 0B20  3889          BISW    #CDDB$M_NOCONN, -       ; Set no connection active flag.
                        0B26  3890                  CDDB$W_STATUS(R0)
                   04   E5 0B26  3891          BBCC    #CDDB$V_RESYNCH,-       ; Do NOT branch around if we were called
                1C 12 A0    0B28  3892                  CDDB$W_STATUS(R0),2$    ;  in order to re-synchronize.
             53   1C A3   D0 0B2B  3893          MOVL    CDT$L_PB(R3),R3         ; R3 => Path Block for MRESET, etc.
                        0B2F  3894          MRESET  PB$B_RSTATION(R3),#1    ; Force controller to reset itself.
                        0B39  3895          MSTART  PB$B_RSTATION(R3)       ; And force controller to restart itself.
                   05   0B46  3896          RSB                             ; Kill this thread. Rely on Port
                        0B47  3897                                          ;  Driver calling error routine as
                        0B47  3898                                          ;  a result of MRESET to accomplish
                        0B47  3899                                          ;  DISCONNECT and subsequent logic.
                        0B47  3900 2$:
                        0B47  3901          DISCONNECT      #DISCONNECT_REASON
                        0B50  3902
                        0B50  3903                                          ; Restore R3 => CDDB after disconnect.
                        0B50  3904          PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                        0B50  3905                  cdrp=R5, cddb=R3
                        0B57  3906
                        0B57  3907 ;
                        0B57  3908 ; Deallocate mapping resources
                        0B57  3909 ;
```

```
                        0B57  3910
                        0B57  3911          ; Any mapping resources still owned by CDRPs on the restart queue are
                        0B57  3912          ; deallocated here.  This deallocation is delayed until after the
                        0B57  3913          ; DISCONNECT (and possible MRESET) to prevent an "insane" controller
                        0B57  3914          ; from continuing to transfer via possibly re-allocated mapping
                        0B57  3915          ; resources.  Once the mapping resources are released, I/O requests
                        0B57  3916          ; related to "mounting" a disk are released so that $MOUNT or mount
                        0B57  3917          ; verification can take appropriate action.  This is done by
                        0B57  3918          ; releaseing any request with IRP$V_MVIRP or directed to a device with
                        0B57  3919          ; UCB$V_VAILD clear is released.  The mount verification case is
                        0B57  3920          ; obvious and experience shows that $MOUNT does not set UCB$V_VALID
                        0B57  3921          ; before issuing the IO$_PACKACK.
                        0B57  3922
          3C A3   9F    0B57  3923          PUSHAB  CDDB$L_RSTRTQFL(R3)     ; Setup listhead address.
          3C A3   DD    0B5A  3924          PUSHL   CDDB$L_RSTRTQFL(R3)     ; Setup first CDRP address.
                        0B5D  3925
            55 8ED0     0B5D  3926 4$:      POPL    R5                      ; Get next CDRP address.
       6E   55   D1     0B60  3927          CMPL    R5, (SP)                ; Is it the listhead?
            25   13     0B63  3928          BEQL    6$                      ; If yes, all deallocations are done.
          F498'  30     0B65  3929          BSBW    DUTU$DEALLOC_ALL        ; Free MAP resources owned by this CDRP.
            65   DD     0B68  3930          PUSHL   (R5)                    ; Push next CDRP address.
    50   BC A5   D0     0B6A  3931          MOVL    CDRP$L_UCB(R5), R0      ; Get UCB address.
05 64 A0    0B   E1     0B6E  3932          BBC     #UCB$V_VALID, -         ; Branch to post the IRP if this
                        0B73  3933                  UCB$L_STS(R0), 5$       ; device is not "volume valid."
E5 CA A5    0D   E1     0B73  3934          BBC     #IRP$V_MVIRP, -         ; Is this a mount verification IRP?
                        0B78  3935                  CDRP$W_STS(R5), 4$      ; Branch if not an MV IRP.
       50   65   0F     0B78  3936 5$:      REMQUE  (R5), R0                ; Else, remove IRP/CDRP from restart
                        0B7B  3937          POST_CDRP status=SS$_MEDOFL     ; queue and send it to post processing.
            D3   11     0B88  3938          BRB     4$                      ; Loop till all restart CDRPs are done.
                        0B8A  3939
            8E   D5     0B8A  3940 6$:      TSTL    (SP)+                   ; Clear listhead pointer from stack.
                        0B8C  3941
                        0B8C  3942          ; Deallocate mapping resources whose description is stored in the
                        0B8C  3943          ; CDDB permanent CDRP.  This information was placed there by
                        0B8C  3944          ; DUTU$INSERT_RESTARTQ when it discovered that the HIRT permanent CDRP
                        0B8C  3945          ; owned mapping resources.  In this way, another thread is allowed to
                        0B8C  3946          ; use the HIRT permanent CDRP while this connection is broken.
                        0B8C  3947
55  00D0 C3   9E        0B8C  3948          MOVAB   CDDB$A_PRMCDRP(R3), R5  ; Get CDRP in R5.
       F46C'  30        0B91  3949          BSBW    DUTU$DEALLOC_ALL        ; Free old HIRT MAP resources.
                        0B94  3950                                          ;  the HIRT CDRP and whose ownership
                        0B94  3951                                          ;  has been transferred here.
                        0B94  3952
         F760   30      0B94  3953          BSBW    DU_REVALIDATE_DISKS     ; Start mount verification for online
                        0B97  3954                                          ; devices.  Starting MV here allows
                        0B97  3955                                          ; MV to perform failover, if that's
                        0B97  3956                                          ; possible.
                        0B97  3957
                        0B97  3958 ;
                        0B97  3959 ; re-CONNECT - Here we call an internal subroutine which:
                        0B97  3960 ;
                        0B97  3961 ;       1. Makes a connection to the MSCP server in the intelligent
                        0B97  3962 ;             controller.
                        0B97  3963 ;
                        0B97  3964 ;       2. Sends an MSCP command to SET CONTROLLER CHARACTERISTICS.
                        0B97  3965 ;
                        0B97  3966 ;       3. Allocates an MSCP buffer and RSPID for our future use in
```

```
                     0B97  3967 ;                  connection management.
                     0B97  3968 ;
                     0B97  3969 ; Upon return R4 => PDT and R5 => CDRP.
                     0B97  3970 ;
                     0B97  3971
 55   00D0 C3   9E   0B97  3972          MOVAB   CDDB$A_PRMCDRP(R3), R5  ; Get permanent CDRP address.
        F604   30   0B9C  3973          BSBW    MAKE_CONNECTION         ; Call subroutine to connect.
                     0B9F  3974
                     0B9F  3975          PERMCDRP_TO_CDDB -              ; Get CDDB address in R3.
                     0B9F  3976                  cdrp=R5, cddb=R3
03 28 A3    0F   E0  0BA6  3977          BBS     #MSCP$V_CF_REPLC, -     ; Branch if controller will perform
                     0BAB  3978                  CDDB$W_CNTRLFLGS(R3),10$; bad block replacement operations.
        F452'   30   0BAB  3979          BSBW    DU$INIT_HIRT            ; Else, initialize HIRT.
                     0BAE  3980
                     0BAE  3981 10$:     ; Now it is necessary to propogate all the connection dependent
                     0BAE  3982          ; information regarding the newly formed connection to the MSCP server
                     0BAE  3983          ; to all the UCB's in the primary chain for this CDDB.  At the same
                     0BAE  3984          ; time, every RWAITCNT value is tested to insure that it is consistant
                     0BAE  3985          ; with what would be expected based upon the various possible reasons
                     0BAE  3986          ; which cause it to be bumped.  This is merely a debugging exercise.
                     0BAE  3987          ; In END_SINGLE_STREAM, RWAITCNT will be reduced by one and the wait
                     0BAE  3988          ; count bumped flag will be cleared.
                     0BAE  3989
   55   84 A3   9E   0BAE  3990          MOVAB   <CDDB$L_UCBCHAIN -      ; Setup "previous" UCB address.
                     0BB2  3991                  -UCB$L_CDDB_LINK>(R3), -
                     0BB2  3992                  R5
 55   00C4 C5   D0   0BB2  3993 15$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Link to next UCB.
          08   13   0BB7  3994          BEQL    30$                     ; Branch if no more UCBs to test.
        F444'   30   0BB9  3995          BSBW    DUTU$INIT_CONN_UCB      ; Setup connection dep. UCB fields.
        F441'   30   0BBC  3996          BSBW    DUTU$CHECK_RWAITCNT     ; Validate the wait count value.
          F1    11   0BBF  3997          BRB     15$                     ; Loop through all UCBs.
                     0BC1  3998
                     0BC1  3999 30$:     ; It is now possible to execute requests on behalf of any pending
                     0BC1  4000          ; mount verification threads.  Therefore, the CDDB$V_NOCONN bit is
                     0BC1  4001          ; cleared.
                     0BC1  4002
    12 A3    0080 8F   AA   0BC1  4003          BICW    #CDDB$M_NOCONN, CDDB$W_STATUS(R3)
       50    18 A3   D0   0BC7  4004          MOVL    CDDB$L_CRB(R3),R0       ; Get CRB address.
    1C A0  0C7A'CF   9E   0BCB  4005          MOVAB   W^DU$TMR, -             ; Establish permanent timeout routine.
                     0BD1  4006                  CRB$L_TOUTROUT(R0)
       51    2A A3   3C   0BD1  4007          MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Get controller timeout interval.
18 A0  00000000'GF   51   C1   0BD5  4008          ADDL3   R1, G^EXE$GL_ABSTIM, -  ; Use that to set next timeout
                     0BDE  4009                  CRB$L_DUETIME(R0)       ; wakeup time.
                     0BDE  4010
                     0BDE  4011          ; The normal MSCP timeout mechanism is now in effect.  Henceforth,
                     0BDE  4012          ; no fork thread may use the CDDB permanent CDRP as a fork block.
                     0BDE  4013          ;
                     0BDE  4014          ; While the mount verification threads are being handled, also poll
                     0BDE  4015          ; for any previously unknown devices handled by this server.
                     0BDE  4016
                     0BDE  4017          ASSUME  CDDB$V_DAPBSY GE 8
    13 A3    04   88   0BDE  4018          BISB    #<CDDB$M_DAPBSY @ -8>, -; Set DAP CDRP in use flag.
                     0BE2  4019                  CDDB$W_STATUS+1(R3)
 55    54 A3   D0   0BE2  4020          MOVL    CDDB$L_DAPCDRP(R3), R5  ; Get DAP CDRP address.
        F417'   30   0BE6  4021          BSBW    DUTU$POLL_FOR_UNITS     ; Poll for units.
    13 A3    04   8A   0BE9  4022          BICB    #<CDDB$M_DAPBSY @ -8>, -; Clear DAP CDRP in use flag.
                     0BED  4023                  CDDB$W_STATUS+1(R3)
```

```
                                0BED  4024
                                0BED  4025          ; At this point, there once was a test for an empty CDDB$L_CDRPQFL.
                                0BED  4026          ; Such a test is no longer valid because mount verification processing
                                0BED  4027          ; may very well have CDRPs active on the connection represented by
                                0BED  4028          ; this CDDB.
                                0BED  4029
                                0BED  4030 ;
                                0BED  4031 ; Restore normal CRB timeout routine and interval.
                                0BED  4032 ;
            50    18 A3   D0    0BED  4033          MOVL    CDDB$L_CRB(R3), R0      ; Get CRB address.
      1C A0     0C7A'CF   9E    0BF1  4034          MOVAB   W^DU$TMR, -             ; Re-establish normal timeout
                                0BF7  4035                  CRB$L_TOUTROUT(R0)      ; routine.
            51    2A A3   3C    0BF7  4036          MOVZWL  CDDB$W_CNTRLTMO(R3), R1 ; Establish controller specified
   18 A0    00000000'GF   51    C1    0BFB  4037          ADDL3   R1, G^EXE$GL_ABSTIM, -  ; delta time as normal timeout
                                0C04  4038                  CRB$L_DUETIME(R0)       ; interval for timeout mechanism.
                                0C04  4039
                                0C04  4040 ; At this point, the reconnection thread may need to be suspended (you may
                                0C04  4041 ; wish to think of it as ended) until mount verification completes for all the
                                0C04  4042 ; disks (or UCBs) which had mount verification started by DU_REVALIDATE_DISKS.
                                0C04  4043 ; This is necessary if and only if CDDB$W_WTUCBCTR is non-zero, indicating
                                0C04  4044 ; that one or more UCBs are waiting for mount verification to complete.  When
                                0C04  4045 ; this thread is "suspended," the end mount verification routine will call
                                0C04  4046 ; RESTART_NEXT_CDRP when the count of UCBs waiting for mount verification to
                                0C04  4047 ; complete reaches zero.  This scheme is described in excruciating detail in
                                0C04  4048 ; the DU_REVALIDATE_DISKS preamble.
                                0C04  4049
                  5E A3   B5    0C04  4050          TSTW    CDDB$W_WTUCBCTR(R3)     ; Are any UCBs waiting for mnt. ver.?
                     07   13    0C07  4051          BEQL    RESTART_NEXT_CDRP       ; Branch if no UCBs are waiting.
      12 A3    0100 8F   A8    0C09  4052          BISW    #CDDB$M_RSTRTWAIT, -    ; Else, signal that connection is
                                0C0F  4053                  CDDB$W_STATUS(R3)       ; waiting to restart CDRPs and
                     05   0C0F  4054          RSB                                 ; "suspend" the reconnect thread.
```

```
                        0C10  4056 RESTART_NEXT_CDRP:
                        0C10  4057
                        0C10  4058 ;
                        0C10  4059 ; Here we attempt to initiate the first (i.e. next) CDRP on the restart queue.
                        0C10  4060 ;       In order to prevent getting caught in an infinite loop trying to
                        0C10  4061 ;       initiate an operation that the controller cannot complete for
                        0C10  4062 ;       one reason or another, we maintain a retry count and the address
                        0C10  4063 ;       of the CDRP that we are currently single streaming.
                        0C10  4064 ;
                        0C10  4065 ;       In the normal case this is an isolated re-CONNECTION and the
                        0C10  4066 ;       first CDRP on the restart queue is a random CDRP.  We notice this
                        0C10  4067 ;       by seeing that the address of our first CDRP is not equal to the
                        0C10  4068 ;       current contents of CDDB$L_RSTRTCDRP.
                        0C10  4069 ;
                        0C10  4070 ;       In the other case the connection failed while we were in single
                        0C10  4071 ;       stream mode and the CDRP which we happened to be processing is the
                        0C10  4072 ;       same CDRP that now heads our restart queue.  In this case, before
                        0C10  4073 ;       initiating the processing of this CDRP, we decrement 1 from the
                        0C10  4074 ;       retry count and if it remains non-zero, we restart the CDRP
                        0C10  4075 ;       processing.  If the decrementing results in a zero retry count,
                        0C10  4076 ;       then we log the event and effectively abort the CDRP by branching to
                        0C10  4077 ;       FUNCTION_EXIT with an appropriate error status.  FUNCTION_EXIT, due
                        0C10  4078 ;       to the setting of the CDDB$M_SNGLSTRM bit will then start the
                        0C10  4079 ;       processing of the next CDRP on the restart queue.
                        0C10  4080 ;
                        0C10  4081 ; We can arrive here either by falling through from the above code or via
                        0C10  4082 ;       a branch from FUNCTION_EXIT.  In either case we have:
                        0C10  4083 ;
                        0C10  4084 ; INPUT:
                        0C10  4085 ;       R3 => CDDB
                        0C10  4086 ;
                        0C10  4087
     55   3C B3   0F    0C10  4088          REMQUE  @CDDB$L_RSTRTQFL(R3),R5 ; R5 => 1st CDRP on restart queue.
                2F    1D  0C14  4089          BVS     END_SINGLE_STREAM       ; VS implies restart was empty.
                00    E3  0C16  4090          BBCS    #CDDB$V_SNGLSTRM,-      ; Set bit and if clear, this is 1st
              1B 12   A3        0C18  4091                  CDDB$W_STATUS(R3),20$   ;  time here for this CDRP, so branch.
          34 A3       55    D1  0C1B  4092          CMPL    R5,CDDB$L_RSTRTCDRP(R3) ; See if same CDRP as last time.
                15    12  0C1F  4093          BNEQ    20$                     ; NEQ implies not the same.
             38 A3    97  0C21  4094          DECB    CDDB$B_RETRYCNT(R3)     ; If same, decrement 1 from retries.
                18    12  0C24  4095          BNEQ    30$                     ; NEQ implies retries remaining.
                          0C26  4096
                          0C26  4097 ;
                          0C26  4098 ; ********************************Log this error.*******************************
                          0C26  4099 ;
                          0C26  4100
     50   00000054 8F    D0  0C26  4101          MOVL    #SS$_CTRLERR,R0         ; Indicate appropriate error status.
                51    D4  0C2D  4102          CLRL    R1                      ;  And set second part of I/O status.
           53   BC A5    D0  0C2F  4103          MOVL    CDRP$L_UCB(R5),R3       ; R3 => UCB.
                FD21  31  0C33  4104          BRW     FUNCTION_EXIT
                          0C36  4105 20$:
          34 A3    55    D0  0C36  4106          MOVL    R5,CDDB$L_RSTRTCDRP(R3) ; Establish new single stream CDRP.
                02    90  0C3A  4107          MOVB    #MAX_RETRY,-            ; Establish fresh retry count.
             38 A3        0C3C  4108                  CDDB$B_RETRYCNT(R3)
                          0C3E  4109 30$:
           53   BC A5    D0  0C3E  4110          MOVL    CDRP$L_UCB(R5),R3       ; R3 => UCB.
                F8BF  31  0C42  4111          BRW     DU_RESTARTIO            ; Restart the CDRP.
                          0C45  4112
```

```
                        0C45  4113
                        0C45  4114 END_SINGLE_STREAM:
                        0C45  4115
                        0C45  4116 ;
                        0C45  4117 ; Here we want to resume normal operation and get each unit going.
                        0C45  4118 ;       To do this we pickup each UCB in turn and call SCS$UNSTALLUCB
                        0C45  4119 ;       for it.  This has the effect of starting up as many (perhaps all)
                        0C45  4120 ;       of the IRP's (that's right IRP's) as possible that may have
                        0C45  4121 ;       backed up on the UCB input queue while we were in single stream mode.
                        0C45  4122 ;       We then go on to the next UCB until we exhaust all UCB's connected
                        0C45  4123 ;       to this CDDB.
                        0C45  4124 ;
                        0C45  4125
    12 A3   01    AA    0C45  4126          BICW    #CDDB$M_SNGLSTRM, -     ; Clear single streaming CDRPs flag.
                        0C49  4127                  CDDB$W_STATUS(R3)
    50   3A A3   3C     0C49  4128          MOVZWL  CDDB$W_RSTRTCNT(R3), R0 ; Get current restart count.
    55   84 A3   9E     0C4D  4129          MOVAB   <CDDB$L_UCBCHAIN -      ; Setup "previous" UCB address.
                        0C51  4130                  -UCB$L_CDDB_LINK>(R3), -
                        0C51  4131                  R5
                        0C51  4132
    55   00C4 C5   D0   0C51  4133 10$:     MOVL    UCB$L_CDDB_LINK(R5), R5 ; Point to next UCB.
                  1D   13   0C56  4134          BEQL    30$                     ; Branch if no more UCBs to process.
68 A5   0400 8F     AA    0C58  4135          BICW    #UCB$M_MSCP_WAITBMP, -  ; Indicate RWAITCNT no longer bumped.
                        0C5E  4136                  UCB$W_DEVSTS(R5)
          56 A5   B7     0C5E  4137          DECW    UCB$W_RWAITCNT(R5)      ; Unbump wait count.
           F39C'  30     0C61  4138          BSBW    DUTU$CHECK_RWAITCNT     ; Else, check wait count and
                  09   BB   0C64  4139          PUSHR   #^M<R0,R3>              ; Save restart cnt. and CDDB address.
00000000'GF       16     0C66  4140          JSB     G^SCS$UNSTALLUCB        ; Start up IRPs on UCB.
                  09   BA   0C6C  4141          POPR    #^M<R0,R3>              ; Restore restart cnt. and CDDB address.
    3A A3   50     B1     0C6E  4142          CMPW    R0, CDDB$W_RSTRTCNT(R3) ; Did the unstall cause a restart?
                  DD   13   0C72  4143          BEQL    10$                     ; Branch if no restart was caused.
                  05   0C74  4144          RSB                             ; Else, discontinue this thread.
                        0C75  4145
    12 A3   08    AA    0C75  4146 30$:     BICW    #CDDB$M_RECONNECT, -    ; Clear reconnect in progress bit.
                        0C79  4147                  CDDB$W_STATUS(R3)
                  05   0C79  4148          RSB                             ; Ta De, Ta De, that's all folks.
```

```
                        .SBTTL  DU$TMR - Class Driver Timeout Mechanism Routine

;+
; DU$TMR - Time out Mechanism Routine.  This routine is called
;  periodically whenever CRB$L_DUETIME becomes due.  At the time of a
;  periodic call to DU$TMR the Class Driver is in one of three states
;  with respect to the intelligent mass storage controller associated
;  with the CRB pointed at by R3.
;
;       1. State #1, the "normal" state for which this routine is optimized,
;          is characterized by the following two conditions:
;
;              a) One or more MSCP commands are outstanding to the controller.
;                 This is determined by having a NON-empty queue of CDRP's
;                 hanging off the CDDB.
;
;              b) The oldest outstanding command was initiated since the
;                 previous invocation of DU$TMR and is therefore not very
;                 old.  This is determined by comparing the RSPID of the
;                 currently oldest command to the RSPID of the oldest request
;                 at the time of the previous invocation.  If they are not
;                 equal then we are in State #1.
;
;       2. State #2 is characterized by having NO outstanding MSCP commands in
;          the controller.  This is determined by finding an empty CDRP queue
;          in the CDDB.
;
;       3. State #3 is the state where MSCP commands are outstanding and the
;          oldest one has been outstanding for at least one previous DU$TMR
;          invocation.
;
; If we determine that we are in state #1, we simply record the RSPID of the
;  currently oldest outstanding MSCP command in CDDB$L_OLDRSPID and we initial-
;  ize CDDB$L_OLDCMDSTS to all 1's.  We then calculate a new due time,
;  place it in CRB$L_DUETIME and return to our caller, which results
;  in scheduling ourselves for the next invocation of DU$TMR.
;
; States #2 and #3 share some common code.  In both cases we will issue an
;  IMMEDIATE command to the controller but for diverse reasons.  In the case
;  of state #2 it will be an effective NOP command that is only issued to
;  insure against the controller timing out the host (i.e. us) due to lack of
;  activity on our part.  In the case of state #3, the IMMEDIATE command will
;  be a "GET COMMAND STATUS" for the oldest outstanding MSCP command.
;
; The common code they share consists of code to appropriate the pre-allocated
;  MSCP buffer pointed at by CDRP$L_MSG_BUF and to pick up the pre-allocated
;  RSPID identified by CDRP$L_RSPID.  Both these items are located in
;  the permanent CDRP which is appended to the CDDB of this intelligent
;  controller.  Also at this time a new due time is calculated prior to
;  doing the DRIVER_SEND_MSG so that we will be able to time out the
;  Immediate command.  Then the code for these two states diverges for
;  a while to prepare distinct MSCP packets, do the SEND_MSG_BUF,
;  and in the case of state #3, to do some specific processing upon
;  receipt of the END PACKET for the IMMEDIATE command.  This processing
;  consists of insuring that the command status returned in the END PACKET
;  indicates progress being made on the oldest outstanding command; and also
;  of saving this received command status in the CDDB$L_OLDCMDSTS so as to
```

```
                     0C7A  4207 ;  have it available at the next invocation, if this oldest command is still
                     0C7A  4208 ;  outstanding.  Following this the two code paths converge to recycle the
                     0C7A  4209 ;  received END PACKET for use as the next IMMEDIATE MSCP buffer and to also
                     0C7A  4210 ;  recycle the RSPID by bumping its sequence number.
                     0C7A  4211 ;
                     0C7A  4212 ; INPUTS:
                     0C7A  4213 ;       R3 => CRB of the intelligent disk controller
                     0C7A  4214 ;
                     0C7A  4215 ; OUTPUTS:
                     0C7A  4216 ;       Registers R0 through R5 are all possibly modified.
                     0C7A  4217 ;
                     0C7A  4218
                     0C7A  4219 DU$TMR:
                     0C7A  4220          SETIPL  #IPL$_SCS                   ; After wakeup lower IPL.
51    10 A3    D0    0C7D  4221          MOVL    CRB$L_AUXSTRUC(R3),R1       ; R1 => CDDB.
                     0C81  4222
                     0C81  4223          ASSUME  CDDB$L_CDRPQFL  EQ  0
   51    61    D1    0C81  4224          CMPL    (R1),R1                     ; If =, then list of CDRP's is empty
         21    13    0C84  4225          BEQL    20$                         ; EQL means empty list of CDRP's,
                     0C86  4226                                              ;  which implies we are in State #2.
   50    61    D0    0C86  4227          MOVL    (R1),R0                     ; R0 => CDRP associated with "oldest"
                     0C89  4228                                              ;  outstanding MSCP command.
                     0C89  4229
      20 A0    D1    0C89  4230          CMPL    CDRP$L_RSPID(R0),-          ; Compare RSPID of oldest request to
      2C A1          0C8C  4231                  CDDB$L_OLDRSPID(R1)         ;  that of request current at time of
                     0C8E  4232                                              ;  previous invocation of DU$TMR.
         1C    13    0C8E  4233          BEQL    30$                         ; EQL implies State #3, i.e. current
                     0C90  4234                                              ;  oldest has been around for awhile.
                     0C90  4235
      20 A0    D0    0C90  4236          MOVL    CDRP$L_RSPID(R0),-          ; State #1, we have a NEW oldest request
      2C A1          0C93  4237                  CDDB$L_OLDRSPID(R1)         ;  so record its RSPID in CDDB field.
30 A1    01    CE    0C95  4238          MNEGL   #1,CDDB$L_OLDCMDSTS(R1) ; And initialize its associated status.
                     0C99  4239 10$:
7E    2A A1    3C    0C99  4240          MOVZWL  CDDB$W_CNTRLTMO(R1),-(SP); Pickup controller delta.
         8E    C1    0C9D  4241          ADDL3   (SP)+,-                     ; Calculate delta time for next
00000000'GF          0C9F  4242                  G^EXE$GL_ABSTIM,-           ;  periodic invocation of DU$TMR.
      18 A3          0CA4  4243                  CRB$L_DUETIME(R3)
               05    0CA6  4244          RSB                                 ; And return to caller.
                     0CA7  4245
                     0CA7  4246 20$:                                         ; If we are here, there are NO outstand-
                     0CA7  4247                                              ;  ing requests in the controller since
                     0CA7  4248                                              ;  CDRP list is empty.
      50    D4       0CA7  4249          CLRL    R0                          ; R0 flagged to indicate State #2.
      2C A1    D4    0CA9  4250          CLRL    CDDB$L_OLDRSPID(R1)         ; Set to impossible value to prevent
                     0CAC  4251                                              ;  inadvertent comparison error.
                     0CAC  4252
                     0CAC  4253 30$:                                         ; Common State #2, State #3 code path.
                     0CAC  4254                                              ; If here, for sure we will be issuing
                     0CAC  4255                                              ;  an immediate command to the controller.
                     0CAC  4256                                              ;  If we are in State #2, it will be a
                     0CAC  4257                                              ;  "GET UNIT STATUS" (NOP) command but
                     0CAC  4258                                              ;  if we are in State #3, it will be
                     0CAC  4259                                              ;  a "GET COMMAND STATUS" command.  For
                     0CAC  4260                                              ;  either case we begin the common setup.
                     0CAC  4261
                     0CAC  4262
54    14 A1    D0    0CAC  4263          MOVL    CDDB$L_PDT(R1),R4           ; Setup for SEND_MSG_BUF, R4=>PDT.
```

```
                55   00D0 C1   9E  0CB0  4264          MOVAB   CDDB$A_PRMCDRP(R1), R5  ; Get CDRP appended to CDDB.
                          01   E3  0CB5  4265          BBCS    #CDDB$V_IMPEND,-        ; Branch if an immediate command is NOT
                     03 12 A1        0CB7  4266                  CDDB$W_STATUS(R1),40$   ;  pending. Also set bit to show that
                                     0CBA  4267                                          ;  one WILL be pending momentarily.
                         FDD1   31   0CBA  4268          BRW     DU$RE_SYNCH             ; Bit set implies that an immediate
                                     0CBD  4269                                          ;  "GET STATUS" type command has not
                                     0CBD  4270                                          ;  completed in the timeout interval.
                                     0CBD  4271                                          ;  So we goto resynchronization logic.
                                     0CBD  4272
                                     0CBD  4273  40$:
                     7E   50   7D    0CBD  4274          MOVQ    R0, -(SP)               ; Save valuable registers.
                                     0CC0  4275          INIT_MSCP_MSG                   ; Initalize buffer for MSCP message.
                     50   8E   7D    0CC3  4276          MOVQ    (SP)+, R0               ; Restore valuable registers.
                                     0CC6  4277
                          D1   10    0CC6  4278          BSBB    10$                     ; Establish due time so as to be able
                                     0CC8  4279                                          ;  to timeout Immediate command.
                          50   D5    0CC8  4280          TSTL    R0                      ; Test for State #2 or State #3.
                          09   12    0CCA  4281          BNEQ    50$                     ; NEQ implies State #3. Branch to handle it.
                                     0CCC  4282
                                     0CCC  4283  ; State #2 specific code.
                                     0CCC  4284  ; Here we prepare the MSCP packet for the "GET UNIT STATUS" command for
                                     0CCC  4285  ;       unit #0, which is an effective NOP command.  This is done to
                                     0CCC  4286  ;       maintain minimum activity so that the controller will not time
                                     0CCC  4287  ;       out the host (i.e. us). NOTE that since the MSCP buffer has been
                                     0CCC  4288  ;       cleared above, there is no need to specify unit #0 in the command
                                     0CCC  4289  ;       buffer.
                                     0CCC  4290  ;
                                     0CCC  4291
                          03   90    0CCC  4292          MOVB    #MSCP$K_OP_GTUNT,-      ; Move in "GET UNIT STATUS" opcode.
                     08 A2            0CCE  4293                  MSCP$B_OPCODE(R2)
                                     0CD0  4294
                                     0CD0  4295          SEND_MSCP_MSG DRIVER            ; Here we call to send the MSCP packet
                                     0CD3  4296                                          ;  to the intelligent disk controller.
                                     0CD3  4297
                                     0CD3  4298                                          ; Return is experienced here after
                                     0CD3  4299                                          ;  receipt of the END PACKET correspond-
                                     0CD3  4300                                          ;  ing to the MSCP NOP sent above.  We
                                     0CD3  4301                                          ;  regain control due to a callback
                                     0CD3  4302                                          ;  from our own INPUT DISPATCHER
                                     0CD3  4303                                          ;  ROUTINE.  Passed to us at this call-
                                     0CD3  4304                                          ;  back are R2 => END PACKET, R3 => CRB,
                                     0CD3  4305                                          ;  R4 => PDT and R5 => CDRP.
                                     0CD3  4306                                          ; All we want to do is recycle the
                                     0CD3  4307                                          ;  END PACKET for use as our next MSCP
                                     0CD3  4308                                          ;  packet and recycle the RSPID.
                                     0CD3  4309                                          ;  To do this we branch to common code.
                          35   11    0CD3  4310          BRB     70$
                                     0CD5  4311
                                     0CD5  4312  50$:
                                     0CD5  4313
                                     0CD5  4314  ; State #3 specific code.
                                     0CD5  4315  ; Here we prepare the MSCP packet for a "GET COMMAND STATUS" command.
                                     0CD5  4316
                50   BC A0   D0    0CD5  4317          MOVL    CDRP$L_UCB(R0),R0       ; R0 => UCB for oldest outstanding request.
                                     0CD9  4318
                   00D4 C0   B0    0CD9  4319          MOVW    UCB$W_MSCPUNIT(R0),-    ; Setup UNIT field.
                     04 A2           0CDD  4320                  MSCP$W_UNIT(R2)
```

```
                     02   90   0CDF  4321          MOVB    #MSCP$K_OP_GTCMD,-      ; Setup OPCODE field.
                 08  A2        0CE1  4322                  MSCP$B_OPCODE(R2)
                               0CE3  4323
                 2C  A1   D0   0CE3  4324          MOVL    CDDB$L_OLDRSPID(R1),-   ; Setup OUTSTANDING COMMAND REFERENCE
                 0C  A2        0CE6  4325                  MSCP$L_OUT_REF(R2)      ;  field.
                               0CE8  4326
                               0CE8  4327          SEND_MSCP_MSG DRIVER            ; Here we call to send the MSCP packet
                               0CEB  4328                                          ;  to the intelligent disk controller.
                               0CEB  4329
                               0CEB  4330                                          ; We experience return here upon receipt
                               0CEB  4331                                          ;  of the END PACKET for the above "GET
                               0CEB  4332                                          ;  COMMAND STATUS" command. We must make
                               0CEB  4333                                          ;  sure that progress has indeed been
                               0CEB  4334                                          ;  made on the outstanding command.  We
                               0CEB  4335                                          ;  therefore compare the outstanding
                               0CEB  4336                                          ;  command status returned in the END
                               0CEB  4337                                          ;  PACKET to the previous value in CDDB
                               0CEB  4338                                          ;  field CDDB$L_OLDCMDSTS.
                               0CEB  4339                                          ; Here R2=>END PACKET, R3=>CRB, R4=>PDT
                               0CEB  4340                                          ;  and R5=>CDRP.
                               0CEB  4341
             51    10  A3   D0   0CEB  4342          MOVL    CRB$L_AUXSTRUC(R3),R1   ; R1 => CDDB.
                   10  A2   D1   0CEF  4343          CMPL    MSCP$L_CMD_STS(R2),-    ; Compare received outstanding command
                   30  A1        0CF2  4344                  CDDB$L_OLDCMDSTS(R1)    ;  status to previous value.
                       0F   1F   0CF4  4345          BLSSU   60$                     ; LSSU implies progress made so branch.
                       0A   12   0CF6  4346          BNEQ    55$                     ; If not equal, progress went the
                                 0CF8  4347                                          ;  wrong direction; a sure sign of
                                 0CF8  4348                                          ;  an insane controller.
       10  A2   FFFFFFFF  8F   D1   0CF8  4349          CMPL    #-1, MSCP$L_CMD_STS(R2) ; If equal to last time, is this the
                                 0D00  4350                                          ;  multi-host busy somewhere else value?
                         03   13   0D00  4351          BEQL    60$                     ; Branch if it is busy somewhere else.
                       FD89   31   0D02  4352  55$:    BRW     DU$RE_SYNCH             ; Anything else, implies no progress
                                 0D05  4353                                          ;  has been made.  So we goto
                                 0D05  4354                                          ;  re-synchronize with the intelligent
                                 0D05  4355                                          ;  disk controller and re-issue all
                                 0D05  4356                                          ;  outstanding commands.
                                 0D05  4357
                                 0D05  4358  60$:
                   10  A2   D0   0D05  4359          MOVL    MSCP$L_CMD_STS(R2),-    ; Remember this received outstanding
                   30  A1        0D08  4360                  CDDB$L_OLDCMDSTS(R1)    ;  command status for next time.
                                 0D0A  4361
                                 0D0A  4362  70$:                                    ; States #2 and #3 code paths merge here.
                                 0D0A  4363          RECYCH_MSG_BUF          ; Recycle END PACKET.
                                 0D0D  4364          RECYCL_RSPID                    ; Likewise the RSPID.
                                 0D13  4365
             51    10  A3   D0   0D13  4366          MOVL    CRB$L_AUXSTRUC(R3),R1   ; R1 => CDDB.
                       02   AA   0D17  4367          BICW    #CDDB$M_IMPEND,-        ; Indicate that immediate command is
                   12  A1        0D19  4368                  CDDB$W_STATUS(R1)       ;  no longer pending.
                    F2E2'  31   0D1B  4369          BRW     DUTU$DODAP              ; Continue by doing DAP processing.
```

```
                     OD1E  4371          .SBTTL  DU$IDR - Class Driver Input Dispatch Routine
                     OD1E  4372
                     OD1E  4373 ;+
                     OD1E  4374 ; DU$IDR - Class Driver Input Dispatching Routine.  This routine is to
                     OD1E  4375 ;          the class driver what the interrupt Service Routine is to a
                     OD1E  4376 ;          conventional driver.  We are called here by the Port Driver
                     OD1E  4377 ;          and we are passed the address of an END PACKET or an ATTENTION
                     OD1E  4378 ;          MESSAGE buffer.  By testing a bit in the ENDCODE field of the
                     OD1E  4379 ;          received buffer we determine which of the two has been received.
                     OD1E  4380 ;          For ATTENTION MESSAGES we immediately branch to ATTN_MSG.
                     OD1E  4381 ;
                     OD1E  4382 ;          For END PACKETs we first determine if the END PACKET is still of
                     OD1E  4383 ;          interest.  This is done by testing whether the COMMAND REFERENCE
                     OD1E  4384 ;          NUMBER returned in the END PACKET, interpreted as a RSPID, is
                     OD1E  4385 ;          still valid.  If not, we merely deallocate the END PACKET and
                     OD1E  4386 ;          return to our caller in the Port Driver.
                     OD1E  4387 ;
                     OD1E  4388 ;          If the END PACKET is still of interest then before dispatching
                     OD1E  4389 ;          to the code that originally issued the MSCP command for which we
                     OD1E  4390 ;          have just received the END PACKET, we first remove the
                     OD1E  4391 ;          CDRP associated with the command from the list of active CDRP's
                     OD1E  4392 ;          defined by the listhead located at CDDB$L_CDRPQFL.
                     OD1E  4393 ;
                     OD1E  4394 ; INPUTS:
                     OD1E  4395 ;          R1 =  Message Length
                     OD1E  4396 ;          R2 => END PACKET or ATTENTION MESSAGE BUFFER
                     OD1E  4397 ;          R3 => Connection Data Block
                     OD1E  4398 ;-
                     OD1E  4399
                     OD1E  4400 DU$IDR::
    08 A2    80 8F    93  OD1E  4401          BITB    #MSCP$M_OP_END,-        ; Is this an ATTENTION MESSAGE?
                     OD23  4402                       MSCP$B_OPCODE(R2)
                 57   13  OD23  4403          BEQL    ATTN_MSG                ; Branch if its an ATTENTION MESSAGE.
                     OD25  4404
                     OD25  4405
                     OD25  4406 ;
                     OD25  4407 ; Process command END MESSAGES
                     OD25  4408 ;
                     OD25  4409
                     OD25  4410          ; NOTE: The following is a duplication of the code in FIND_RSPID_RDTE.
                     OD25  4411          ; The code is reproduced inline here to avoid two branches and make
                     OD25  4412          ; this code path as fast as possible.
                     OD25  4413
50   00000000'GF     D0  OD25  4414          MOVL    G^SCS$GL_RDT,R0         ; R0 => base of the RDT.
        55   62      3C  OD2C  4415          MOVZWL  MSCP$L_CMD_REF(R2),R5   ; Pickup RDTE index from END PACKET.
     F8 A0   55      D1  OD2F  4416          CMPL    R5,RDT$L_MAXRDIDX(R0)   ; Bounds check on index.
             79      18  OD33  4417          BGEQ    FINISHED_WITH_MESSAGE   ; GEQ implies out of bounds.
      55   6045      7E  OD35  4418          MOVAQ   (R0)[R5],R5             ; R5 => RDTE of interest.
                     OD39  4419
          06 A5      B1  OD39  4420          CMPW    RD$W_SEQNUM(R5),-       ; See if RSPID is still valid by com-
          02 A2          OD3C  4421                  MSCP$L_CMD_REF+2(R2)    ;  paring the sequence number.
             6E      12  OD3E  4422          BNEQ    FINISHED_WITH_MESSAGE   ; NEQ implies invalid RSPID.
        55   65      D0  OD40  4423          MOVL    RD$L_CDRP(R5),R5        ; R5 => CDRP.
     50   24 A5      D0  OD43  4424          MOVL    CDRP$L_CDT(R5),R0       ; R0 => CDT.
     50   5C A0      D0  OD47  4425          MOVL    CDT$L_AUXSTRUC(R0),R0   ; R0 => CDDB.
          2C A0      D1  OD4B  4426          CMPL    CDDB$L_OLDRSPID(R0),-   ; See if oldest outstanding command has
             62          OD4E  4427                  MSCP$L_CMD_REF(R2)      ;  this Command Reference Number.
```

```
                03   12  0D4F   4428          BNEQ    20$                             ; If not, branch around.
              2C A0  D4  0D51   4429          CLRL    CDDB$L_OLDRSPID(R0)             ; Prevent inadvertent timeouts due to
                         0D54   4430                                                  ;  reuse of RSPID in error situations.
                         0D54   4431  20$:    ASSUME  MSCP$K_LEN LT 32767
          46 A5   51   B0  0D54   4432          MOVW    R1, CDRP$W_ENDMSGSIZ(R5); Save length of incomming packet.
          1C A5   52   D0  0D58   4433          MOVL    R2, CDRP$L_MSG_BUF(R5)  ; Save address of incomming packet.
                         0D5C   4434
             55   65   0F  0D5C   4435          REMQUE  (R5),R5                 ; Remove R5=>CDRP from list.
                         0D5F   4436
                         0D5F   4437          ASSUME  CDRP$V_CAND EQ 0
             0F 40 A5    E8  0D5F   4438          BLBS    CDRP$L_DUTUFLAGS(R5), - ; Has request been canceled?
                         0D63   4439                  30$                     ; If so, do cancel completion work.
                         0D63   4440
       CA A5   0080 8F   B3  0D63   4441  23$:    BITW    #IRP$M_DIAGBUF, -       ; Was a diagnostic buffer supplied?
                         0D69   4442                  CDRP$W_STS(R5)
                   0C    12  0D69   4443          BNEQ    50$                     ; Branch if diagnostic buffer present.
                         0D6B   4444
          53   10 A5   7D  0D6B   4445  25$:    MOVQ    CDRP$L_FR3(R5), R3      ; Restore fork registers, R3 & R4.
               0C B5   17  0D6F   4446          JMP     @CDRP$L_FPC(R5)         ; Dispatch to issuer of MSCP command
                         0D72   4447                                              ; who will return to our caller.
                         0D72   4448
                F28B'    30  0D72   4449  30$:    BSBW    DUTU$TEST_CANCEL_DONE   ; If this request completes a cancel
                         0D75   4450                                              ; operation, cleanup that operation.
                  EC     11  0D75   4451          BRB     23$                     ; Branch back to normal flow.
                         0D77   4452
                F286'    30  0D77   4453  50$:    BSBW    DUTU$DUMP_ENDMESSAGE    ; If diagnostic buffer, record MSCP
                         0D7A   4454                                              ; end message sent in the buffer.
                  EF     11  0D7A   4455          BRB     25$                     ; Branch back to normal flow.
                         0D7C   4456
                         0D7C   4457
                         0D7C   4458
                         0D7C   4459 ;
                         0D7C   4460 ; Process ATTENTION MESSAGES
                         0D7C   4461 ;
                         0D7C   4462
                         0D7C   4463 ATTN_MSG:
                   1E    BB  0D7C   4464          PUSHR   #^M<R1,R2,R3,R4>        ; Save vital registers.
          53   5C A3   D0  0D7E   4465          MOVL    CDT$L_AUXSTRUC(R3), R3  ; Get CDDB address.
               AC'AF    9F  0D82   4466          PUSHAB  B^EXIT_ATTN_MSG         ; Make DISPATCH look like a BSBx.
                         0D85   4467          DISPATCH -                      ; Dispatch to attention message
                         0D85   4468                  MSCP$B_OPCODE(R2), -    ;  specific processing:
                         0D85   4469                  type=B, prefix=MSCP$K_OP, < -
                         0D85   4470                  <AVATN, UNIT_AVAILABLE_ATTN>, -
                         0D85   4471                  <DUPUN, DUPLICATE_UNIT_ATTN>, -
                         0D85   4472                  <ACPTH, ACCESS_PATH_ATTN>, -
                         0D85   4473                  >
                         0D91   4474  INV_ATTN_MSG:                              ; Process invalid ATTENTION MESSAGE.
                   8E    D5  0D91   4475          TSTL    (SP)+                   ; Pop "return" address.
                50   0A   3C  0D93   4476          MOVZWL  #EMB$C_INVATT, R0       ; Invalid attention message type.
       00000000'GF   16  0D96   4477          JSB     G^ERL$LOG_DMSCP         ; Log incorrect DISK MSCP message.
                   1E    BA  0D9C   4478          POPR    #^M<R1,R2,R3,R4>        ; Restore vital registers.
                         0D9E   4479          DEALLOC_MSG_BUF_REG             ; Deallocate ATTN MSG buffer.
          53   5C A3   D0  0DA1   4480          MOVL    CDT$L_AUXSTRUC(R3), R3  ; Get CDDB again.
          53   18 A3   D0  0DA5   4481          MOVL    CDDB$L_CRB(R3), R3      ; From that get the CRB address.
                FCE2    31  0DA9   4482          BRW     DU$RE_SYNCH             ; Re-synchronize with controller.
                         0DAC   4483
                         0DAC   4484  EXIT_ATTN_MSG:
```

```
1E   BA  0DAC  4485          POPR    #^M<R1,R2,R3,R4>        ; Restore vital registers.
         0DAE  4486 FINISHED_WITH_MESSAGE:
         0DAE  4487          DEALLOC_MSG_BUF_REG             ; Deallocate ATTN MSG buffer.
     05  0DB1  4488          RSB                             ; Return to SCS caller.
```

```
                     0DB2  4490          .SBTTL Attention Message Processing
                     0DB2  4491          .SBTTL   - Process Unit Available Attention Message
                     0DB2  4492
                     0DB2  4493 ;++
                     0DB2  4494 ;
                     0DB2  4495 ; Functional Description:
                     0DB2  4496 ;
                     0DB2  4497 ;       This routine processes unit available attention messages.  If the
                     0DB2  4498 ;       available unit is already known in the I/O database, no action is
                     0DB2  4499 ;       taken.  If the available unit represents a second path to an already
                     0DB2  4500 ;       known unit, the I/O database is altered to show the alternate path
                     0DB2  4501 ;       availability.  If the available unit represents a totally new device,
                     0DB2  4502 ;       it is added to the I/O database.
                     0DB2  4503 ;
                     0DB2  4504 ; Inputs:
                     0DB2  4505 ;
                     0DB2  4506 ;       R1      attention message size
                     0DB2  4507 ;       R2      attention message address
                     0DB2  4508 ;       R3      CDDB address
                     0DB2  4509 ;
                     0DB2  4510 ; Outputs:
                     0DB2  4511 ;
                     0DB2  4512 ;       R0 - R5 destroyed
                     0DB2  4513 ;       All other registers preserved
                     0DB2  4514 ;--
                     0DB2  4515
                     0DB2  4516 UNIT_AVAILABLE_ATTN:
                     0DB2  4517
03 12 A3   05   E0   0DB2  4518          BBS     #CDDB$V_POLLING, -      ; Is a poll for units in progress?
                     0DB7  4519                  CDDB$W_STATUS(R3), 90$  ; Branch if poll for units active.
         F246'   30  0DB7  4520          BSBW    DUTU$NEW_UNIT           ; Process possible new unit.
                 05  0DBA  4521 90$:     RSB
```

```
                     0DBB  4523          .SBTTL   - Process Duplicate Unit Attention Message
                     0DBB  4524
                     0DBB  4525 ;++
                     0DBB  4526 ;
                     0DBB  4527 ; Functional Description:
                     0DBB  4528 ;
                     0DBB  4529 ;       This routine processes duplicate unit attention messages.
                     0DBB  4530 ;       Notification of the condition is sent to the operator's console and
                     0DBB  4531 ;       an entry is made in the error log.  If the unit described in the
                     0DBB  4532 ;       message cannot be found, an invalid MSCP message error log entry is
                     0DBB  4533 ;       generated.
                     0DBB  4534 ;
                     0DBB  4535 ; Inputs:
                     0DBB  4536 ;
                     0DBB  4537 ;       R1      attention message size
                     0DBB  4538 ;       R2      attention message address
                     0DBB  4539 ;       R3      CDDB address
                     0DBB  4540 ;
                     0DBB  4541 ; Outputs:
                     0DBB  4542 ;
                     0DBB  4543 ;       R0 - R5 destroyed
                     0DBB  4544 ;       All other registers preserved
                     0DBB  4545 ;--
                     0DBB  4546
                     0DBB  4547          .ENABLE LSB
                     0DBB  4548
                     0DBB  4549 DUPLICATE_UNIT_ATTN:
                     0DBB  4550
        F242'  30    0DBB  4551          BSBW    DUTU$LOOKUP_UCB         ; Locate UCB for this message.
53       50    D0    0DBE  4552          MOVL    R0, R3                  ; Setup UCB address.
          0C   13    0DC1  4553          BEQL    90$                     ; If no UCB found, ignore message.
        F23A'  30    0DC3  4554          BSBW    DUTU$SEND_DUPLICATE_UNIT; Send message to operator.
50       06    3C    0DC6  4555          MOVZWL  #EMB$C_DUPUN, R0        ; Setup duplicate unit error log code.
                     0DC9  4556
                     0DC9  4557 LOG_ATTENTION_MESSAGE:
00000000'GF    16    0DC9  4558          JSB     G^ERL$LOGMESSAGE        ; Error log attention message
               05    0DCF  4559 90$:     RSB                             ; and return to caller.
                     0DD0  4560
                     0DD0  4561          .DISABLE LSB
```

```
                     0DD0  4563          .SBTTL   - Process Access Path Attention Message
                     0DD0  4564
                     0DD0  4565 ;++
                     0DD0  4566 ;
                     0DD0  4567 ; Functional Description:
                     0DD0  4568 ;
                     0DD0  4569 ;       This routine processes access path attention messages.  If the access
                     0DD0  4570 ;       path represents a second path to an already known unit, the I/O
                     0DD0  4571 ;       database is altered to show the alternate path availability, and an
                     0DD0  4572 ;       entry is made in the error log indicating receipt of the message.
                     0DD0  4573 ;       If the unit described in the message cannot be found, an invalid MSCP
                     0DD0  4574 ;       message error log entry is generated.
                     0DD0  4575 ;
                     0DD0  4576 ; Inputs:
                     0DD0  4577 ;
                     0DD0  4578 ;       R1      attention message size
                     0DD0  4579 ;       R2      attention message address
                     0DD0  4580 ;       R3      CDDB address
                     0DD0  4581 ;
                     0DD0  4582 ; Outputs:
                     0DD0  4583 ;
                     0DD0  4584 ;       R0 - R5 destroyed
                     0DD0  4585 ;       All other registers preserved
                     0DD0  4586 ;--
                     0DD0  4587
                     0DD0  4588 ACCESS_PATH_ATTN:
                     0DD0  4589
   F22D'  30         0DD0  4590          BSBW    DUTU$SETUP_DUAL_PATH    ; Process possible dual path unit.
53   50  D0         0DD3  4591          MOVL    R0, R3                  ; Get UCB address.
     06  13         0DD6  4592          BEQL    90$                     ; If no UCB found, ignore the message.
         05         0DD8  4593          RSB                             ; Return w/o logging message, but
                    0DD9  4594                                          ; leave message logging code in place
                    0DD9  4595                                          ; just in case its needed.
50   08  9A         0DD9  4596          MOVZBL  #EMB$C_ACPTH, R0        ; Setup ERL$LOGMESSAGE code.
     EB  11         0DDC  4597          BRB     LOG_ATTENTION_MESSAGE   ; Join common log message path.
         05         0DDE  4598 90$:     RSB                             ; If no UCB, exit.
```

```
                        0DDF  4600          .SBTTL  DU$DGDR - Data Gram Dispatch Routine
                        0DDF  4601 ;
                        0DDF  4602 ; Inputs:
                        0DDF  4603 ;
                        0DDF  4604 ;        R1 =  length of datagram
                        0DDF  4605 ;        R2 => datagram
                        0DDF  4606 ;        R3 => CDT
                        0DDF  4607 ;        R4 => PDT
                        0DDF  4608
                        0DDF  4609 DU$DGDR:
                        0DDF  4610
          50   5C A3   D0  0DDF  4611          MOVL    CDT$L_AUXSTRUC(R3),R0   ; R0 => CDDB
               55   53   D0  0DE3  4612          MOVL    R3,R5                   ; Save pointer to CDT.
                    02   91  0DE6  4613          CMPB    #MSLG$K_DISK_TRN,-      ; Look into error log message.
                  08 A2      0DE8  4614                  MSLG$B_FORMAT(R2)       ;  See if disk transfer error.
                    0C   13  0DEA  4615          BEQL    5$                      ; EQL means yes.
                    03   91  0DEC  4616          CMPB    #MSLG$K_SDI,-           ; See if SDI error.
                  08 A2      0DEE  4617                  MSLG$B_FORMAT(R2)
                    06   13  0DF0  4618          BEQL    5$                      ; EQL means yes.
                08 A2   04   91  0DF2  4619          CMPB    #MSLG$K_SML_DSK, -      ; See if disk transfer error.
                        0DF6  4620                  MSLG$B_FORMAT(R2)
                    2C   12  0DF6  4621          BNEQ    30$                     ; NEQ means unit # field maybe invalid.
                        0DF8  4622
                        0DF8  4623 5$:                                          ; If here, then UNIT # field valid.
     50   0000007C 8F   C3  0DF8  4624          SUBL3   #<UCB$L_CDDB_LINK -     ; Get "previous" UCB address in R3.
                        0DFF  4625                   -CDDB$L_UCBCHAIN>, -
                    53      0DFF  4626                  R0, R3
                        0E00  4627
          53   00C4 C3   D0  0E00  4628 10$:    MOVL    UCB$L_CDDB_LINK(R3), R3 ; Chain to next UCB.
                    1D   13  0E05  4629          BEQL    30$                     ; No more UCBs.
               00D4 C3   B1  0E07  4630          CMPW    UCB$W_MSCPUNIT(R3),-    ; See if datagram (error log packet)
                  04 A2      0E0B  4631                  MSCP$Q_UNIT(R2)         ;  for this unit.
                    F1   12  0E0D  4632          BNEQ    10$                     ; If not, branch abck to try next unit.
                        0E0F  4633 15$:
               50   01   3C  0E0F  4634          MOVZWL  #EMB$C_DM,R0            ; Put type of message into R0.
     00000000'GF    16   0E12  4635          JSB     G^ERL$LOGMESSAGE        ; And call to log message.
                        0E18  4636 20$:
               53   55   D0  0E18  4637          MOVL    R5,R3                   ; Restore R3 => CDT.
          52   00B8 C4   C2  0E1B  4638          SUBL    PDT$L_DGOVRHD(R4),R2    ; R2 => SCS header of datagram.
                        0E20  4639          QUEUE_DG_BUF                    ; Requeue datagram buffer.
                    05   0E23  4640          RSB                             ; Return to port.
                        0E24  4641
               53   62   D0  0E24  4642 30$:    MOVL    MSCP$L_CMD_REF(R2),R3   ; Pickup RSPID from END PACKET.
                    1C   13  0E27  4643          BEQL    40$                     ; EQL means no RSPID.
                    22   BB  0E29  4644          PUSHR   #^M<R1,R5>              ; Save registers.
               55   53   D0  0E2B  4645          MOVL    R3, R5                  ; Copy RSPID.
                        0E2E  4646          FIND_RSPID_RDTE                 ; Lookup RDTE for RSPID.
               53   55   D0  0E34  4647          MOVL    R5, R3                  ; Copy RDTE address.
                    22   BA  0E37  4648          POPR    #^M<R1,R5>              ; Restore saved registers.
                 09 50   E9  0E39  4649          BLBC    R0,40$                  ; Branch if lookup error.
               53   63   D0  0E3C  4650          MOVL    RD$L_CDRP(R3),R3        ; Get CDRP address in R3.
          53   BC A3   D0  0E3F  4651          MOVL    CDRP$L_UCB(R3),R3       ; Get UCB address in R3 (if any).
                    CA   12  0E43  4652          BNEQ    15$                     ; If we get a UCB, use it.
                        0E45  4653
          53   5C A5   D0  0E45  4654 40$:    MOVL    CDT$L_AUXSTRUC(R5),R3   ; R3 => CDDB.
               50   0B   3C  0E49  4655          MOVZWL  #EMB$C_NOUNIT_DG,R0     ; Indicate type of message.
     00000000'GF    16   0E4C  4656          JSB     G^ERL$LOG_DMSCP         ; Call to log message.
```

```
C4   11  0E52  4657          BRB     20$                 ; Rejoin code.
```

```
                     0E54  4659          .SBTTL  INVALID_STS
                     0E54  4660
                     0E54  4661 ;+
                     0E54  4662 ; We come here if we get an invalid MSCP status.  We log the MSCP message
                     0E54  4663 ; and then RE-SYNCH the controller.
                     0E54  4664 ;
                     0E54  4665 ; Inputs:
                     0E54  4666 ;       R2 => MSCP packet
                     0E54  4667 ;       R3 => UCB
                     0E54  4668 ;       R4 => PDT
                     0E54  4669 ;       R5 => CDRP
                     0E54  4670 ;       CDRP$L_ENDMSGSIZ(R5) => length of MSCP packet with invalid status
                     0E54  4671 ;
                     0E54  4672
                     0E54  4673 INVALID_STS:
                     0E54  4674
    50    09   3C    0E54  4675          MOVZWL  #EMB$C_INVSTS, R0       ; Indicate type of record to log.
  51    46 A5   3C    0E57  4676          MOVZWL  CDRP$W_ENDMSGSIZ(R5), R1; Pickup length of faulty packet.
53    00BC C3   D0    0E5B  4677          MOVL    UCB$L_CDDB(R3), R3      ; Get CDDB address for logging error.
 00000000'GF   16    0E60  4678          JSB     G^ERL$LOG_DMSCP         ; Log disk MSCP error.
      F197'   30    0E66  4679          BSBW    DUTU$INSERT_RESTARTQ    ; Queue CDRP for retry.
  53    18 A3   D0    0E69  4680          MOVL    CDDB$L_CRB(R3),R3       ; R3 => CRB for re-SYNCH.
      FC1E   31    0E6D  4681          BRW     DU$RE_SYNCH             ; Zap controller.
                     0E70  4682
                     0E70  4683
                     0E70  4684
                     0E70  4685          .END
```

```
$$$                             = 00000020 R     04
$$BASE                          = 00000040
$$BEGIN$$                       = 00000002
$$DISPL                         = 00000043
$$GENSW                         = 00000001
$$HIGH                          = 00000042
$$LIMIT                         = 00000002
$$LOW                           = 00000040
$$MEDIA$$                       = 6B1AC00B
$$MNSW                          = 00000001
$$MXSW                          = 00000001
$$N$$                           = 0000000B
$$OP                            = 00000002
$$S$$                           = 00000002
$$TEMP$$                        = FFFFFFF1
ACCESS_PATH_ATTN                  00000DD0 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
AFTER_MAP                         00000911 R     05
ALLOC_DELTA                     = 00000001
ATS_NULL                        = 00000005
ATE_MSCPCODE                      00000002
ATE_OFFSET                        00000000
ATE_SSCODE                        00000003
ATTN_MSG                          00000D7C R     05
AVAILABLE_ABORT                   00000850 R     05
AVAILABLE_CTRLERR                 00000850 R     05
AVAILABLE_DRVERR                  00000850 R     05
AVAILABLE_MEDOFL                  00000850 R     05
AVAILABLE_SSSC                    00000850 R     05
AVAILABLE_SUCC                    00000850 R     05
AVAIL_IVCMD                       00000848 R     05
AVAIL_IVCMD_END                   0000084E R     05
BUG$_DISKCLASS                    ******** X   05
CDDB$A_2PFKB                      00000174
CDDB$A_DAPCDRP                    00000194
CDDB$A_DAPIRP                     00000134
CDDB$A_PRMCDRP                    00000DD0
CDDB$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_CDRPQBL                  = 00000004
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_QUORLOST                         = 00000200
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$V_NOCONN                           = 00000007
CDDB$V_POLLING                          = 00000005
CDDB$V_QUORLOST                         = 00000009
CDDB$V_RESYNCH                          = 00000004
CDDB$V_RSTRTWAIT                        = 00000008
CDDB$V_SNGLSTRM                         = 00000000
CDDB$W_CNTRLFLGS                        = 00000028
CDDB$W_CNTRLTMO                         = 0000002A
CDDB$W_RSTRTCNT                         = 0000003A
CDDB$W_STATUS                           = 00000012
CDDB$W_WTUCBCTR                         = 0000005E
CDRP$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_ERLIP                    = 00000004
CDRP$Q_NT_PRVMSK                = FFFFFFE0
CDRP$S_LBUFHNDL                 = 0000000C
CDRP$T_LBUFHNDL                 = 00000030
CDRP$V_CAND                     = 00000000
CDRP$V_ERLIP                    = 00000002
CDRP$V_IVCMD                    = 00000008
CDRP$V_PERM                     = 00000003
CDRP$W_ABCNT                    = FFFFFFE0
CDRP$W_BCNT                     = FFFFFFD2
CDRP$W_BOFF                     = FFFFFFD0
CDRP$W_CDRPSIZE                 = 00000008
CDRP$W_CHAN                     = FFFFFFC8
CDRP$W_DUTUCNTR                 = 00000044
CDRP$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                   00000179 R      05
CLU$GL_ALLOCLS                    ******** X    05
CONNECT_DELTA                   = 0000000A
CRB$L_AUXSTRUC                  = 00000010
CRB$L_DUETIME                   = 00000018
CRB$L_INTD                      = 00000024
CRB$L_TOUTROUT                  = 0000001C
DBL_WAIT_UCB                      00000372 R      05
DC$_DISK                        = 00000001
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$M_CDP                       = 00000008
DEV$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_RCT                       = 00000100
DEV$M_RND                       = 10000000
DEV$M_SHR                       = 00010000
DEV$V_2P                        = 00000004
DEV$V_RCT                       = 00000008
DEV$V_SSM                       = 00000006
DISCONNECT_REASON               = 00000001
DO_MAP                            000008F7 R      05
DPT$C_LENGTH                    = 00000038
DPT$C_VERSION                   = 00000004
DPT$INITAB                        00000038 R      04
DPT$M_NOUNLOAD                  = 00000004
DPT$M_SCS                       = 00000008
DPT$REINITAB                      0000007C R      04
DPT$TAB                           00000000 R      04
DSE_IVCMD_END                     000009F8 R      05
DT$_CDR50                       = 00000022
DT$_ML11                        = 00000011
DT$_RA60                        = 00000016
DT$_RA80                        = 00000014
DT$_RA81                        = 00000015
DT$_RA82                        = 0000001E
DT$_RB02                        = 00000012
DT$_RB80                        = 00000013
DT$_RC25                        = 00000017
DT$_RC26                        = 0000001F
DT$_RCF25                       = 00000018
DT$_RCF26                       = 00000020
DT$_RD51                        = 00000019
DT$_RD52                        = 0000001B
DT$_RD53                        = 0000001C
DT$_RK06                        = 00000001
DT$_RK07                        = 00000002
DT$_RL01                        = 00000009
DT$_RL02                        = 0000000A
DT$_RM03                        = 00000006
DT$_RM05                        = 0000000F
DT$_RM80                        = 0000000D
DT$_RP04                        = 00000003
DT$_RP05                        = 00000004
DT$_RP06                        = 00000005
DT$_RP07                        = 00000007
DT$_RP07HT                      = 00000008
DT$_RX01                        = 00000010
DT$_RX02                        = 0000000B
DT$_RX04                        = 0000000C
DT$_RX18                        = 00000025
DT$_RX31                        = 00000023
DT$_RX32                        = 00000024
DT$_RX50                        = 0000001A
DT$_TU58                        = 0000000E
DU$CONNECT_ERR                    00000AA2 R      05
DU$DDT                            00000000 RG     05
DU$DGDR                           00000DDF R      05
DU$DISCONNECT_HIRT                ******** X    05
DU$DSE_FDT                        000003F0 R      05
```

```
DU$IDR                          00000D1E RG    05
DU$INIT_HIRT                    ********   X   05
DU$LOCK_HIRT                    ********   X   05
DU$MOUNTVER                     00000376 R     05
DU$ONLINE_COMPLETE              ********   X   05
DU$REPLACE_LBN                  ********   X   05
DU$RE_SYNCH                     00000A8E R     05
DU$SHADOW_PACKACK_FDT           00000400 R     05
DU$TMR                          00000C7A R     05
DU$UNLOCK_HIRT                  ********   X   05
DUPLICATE_UNIT_ATTN             00000DBB 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$DUMP_COMMAND               ********   X   05
DUTU$DUMP_ENDMESSAGE            ********   X   05
DUTU$END                        ********   X   04
DUTU$FAILOVER_UCB               ********   X   05
DUTU$GET_DEVTYPE                ********   X   05
DUTU$INIT_CONN_UCB              ********   X   05
DUTU$INIT_MSCP_MSG              ********   X   05
DUTU$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$TERMINATE_PENDING          ********   X   05
DUTU$TEST_CANCEL_DONE           ********   X   05
DUTU$UNITINIT                   ********   X   05
DU_BEGIN_IVCMD                  00000543 R     05
DU_BEGIN_MNTVER                 0000037A R     05
DU_CONTROLLER_INIT              000000D8 R     05
DU_END_MNTVER                   000003B1 R     05
DU_FUNCTABLE                    00000038 R     05
DU_RESTARTIO                    00000504 R     05
DU_REVALIDATE_DISKS             000002F7 R     05
DU_STARTIO                      000004D5 R     05
DYN$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$C_DM                      = 00000001
EMB$C_DUPUN                   = 00000006
EMB$C_INVATT                  = 0000000A
EMB$C_INVSTS                  = 00000009
EMB$C_NOUNIT_DG               = 0000000B
END_PACKACK                     000006BC R     05
END_SINGLE_STREAM               00000C45 R     05
ERL$LOGMESSAGE                  ********   X   05
ERL$LOGSTATUS                   ********   X   05
ERL$LOG_DMSCP                   ********   X   05
EXE$FINISHIOC                   ********   X   05
EXE$FORK                        ********   X   05
EXE$GL_ABSTIM                   ********   X   05
EXE$GQ_SYSTIME                  ********   X   05
EXE$INSIOQ                      ********   X   05
EXE$LCLDSKVALID                 ********   X   05
EXE$MOUNTVER                    ********   X   05
EXE$QIODRVPKT                   ********   X   05
EXE$SENSEMODE                   ********   X   05
EXE$SETCHAR                     ********   X   05
EXE$ZEROPARM                    ********   X   05
EXIT_ATTN_MSG                   00000DAC R     05
FINISHED_WITH_MESSAGE           00000DAE R     05
FKB$K_LENGTH                  = 00000018
FUNCTAB_LEN                   = 000000A0
FUNCTION_EXIT                   00000957 R     05
HOST_TIMEOUT                  = 0000001E
HSTIMEOUT_ARRAY                 00000199 R     05
INI$BRK                         ********   X   05
INITIAL_CREDIT                = 0000000A
INITIAL_DG_COUNT              = 00000002
INIT_IMMED_DELTA              = 0000001E
INIT_TIMEOUT                    00000176 R     05
INVALID_STS                     00000E54 R     05
INV_ATTN_MSG                    00000D91 R     05
IO$M_DATACHECK                = 00004000
IO$M_EXPRESS                  = 00000080
IO$M_FCODE                    = 0000003F
IO$M_FORCERR                  = 00000100
IO$M_INHRETRY                 = 00008000
IO$S_FCODE                    = 00000006
IO$V_DATACHECK                = 0000000E
IO$V_EXPRESS                  = 00000007
IO$V_FCODE                    = 00000000
IO$V_FORCERR                  = 00000008
IO$V_INHRETRY                 = 0000000F
IO$V_SHADOW                   = 00000006
IO$_ACCESS                    = 00000032
IO$_ACPCONTROL                = 00000038
IO$_AVAILABLE                 = 00000011
IO$_CREATE                    = 00000033
IO$_DEACCESS                  = 00000034
IO$_DELETE                    = 00000035
```

```
IO$_DSE                 =  00000015
IO$_MODIFY              =  00000036
IO$_MOUNT               =  00000039
IO$_NOP                 =  00000000
IO$_PACKACK             =  00000008
IO$_READLBLK            =  00000021
IO$_READPBLK            =  0000000C
IO$_READVBLK            =  00000031
IO$_SENSECHAR           =  0000001B
IO$_SENSEMODE           =  00000027
IO$_SETCHAR             =  0000001A
IO$_SETMODE             =  00000023
IO$_UNLOAD              =  00000001
IO$_VIRTUAL             =  0000003F
IO$_WRITECHECK          =  0000000A
IO$_WRITELBLK           =  00000020
IO$_WRITEPBLK           =  0000000B
IO$_WRITEVBLK           =  00000030
IOC$ALTREQCOM           ********    X   05
IOC$CVTLOGPHY           ********    X   05
IOC$GL_DU_CDDB          ********    X   06
IOC$RETURN              ********    X   05
IO_STALLED              00000494 R      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$L_DUTUFLAGS         =  000000A0
IRP$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$M_DIAGBUF           =  00000080
IRP$M_PHYSIO            =  00000100
IRP$Q_NT_PRVMSK         =  00000040
IRP$V_MVIRP             =  0000000D
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$W_STS               =  0000002A
LOG_ATTENTION_MESSAGE   00000DC9 R      05
LOG_FINAL_STATUS        00000987 R      05
MAKE_CONNECTION         000001A3 R      05
MASKH                   =  00000008
MASKL                   =  04000000
MAX_RETRY               =  00000002
MIN_SEND_CREDIT         =  00000002
MSCP$B_BUFFER           =  00000010
MSCP$B_CNT_ALCS         =  00000004
MSCP$B_FLAGS            =  00000009
MSCP$B_OPCODE           =  00000008
MSCP$B_RBNS             =  0000002E
MSCP$B_RCT_CPYS         =  0000002F
MSCP$K_CM_EMULA         =  00000004
MSCP$K_CM_HSC50         =  00000001
MSCP$K_CM_RC25          =  00000003
MSCP$K_CM_RDRX          =  00000007
MSCP$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$K_OP_GTCMD         =  00000002
MSCP$K_OP_GTUNT         =  00000003
MSCP$K_OP_ONLIN         =  00000009
MSCP$K_OP_READ          =  00000021
MSCP$K_OP_STCON         =  00000004
MSCP$K_OP_WRITE         =  00000022
MSCP$K_SC_DUPUN         =  00000004
MSCP$K_SC_INOPR         =  00000002
MSCP$K_SC_NOVOL         =  00000001
MSCP$K_SC_ODDBC         =  00000002
MSCP$K_SC_UNKNO         =  00000000
MSCP$K_ST_ABRTD         =  00000002
MSCP$K_ST_AVLBL         =  00000004
MSCP$K_ST_CNTLR         =  0000000A
MSCP$K_ST_COMP          =  00000007
MSCP$K_ST_DATA          =  00000008
MSCP$K_ST_DRIVE         =  0000000B
MSCP$K_ST_HSTBF         =  00000009
```

```
MSCP$K_ST_ICMD          = 00000001
MSCP$K_ST_MFMTE         = 00000005
MSCP$K_ST_OFFLN         = 00000003
MSCP$K_ST_SHST          = 0000000C
MSCP$K_ST_SUCC          = 00000000
MSCP$K_ST_WRTPR         = 00000006
MSCP$L_BYTE_CNT         = 0000000C
MSCP$L_CMD_REF          = 00000000
MSCP$L_CMD_STS          = 00000010
MSCP$L_DEV_PARM         = 0000001C
MSCP$L_EXCL_LBA         = 00000014
MSCP$L_LBN              = 0000001C
MSCP$L_MEDIA_ID         = 0000001C
MSCP$L_OUT_REF          = 0000000C
MSCP$L_UNT_SIZE         = 00000024
MSCP$L_VOL_SER          = 00000028
MSCP$M_EF_BBLKR         = 00000080
MSCP$M_EF_ERLOG         = 00000020
MSCP$M_MD_COMP          = 00004000
MSCP$M_MD_ERROR         = 00001000
MSCP$M_MD_EXPRS         = 00008000
MSCP$M_MD_SEREC         = 00000100
MSCP$M_MD_SHDSP         = 00000010
MSCP$M_MD_SPNDW         = 00000002
MSCP$M_OP_END           = 00000080
MSCP$M_SHADOW           = 00008000
MSCP$M_ST_MASK          = 0000001F
MSCP$M_UF_WRTPD         = 00000100
MSCP$M_UF_WRTPH         = 00002000
MSCP$M_UF_WRTPS         = 00001000
MSCP$Q_CNT_ID           = 00000014
MSCP$Q_TIME             = 00000014
MSCP$Q_UNIT_ID          = 00000014
MSCP$S_ST_MASK          = 00000005
MSCP$S_ST_SBCOD         = 0000000B
MSCP$V_CF_MLTHS         = 00000002
MSCP$V_CF_REPLC         = 0000000F
MSCP$V_CF_SHADW         = 00000001
MSCP$V_EF_ERLOG         = 00000005
MSCP$V_MD_COMP          = 0000000E
MSCP$V_MD_ERROR         = 0000000C
MSCP$V_MD_EXPRS         = 0000000F
MSCP$V_MD_SEREC         = 00000008
MSCP$V_SC_ALONL         = 00000008
MSCP$V_SC_INOPR         = 00000006
MSCP$V_SHADOW           = 0000000F
MSCP$V_ST_MASK          = 00000000
MSCP$V_ST_SBCOD         = 00000005
MSCP$V_UF_576           = 00000002
MSCP$V_UF_REPLC         = 0000000F
MSCP$V_UF_SSMEM         = 0000000E
MSCP$V_UF_WRTPD         = 00000008
MSCP$V_UF_WRTPH         = 0000000D
MSCP$V_UF_WRTPS         = 0000000C
MSCP$W_CNT_FLGS         = 0000000E
MSCP$W_CNT_TMO          = 00000010
MSCP$W_CORT_SPD         = 00000022
MSCP$W_CYLINDER         = 00000028
MSCP$W_EXCL_LBC         = 00000018
MSCP$W_GROUP            = 00000026
MSCP$W_HST_TMO          = 00000010
MSCP$W_MODIFIER         = 0000000A
MSCP$W_RCT_SIZE         = 0000002C
MSCP$W_SHDW_UNT         = 00000020
MSCP$W_STATUS           = 0000000A
MSCP$W_TRACK            = 00000024
MSCP$W_UNIT             = 00000004
MSCP$W_UNT_FLGS         = 0000000E
MSCP_SRVR_NAME            00000189 R      05
MSG_BUF_FAILURE           0000058A R      05
MSLG$B_FORMAT           = 00000008
MSLG$K_DISK_TRN         = 00000002
MSLG$K_SDI              = 00000003
MSLG$K_SML_DSK          = 00000004
NEXT_REVAL_UCB            000002FF R      05
NOP_AVAIL                 000005E5 R      05
NOP_CTRLERR               000005E5 R      05
NOP_DRVERR                000005E5 R      05
NOP_IVCMD                 000005DC R      05
NOP_OFFLINE               000005E5 R      05
NOP_SSSC                  000005E5 R      05
NOP_SUCC                  000005E5 R      05
NORMAL_TRANSFEREND        00000957 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$W_SIZE              = 00000008
P1                      = 00000000
P2                      = 00000004
P3                      = 00000008
P4                      = 0000000C
P5                      = 00000010
P6                      = 00000014
PACKACK_576               000006D5 R      05
PACKACK_BADRCT            000006E1 R      05
PACKACK_CANCEL            000006DC R      05
PACKACK_GTUNT_SUCC        0000068D R      05
PACKACK_IVCMD             000006FA R      05
PACKACK_IVCMD_END         00000700 R      05
PACKACK_OFFLINE           00000704 R      05
PACKACK_SUCC              00000661 R      05
PB$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
PDT$L_SNDCNTMSG         = 00000060
```

```
PHYIO_VOLINV                    00000526 R      05
PHYSICAL                        000008DB R      05
PHYS_IO                         0000086C R      05
PKAK_IVCMD_END                  0000065E R      05
PR$_IPL                       = 00000012
PRP_STCON_MSG                   0000028B R      05
QUEUE_IRP                       0000048A R      05
RD$L_CDRP                     = 00000000
RD$W_SEQNUM                   = 00000006
RDT$C_MAXRDIDX                = FFFFFFF8
RECONN_COMMON                   00000AA6 R      05
RECORD_ONLINE                   00000742 R      05
RECORD_STCON                    000002BF R      05
RECORD_UNIT_STATUS              00000764 R      05
RESTART_FIRST_CDRP              00000B14 R      05
RESTART_NEXT_CDRP               00000C10 R      05
REVAL_STARTED                   00000371 R      05
SCS$ALOC_RSPID                  ********   X    05
SCS$CONNECT                     ********   X    05
SCS$DEALL_RSPID                 ********   X    05
SCS$DISCONNECT                  ********   X    05
SCS$FIND_RDTE                   ********   X    05
SCS$GL_RDT                      ********   X    05
SCS$LKP_RDTCDRP                 ********   X    05
SCS$LKP_RDTWAIT                 ********   X    05
SCS$RECVL_RSPID                 ********   X    05
SCS$UNSTALUCB                   ********   X    05
SINGLE_STREAM                   0000098F R      05
SS$_ABORT                     = 0000002C
SS$_BADRCT                    = 0000216C
SS$_CTRLERR                   = 00000054
SS$_DATACHECK                 = 0000005C
SS$_DEVOFFLINE                = 00000084
SS$_DRVERR                    = 0000008C
SS$_DUPUNIT                   = 000021C4
SS$_FORCEDERROR               = 00002144
SS$_FORMAT                    = 000000BC
SS$_ILLIOFUNC                 = 000000F4
SS$_IVADDR                    = 00000134
SS$_IVBUFLEN                  = 0000034C
SS$_MEDOFL                    = 000001A4
SS$_NORMAL                    = 00000001
SS$_PARITY                    = 000001F4
SS$_SHACHASTA                 = 00002284
SS$_VOLINV                    = 00000254
SS$_WRITLCK                   = 0000025C
START_AVAILABLE                 0000080E R      05
START_DSE                       0000085E R      05
START_LOCAL_DEVICE              000004C0 R      05
START_MOUNT_VER                 00000456 R      05
START_NOP                       000005AA R      05
START_NOSUCH                    0000057D R      05
START_PACKACK                   000005EA R      05
START_READPBLK                  000008E6 R      05
START_READWRITE                 000008EA R      05
START_UNLOAD                    0000080A R      05
START_WITH_MODIFIERS            000008B0 R      05
START_WRITECHECK                0000089A R      05
START_WRITEPBLK                 000008E0 R      05
TEST_PHYSIO                     000008F1 R      05
TRANSFER_DATA_ERROR             00000A29 R      05
TRANSFER_HOST_BUFFER_ERROR      00000A1B R      05
TRANSFER_INVALID_COMMAND        000009E8 R      05
TRANSFER_IVCMD_END              000009EE R      05
TRANSFER_MEDOFL                 00000A0C R      05
TRANSFER_MSCP_ERROR             0000099F R      05
TRANSFER_REPLACE                00000A3E R      05
TRANSFER_RTN_BCNT               00000947 R      05
TRANSFER_SHIFT                  0000094B R      05
UCB$B_DEVCLASS                = 00000040
UCB$B_DIPL                    = 0000005E
UCB$B_DU_RBNPTRK                000000FB   G
UCB$B_DU_RCTCPYS                000000FA   G
UCB$B_FIPL                    = 0000000B
UCB$B_SECTORS                 = 00000044
UCB$B_TRACKS                  = 00000045
UCB$B_TYPE                    = 0000000A
UCB$K_DU_LENGTH               = 00000102
UCB$K_LENGTH                  = 00000090
UCB$K_MSCP_DISK_LENGTH        = 000000EC
UCB$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_DU_TOTSZ                  000000F4   G
UCB$L_DU_USIZE                  000000F0   G
UCB$L_DU_VOLSER                 000000EC   G
UCB$L_IOQBL                   = 00000050
UCB$L_IOQFL                   = 0000004C
UCB$L_LINK                    = 00000030
UCB$L_MAXBLOCK                = 000000B0
UCB$L_MEDIA_ID                = 0000008C
UCB$L_MSCPDEVPARAM            = 000000D8
UCB$L_PDT                     = 00000084
UCB$L_STS                     = 00000064
UCB$L_WAIT_CDDB               = 000000DC
UCB$M_BSY                     = 00000100
UCB$M_CLUTRAN                 = 00200000
UCB$M_MSCP_INITING            = 00000200
UCB$M_MSCP_MNTVERIP           = 00000100
UCB$M_MSCP_PKACK              = 00001000
UCB$M_MSCP_WAITBMP            = 00000400
UCB$M_MSCP_WRTP               = 00002000
UCB$M_NOCNVRT                 = 00000004
UCB$M_ONLINE                  = 00000010
UCB$M_SUPMVMSG                = 00040000
UCB$M_VALID                   = 00000800
UCB$Q_UNIT_ID                 = 000000CC
UCB$V_BSY                     = 00000008
UCB$V_CLUTRAN                 = 00000015
UCB$V_MSCP_MNTVERIP           = 00000008
UCB$V_MSCP_PKACK              = 0000000C
```

```
UCB$V_MSCP_WAITBMP              =  0000000A
UCB$V_MSCP_WRTP                 =  0000000D
UCB$V_VALID                     =  0000000B
UCB$W_CYLINDERS                 =  00000046
UCB$W_DEVBUFSIZ                 =  00000042
UCB$W_DEVSTS                    =  00000068
UCB$W_DU_GRPPCYL                   00000100  G
UCB$W_DU_LBNPTRK                   000000FC  G
UCB$W_DU_RCTSIZE                   000000F8  G
UCB$W_DU_TRKPGRP                   000000FE  G
UCB$W_MSCPUNIT                  =  000000D4
UCB$W_RWAITCNT                  =  00000056
UCB$W_SIZE                      =  00000008
UCB$W_STS                       =  00000064
UCB$W_UNIT_FLAGS                =  000000E0
UNDO_ONLINE                        000006E6  R      05
UNIT_AVAILABLE_ATTN                00000DB2  R      05
VALID_PACKACK                      000006B8  R      05
VEC$L_INITIAL                   =  0000000C
VOLNOTVAL                          000005A0  R      05
VOL_INVALID                        0000058D  R      05
XFER_IVCMD_END                     000008DD  R      05
XFER_NORMAL_END                    00000A8B  R      05
XFER_REPLACE                       0000099C  R      05
XFER_RTN_BCNT                      00000A3B  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
$$$200_TEMPLATE_UCB_01	00000102 (258.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$200_TEMPLATE_ORB_01	00000058 (88.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
$$$105_PROLOGUE	00000087 (135.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
$$$115_DRIVER	00000E70 (3696.)	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	000000AF (175.)	07 (7.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.03	00:00:02.09
Command processing	137	00:00:00.43	00:00:03.27
Pass 1	1388	00:00:37.10	00:02:25.38
Symbol table sort	0	00:00:03.73	00:00:10.58
Pass 2	458	00:00:09.84	00:00:34.53
Symbol table output	2	00:00:00.40	00:00:00.79
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	2025	00:00:51.56	00:03:16.82

The working set limit was 3000 pages.
302915 bytes (592 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3409 non-local and 113 local symbols.
4685 source lines were read in Pass 1, producing 41 object records in Pass 2.
95 pages of virtual memory were used to define 88 macros.

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

Macro library name	Macros defined
_255DUA28:[DRIVER.OBJ]DUTULIB.MLB;1	16
_255DUA28:[SYS.OBJ]LIB.MLB;1	52
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	79

3909 GETS were required to define 79 macros.

There were no errors, warnings or information messages.

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

DUHIRT
LIS

DUDRIVER
LIS